МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
_____
(підпис)
« » червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Сервіс супроводу локалізації додатків»
здобувача групи ІН – 01 Дяговця Валерія Олексійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Валерій ДЯГОВЕЦЬ
(підпис)

Керівник, старший викладач
комп'ютерних наук, кандидат
технічних наук, доцент Борис КУЗІКОВ _____
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-01 Дяговця Валерія

1. Тема роботи: «Сервіс супроводу локалізації додатків»
затверджую наказом по СумДУ від *«22» квітня 2024 р. № 0414-VI*

2. Термін здачі здобувачем кваліфікаційної роботи *до 01 червня 2024 року*

3. Вхідні дані до кваліфікаційної роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для локалізації додатків. 3) Розробка інтелектуальної системи для перекладу додатків 4) Аналіз результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «06» травня 2024 р.

Завдання прийняв до виконання Керівник _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*	06.05.24-08.05.24	
2	*Огляд технологій, що використовуються для локалізації додатків*	08.05.24-10.05.24	
3	*Розробка інтелектуальної системи для перекладу додатків*	10.05.24-16.05.24	

4	*Аналіз результатів*	16.05.24- 20.05.24	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	20.05.24- 31.05.24	

Здобувач вищої освіти Керівник (підпис)

АНОТАЦІЯ

Записка: 52 стр., 25 рис., 1 додаток, 25 використаних джерел.

Обґрунтування актуальності теми роботи – глобалізація є загально тенденцією на світових ринках. IT компанії створюючи продукт стикаються із необхідністю його адоптації для різних мовних груп та культур. Тож розробка інструментів, які автоматизують і оптимізують ці процеси є актуальною.

Об'єкт дослідження —локалізація програмного забезпечення.

Мета роботи — реалізація сервісу створення та підтримки перекладу мовних файлів у форматах YAML, JSON, XML із застосуванням гибридної моделі перекладу для оптимізації витрат.

Методи дослідження — моделі та методи локалізації додатків, моделі і методи машинного перекладу.

Результати — реалізовано сервіс що дозволяє створювати і підтримувати переклади мовних фалів у форматі JSON, YAML, XML. Для оптимізації витрат реалізовано гібридну модель перекладу, де першою ланкою отримуються результати отримані завдяки колаборативному підходу. За відсутності результатів застосовується машинний переклад із використанням DeepL API. Сервіс дозволяє редагувати отримані результати та зберігати їх у різних форматах.

СЕРВІС, ЛОКАЛІЗАЦІЯ ДАНИХ, JAVA, POSTGRESQL, DEEPL API

ЗМІСТ

ВСТУП

Актуальність. Глобалізація є загально тенденцією на світових ринках. ІТ компанії створюючи продукт стикаються із необхідністю його адоптації для різних мовних груп та культур. Локалізація дає можливість компаніям з усього світу адаптувати свій продукт для цільової аудиторії, що надає їм більш комфортний досвід користування продуктом, а отже буде збільшуватися аудиторія. Досить великою перевагою локалізованого продукту є конкурентоспроможність. Користувачі люблять продукти, які схожі та працюють як місцеві, тобто локалізовані. Також важливо зазначити, що використання сервісу зменшує час і витрати на адаптацію продукту для різних ринків. Тож розробка інструментів, які автоматизують і оптимізують ці процеси є актуальною.

Об’єкт дослідження: локалізація програмного забезпечення.

Предмет дослідження: теоретичні та практичні аспекти реалізації сервісу локалізацію мовних файлів із гібридною моделлю перекладу.

Мета дослідження – реалізація сервісу створення та підтримки перекладу мовних файлів у форматах YAML, JSON, XML із застосуванням гибридної моделі перекладу для оптимізації витрат.

Гіпотеза: застосування перекладу на основі колаборативного підходу, автоматизованого перекладу із Deep L та ручного редагування файлів дозволяє зменшити витрати на локалізацію продуктів.

Новизна: розроблено гібридну модель перекладу, що дозволяє використати колаборативний підхід та автоматизований переклад як витрат праці для локалізації продукту.

Структура: Дане робота складається зі вступу, аналітичного огляду, постановки задачі, вибір методу розв’язання поставленої задачі, опису програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

1. АНАЛІТИЧНИЙ ОГЛЯД

1.1. Огляд існуючих способів локалізації

Існує декілька різних способів локалізації додатків. Кожен метод має свої особливості та випадки застосування, які залежать від потреб проекту. Нижче наведені основні способи локалізації, а також їх переваги та недоліки.

1.1.1. Локалізація програмних додатків внутрішнім способом

Суть цього методу полягає у тому, що ресурси, які локалізовано, включені в самі ресурсні файли, або додаток. Виходить що для кожної локалізації потрібно створювати окремий додаток, оскільки усі тексти для однієї з локалізацій знаходяться у файлах самого додатка.

Для цього треба створити набір локалізованих ресурсів, які можуть містити тексти, зображення, звуки та інші ресурси, для кожної мови локалізації окремо. Далі для кожної з мови треба створити окремий додаток, додавши до коду ресурси локалізації саме для цієї мови.

Стосовно переваг, то до них можна віднести високу швидкість доступу до локалізованих даних, бо вони включені у код додатка, та зручність для додатків з малою кількістю локалізованих ресурсів. Ще цей метод буде корисним у тому випадку якщо локалізовані ресурси будуть оновлюватися не часто. Проте цей метод має і недоліки. Коли ми додаємо локалізовані файли в додаток, то збільшуємо його розмір. Це в свою чергу може вплинути на швидкість роботи додатка. Також через те, що при оновлені додатка треба оновлювати одразу всі версії додатку, то процес оновлення стає складним

Приклад використання на мові Python (див. рис. 1.1, 1.2). Результатом роботи програми будуть:

```
Hello, World!
Привіт, світ!
Goodbye!
```

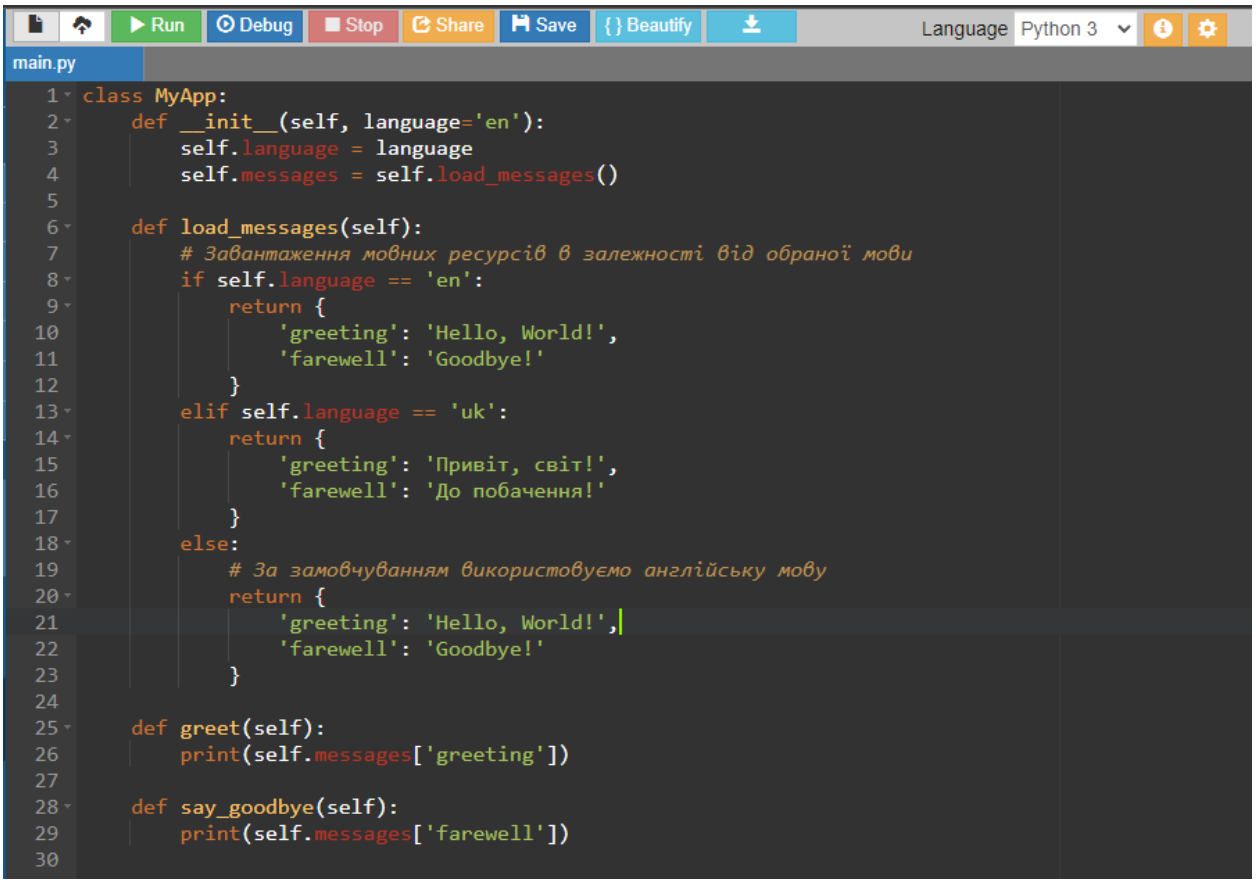

Рисунок 1.1 – Код з локалізованими ресурсами всередені

```
# Приклад використання:
# Створюємо додаток з англійською мовою за замовчуванням
app_en = MyApp()

# Виводимо англійський привітання
app_en.greet()

# Створюємо додаток з українською мовою
app_uk = MyApp(language='uk')

# Виводимо українське привітання
app_uk.greet()

# Виводимо англійське прощання
app_en.say_goodbye()
```

Рисунок 1.2 - Створення додатків під різну мову

1.1.2. Локалізація програмних додатків зовнішнім способом

Сутність цього способу полягає у тому, щоб зберігати файли локалізації окремо від додатку. Наприклад у файлах (JSON, XML, JSON) або базі даних. Дозволяє динамічно змінювати мову у додатку.

Для початку всі локалізовані ресурси заносимо в окремо створений файл або базу даних. Далі необхідно створити логічну структуру для зберігання локалізованих ресурсів, для зручного керування. Потім до коду додатку ми додаємо можливість динамічно змінювати мову користування, підключачи обрану мову з файлу.

Плюси цього методу локалізації: зменшення розміру додатка, тому що ми виносимо локалізовані ресурси до окремого файлу, зручність оновлення і розширення локалізації без того, щоб змінювати код додатку. Але виносячи усі ресурси локалізації окремо – збільшується час завантаження ресурсів, бо вони завантажуються під час роботи додатка. (дивись рисунок 1.3 – 1.5)

```yaml
eng:
  greeting: Greetings. This is an example of application localisation using a YAML file.
ukr:
  greeting: Вітаю Вас. Це програма приклад локалізації додатків за допомогою YAML файлу.
ger:
  greeting: Herzlichen Glückwunsch! Dies ist ein Beispiel für die Lokalisierung einer Anwendung mit Hilfe einer YAML-Datei.
```

Рисунок 1.3 - Простий приклад YAML файлу

```
Choose the language:
1.English      2.Українська      3.Deutsch
2
Вітаю Вас. Це програма приклад локалізації додатків за допомогою YAML файлу.

Process finished with exit code 0
```

Рисунок 1.4 - Результати роботи програми

1.1.3. Локалізація додатків внутрішнім та зовнішнім способами

Коли ми локалізуємо додаток комбінованим способом, то локалізацію через внутрішні ресурси ми використовуємо для основних та незмінних текстів, а зовнішній метод локалізації ресурсів – для змінних текстів.

Перевагами цього методу є зручність та ефективність підтримки для текстів, що збережені статично, а також перевагами є те, що легко оновлювати локалізовані тексти в зовнішніх ресурсах без необхідності змінювати код додатка.

Приклад роботи програми (див. рис. 1.6 – 1.7)

```java
package org.example;

import org.yaml.snakeyaml.Yaml;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Map;
import java.util.Scanner;

public class App
{
    public static void main(String[] args) {
        try (InputStream inputStream = new FileInputStream("D:\\untitled1\\localization.yaml")) {
            Yaml yaml = new Yaml();
            Scanner scanner = new Scanner(System.in);
            Map<String, Map<String, String>> localizationData = yaml.load(inputStream);

            System.out.println("Choose the language:");
            System.out.println("1.English      2.Українська      3.Deutsch");

            int lang = scanner.nextInt();
            while(lang != 1 && lang != 2 && lang != 3){
                System.out.println("You choose incorrect language, please choose the language");
            }

            String currentLanguage = "";

            if(lang == 1){
                currentLanguage = "eng";
            }
            if(lang == 2){
                currentLanguage = "ukr";
            }
            if(lang == 3){
                currentLanguage = "ger";
            }

            String greeting = localizationData.get(currentLanguage).get("greeting");
            System.out.println(greeting);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Рисунок 1.5 - Код з локалізованими ресурсами зовнішнім способом

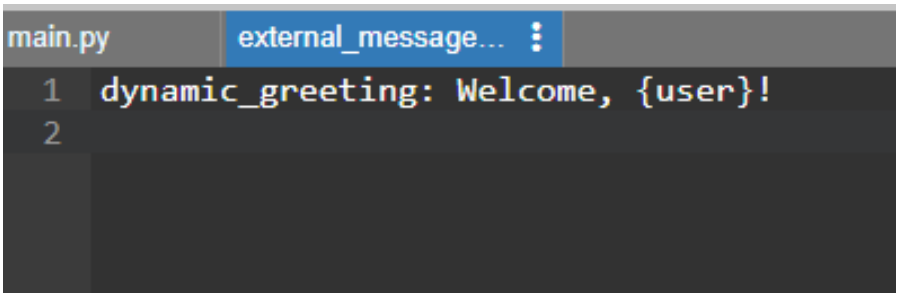

Рисунок 1.6 - external_message.yaml

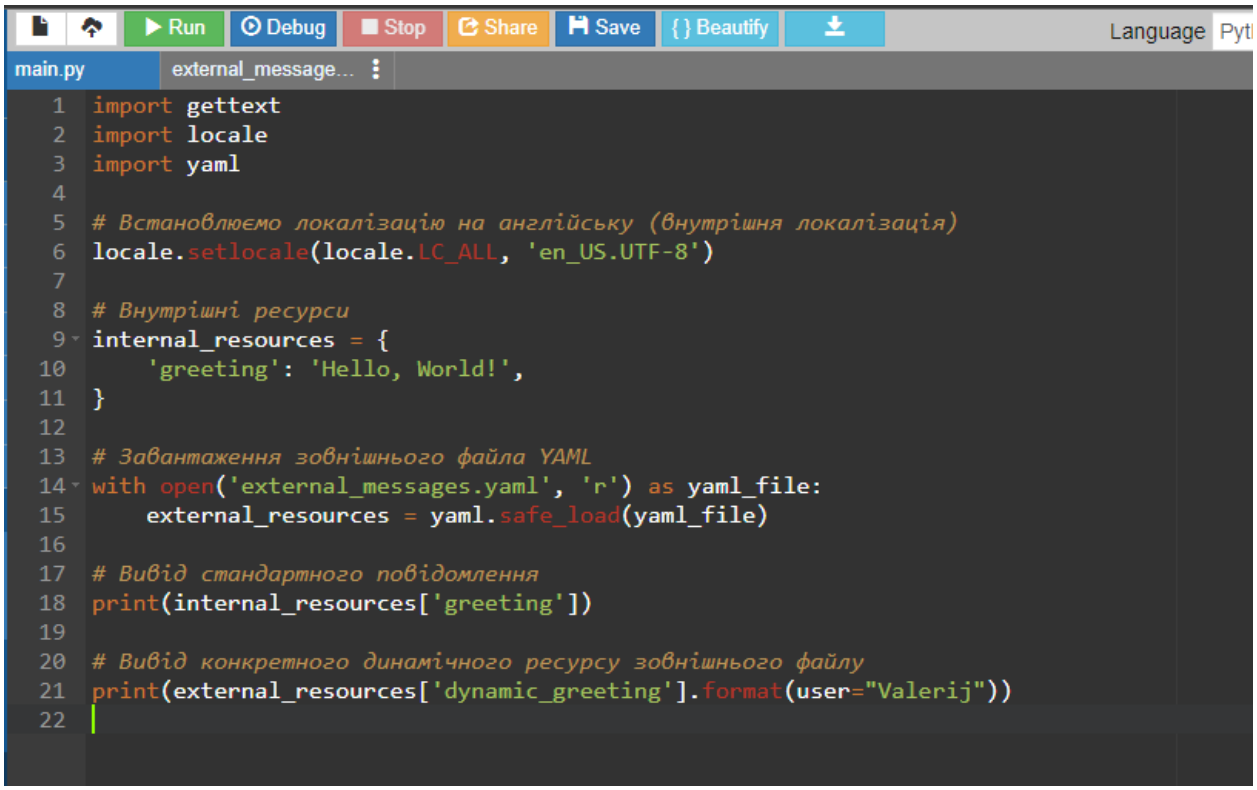

Рисунок 1.7 - Локалізація програмних додатків внутрішнім та зовнішнім способами (мікс)

Результати роботи програми:

```
Hello, World!
Welcome, Valerij!
```

1.1.4. Локалізація програмних додатків за допомогою API сервісів

API сервіси дають змогу динамічно завантажувати переклади під час роботи додатка, тим самим спрощуючи управління локалізацією та також дозволяють отримувати більш якісний переклад. Це є одним з найефективнішим способом отримання локалізації.

Сервіси API зазвичай підтримують велику кількість мов, що в свою чергу забезпечує високу якість перекладу. Для того щоб отримати переклад треба надіслати API запит на сервер де й буде здійснено переклад та результат повернеться.

Перш за все треба обрати API сервіс з багатьох, який буде підходити до поставленої задачі. Далі необхідно зареєструватися на цьому сервісі та

отримати API ключ для доступу до сервісу та додати його до додатку. Цей ключ і буде використовувати додаток для користування API додатком.

Використання API сервісів для локалізації має багато переваг: зручність, можливість швидко отримувати актуальні переклади і спрощення процесу локалізації.

До мінусів слід враховувати обмеження, такі як обмежена кількість безкоштовних запитів на деяких платформах та можливість обмежень у використанні API, які можуть призвести до вартості.

1.1.5. Використання мовних плагінів

Цей спосіб автоматично та динамічно перекладає данні на обрану мову, за допомогою спеціальних бібліотек або розширень. Тож не треба створювати окремі версії додатку.

Переклад додатку відбувається безпосередньо в додатку. Тобто додатку не треба звертатися до зовнішніх сервісів з перекладу, тому що все перекладається локально в самому додатку через розширення та плагіни.

Перш за все треба обрати мовний плагін або бібліотеку з автоматичним перекладом тексту. Ось приклад найпопулярніших серед них: Google Translate API, Microsoft Translator API та інші. Далі підключаємо бібліотеку до коду на налаштовуємо її підключаючи до API плагіну. Далі в коді прописуємо, що коли треба локалізувати дані, то ми передаємо рядок до плагіну та отримуємо локалізований текст.

Сильною стороною мовного плагіну є автоматичний переклад, що є швидшим та зручнішим у роботі додатку.

Проте є і мінуси: переклад через мовний плагін є менш точним, ніж ручний переклад. Найбільш очевидною ця проблема постає у складних текстах. Проте якість залежить від конкретного мовного плагіна.

1.2. Існуючі формати файлів локалізацій

Існує кілька основних форматів ключ-значення для інтернаціоналізації (i18n) та локалізації (l10n) програмних додатків. Кожен з цих форматів має як свої сильні так і слабкі сторони та особливості. Тому використовувати їх треба в залежності від конкретних потреб продукту та вимог.

1.2.1. Extensible Markup Language

Синтаксис: дані в XML подані в вигляді тегів та структур дерева. Відокремлення тегу відбувається шляхом написання кутових дужків напочатку та у кінці тегу <tag>, також теги можуть містити атрибути.[1]

Читабельність: читання XML-файлів є незручною справою, оскільки цей файл містить велику кількість тегів і закриваючих елементів. Серед трьох представлених форматів файлів (XML, YAML, JSON) – XML найоб'ємніший, тому і читабельність у нього гірша за інших (див. рис. 1.8.).

```
<?xml version="1.0" encoding="utf-8"?>
    <resources>
            <!-- Please don't change the name of the application -->
            <string name="app.name">"TODO List"</string>
            <string name="app.login.text">"Enter your credentials below to login"</string>
            <string name="app.login.message.success">"Login successful !
            Welcome back %s !"</string>
            <string name="app.login.message.error">"Make sure you've entered the correct username and password"</string>
            <string name="app.login.password.forget">"I forgot my password"</string>
            <string name="app.login.password.reset">"Enter your address in the field below. A new password will be sent to your inbox."</string>
            <string name="app.register.text">"Sign up for free! No credit card required!"</string>
            <string name="app.register.message.success">"Thank you for signing up !
            Please check your email address to activate your account."</string>
            <string name="app.register.message.error">"We were unable to sign you up.
            Please correct the marked fields."</string>
            <string name="app.menu.terms">"Terms and conditions"</string>

            <!-- form_label -->
            <string name="app.login.text">"Username"</string>
            <plurals name="app.users.active">
                    <item quantity="one">"Just one user online"</item>
                    <item quantity="other">"There are %d users online"</item>
            </plurals>

            <!-- form_label -->
            <string name="app.login.pass">"Password"</string>
    </resources>
```

Рисунок 1.8 - приклад XML файлу

Призначення: файли XML частіше за все використовується для обміну структурованими даними між системами або для зберігання конфігураційних файлів.

1.2.2. YAML Ain't Markup Language

Синтаксис: У своєму синтаксисі YAML містить численні пробіли та відступи, також використовує двокрапку. Це надає структурі даних файлу легкого читання і редагування. [2]

Читабельність: YAML файл є найчитабельнішим з усіх трьох варіантів через те, що не використовує теги (див. рис. 1.9). Через свою гарну читабельність ідеально підходить для конфігураційних файлів і конфігурацій в програмах.

```yaml
en:
  app:
    name: TODO List
    login:
      text: Enter your credentials below to login
      message:
        success: "Login successful!\nWelcome back %s!"
        error: Make sure you've entered the correct username and password
      password:
        forget: I forgot my password
        reset: Enter your address in the field below. A new password will be sent to your inbox.
      user: Username
      pass: Password
    register:
      text: Sign up for free! No credit card required!
      message:
        success: "Thank you for signing up !\nPlease check your email address to activate your account."
        error: "We were unable to sign you up.\nPlease correct the marked fields."
    menu:
      terms: Terms and conditions
```

Рисунок 1.9 - приклад YAML файлу

Призначення: YAML краще всього використовувати для конфігураційних файлів або даних, для опису об’єктів. Також його можна використати для структурування, читання та легкого редагування.

1.2.3. JavaScript Object Notation

Синтаксис: JSON файл у своєму синтаксисі має пари ключ-значення. А саме: лапками для рядків та квадратними дужками для масивів. Синтаксис JSON файлу дуже схоже на JavaScript, оскільки він походить від нього. JSON файл використовується в багатьох мовах програмування для обміну даними.

Читабельність: синтаксис JSON файлу використовує багато лапок,

дужків, що ускладнює читання файлу (див. рис. 1.10). Але для машинного оброблення це надає перевагу. [3]


```
[
  {
    "term": "app.name",
    "definition": "TODO List",
    "context": "",
    "term_plural": "",
    "reference": "\/app\/modules\/views",
    "comment": "Please don't change the name of the application"
  },
  {
    "term": "app.login.text",
    "definition": "Enter your credentials below to login",
    "context": "",
    "term_plural": "",
    "reference": "\/app\/modules\/views",
    "comment": ""
  },
  {
    "term": "app.login.message.success",
    "definition": "Login successful !\nWelcome back %s !",
    "context": "",
    "term_plural": "",
    "reference": "\/app\/modules\/views",
    "comment": ""
  },
  {
    "term": "app.login.message.error",
    "definition": "Make sure you've entered the correct username and password",
    "context": "",
    "term_plural": "",
    "reference": "\/app\/modules\/views",
    "comment": ""
  },
```


Рисунок 1.10 - приклад JSON файлу

Призначення: синтаксис та читабельність JSON файлу робить його одним з найкращих для обміну даними між програмами та веб-сервісами.

Отже, кожен з цих форматів має свої недоліки та переваги. Наприклад YAML є найчитабельнішим, у порівнянні з іншими форматами файлів локалізації. Тож вибір між цими форматами повинен відштовхуватися від вашої цілі. Наприклад: якщо треба обмінюватися даними між веб-сервісами, то JSON буде кращим вибором. Для зберігання конфігураційних або структурованих файлів слід надати перевагу YAML. XML у свою чергу найкращий для великого обсягу структурованих даних.

1.3. Сервіси, що допомагають при перекладі файлів

Сьогодні існує безліч сервісів, що допомагають при перекладі файлів. Ці сервіси допомагають перекласти файли, тексти програмного забезпечення, документацію, веб-сторінки або інші тексти. Деякі з цих сервісів є безкоштовними, у інших – ціна залежить від обсягу використання. Також однією з головних характеристик є кількість мов, що підтримуються сервісом. Далі наведено приклади кількох з них:

Crowdin – це один з найбільших сервісів для локалізації та перекладу програмного забезпечення та веб-проектів, який підтримує XML, YAML, JSON і багато інших форматів файлів . Цей сервіс співпрацює з такими відомими компаніями як: Valve, Discord, Microsoft, GitHub та іншими. Цей сервіс не є безкоштовним (див. рис. 1.11). [4]

Веб-додаток стане у нагоді для таких випадків як: локалізація програмного забезпечення, локалізація інтерфейсу користувача, локалізація веб-сайту, локалізація ігор та локалізація мобільних додатків.

Сервіс був створений українським програмістом Сергієм Дмитришиним і наразі налічує близько 174 тисяч проєктів локалізації.

Transifex - веборієнтована платформа управління локалізацією. Платформа розроблена для технічних проектів з частим оновленням змісту: програмного забезпечення, документації та вебсайтів. Технічні засоби, що надаються платформою, дозволяють розробникам автоматизувати процес локалізації. До речі з вільних проектів плата за використання сервісу не стягується. Цей сервіс співпрацює з такими проектами, як: Trello, SoundCloud, Creative Commons (див. рис. 1.12) [5]

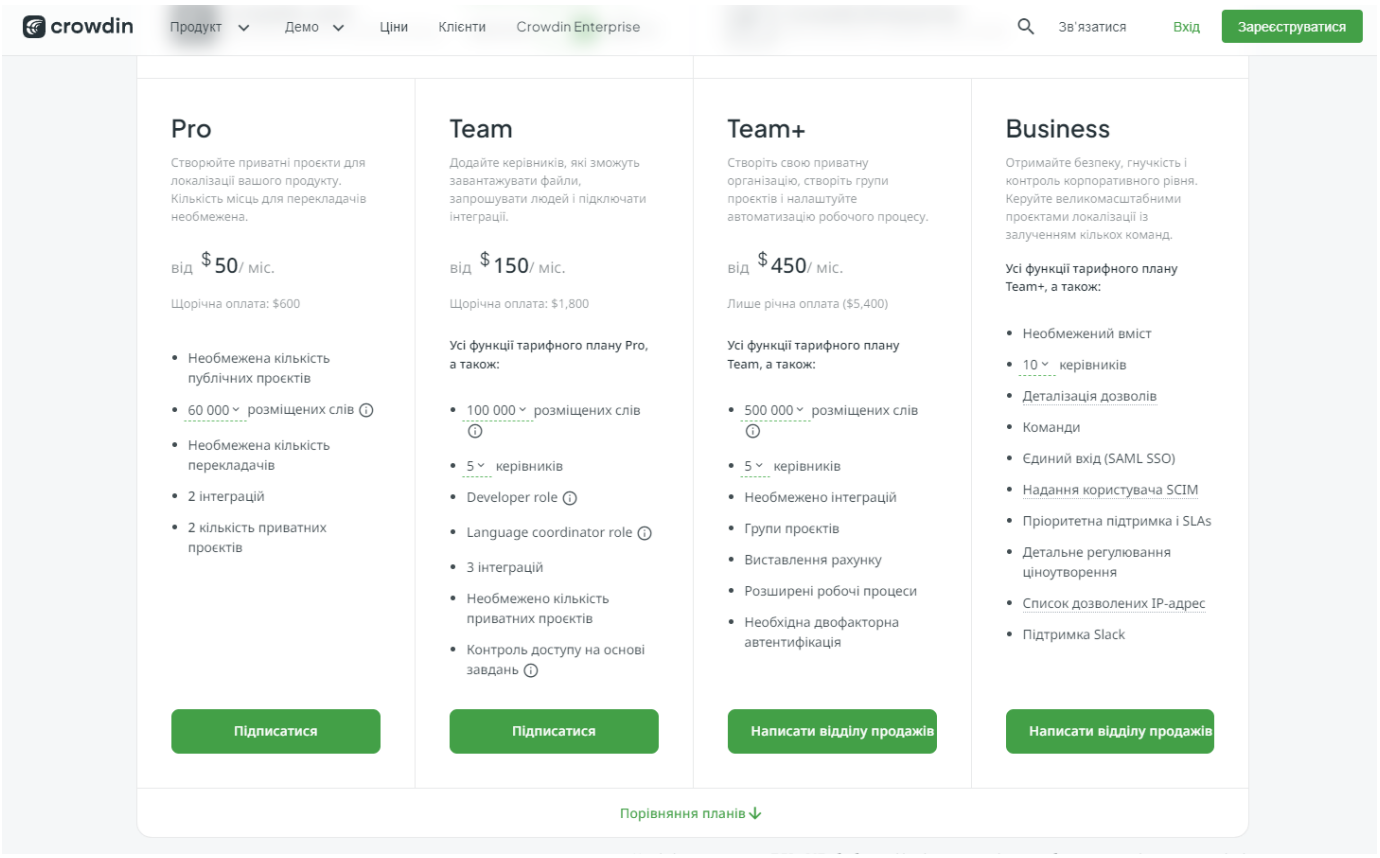

Рисунок 1.11 - ціни сервісу Crowdin

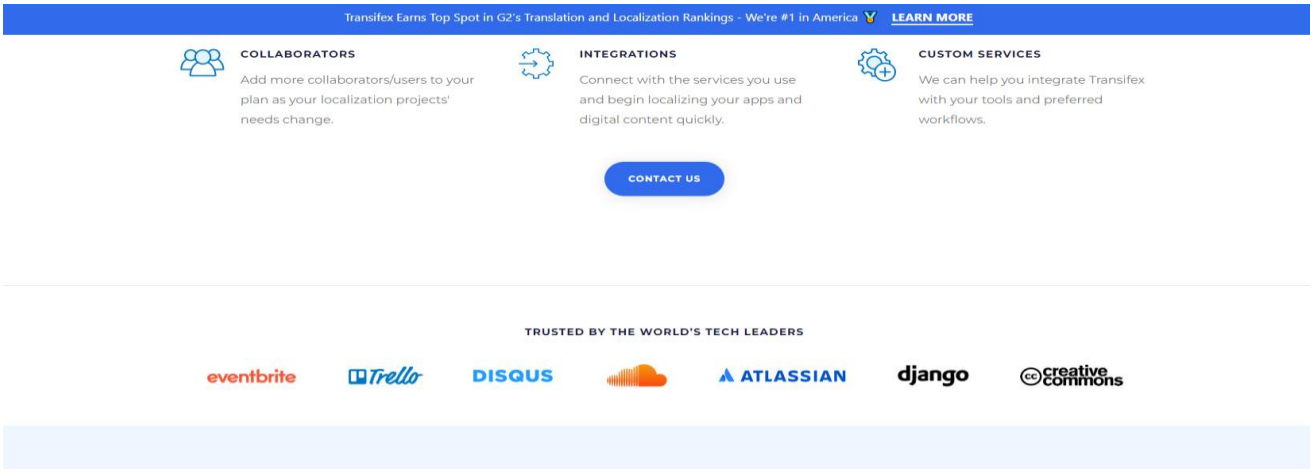

Рисунок 1.12 - співпраця з проектами Transifex

POEditor - це сервіс для локалізації та перекладу текстових ресурсів. Цей сервіс підтримує велику кількість форматів локалізації (див. рис. 1.13)

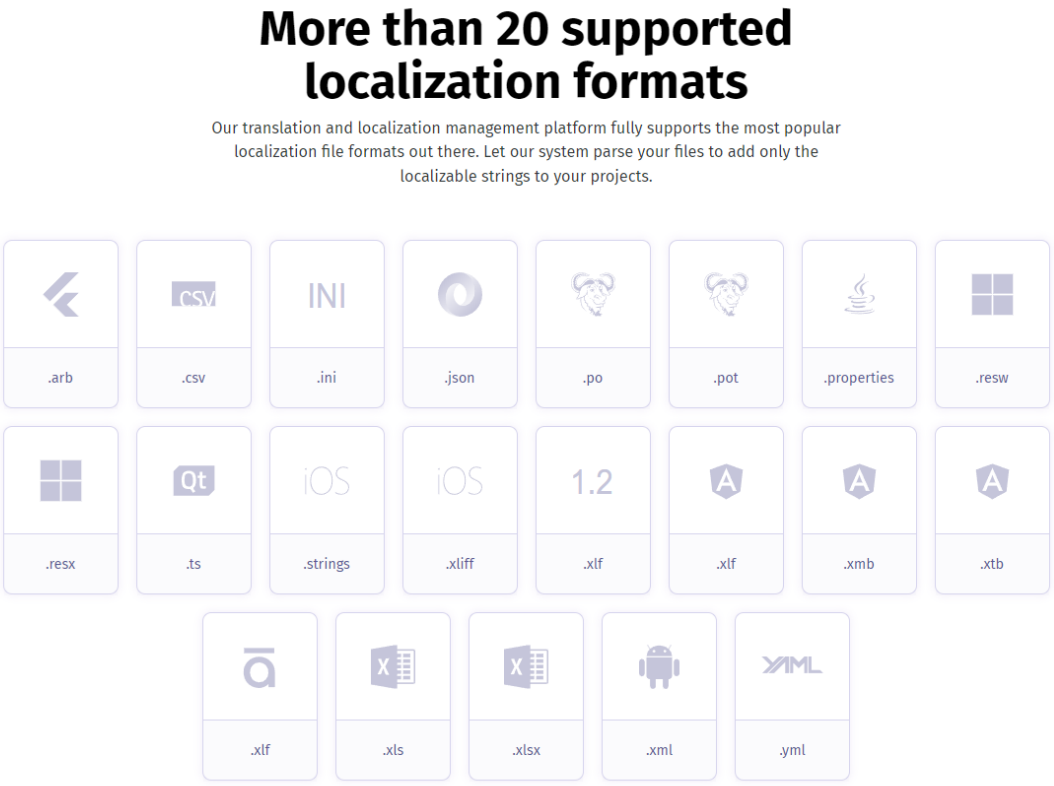

Рисунок 1.13 - формати локалізації

Також цей сервіс є безкоштовним для малих проєктів (див. рис. 1.14)

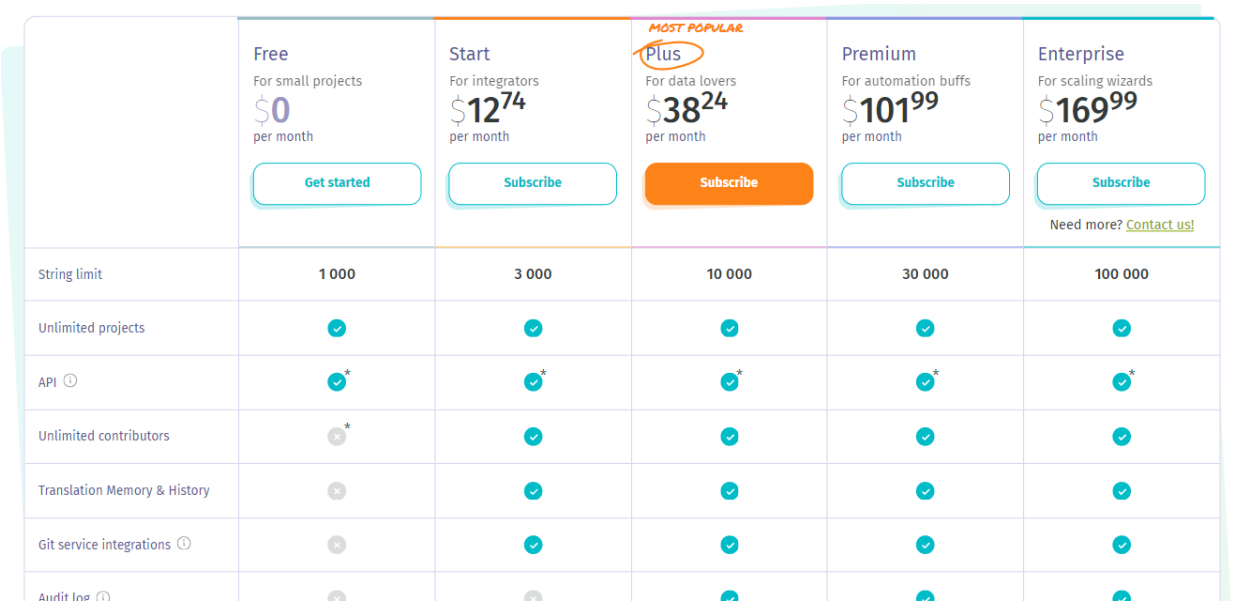

Рисунок 1.14 - цінова політика сервісу

Також сервіс POEditor інтегрований з GitHub і є автоматична синхронізація з репозиторіями. Не потрібно навіть торкатися мовного файлу. А зв'язки зі Slack та HipChat дозволять легко отримувати оновлення в реальному часі про процес локалізації. [6]

Lokalise — це хмарна система керування локалізацією та перекладом. Компанія була заснована в Ризі, Латвія, і є розподіленою компанією; всі співробітники працюють віддалено. Цей сервіс підтримує XML, YAML, JSON і багато інших форматів файлів. З цим сервісом співпрацюють mastercard, Arduino, Hyundai, Starbucks (див. рис. 1.15)

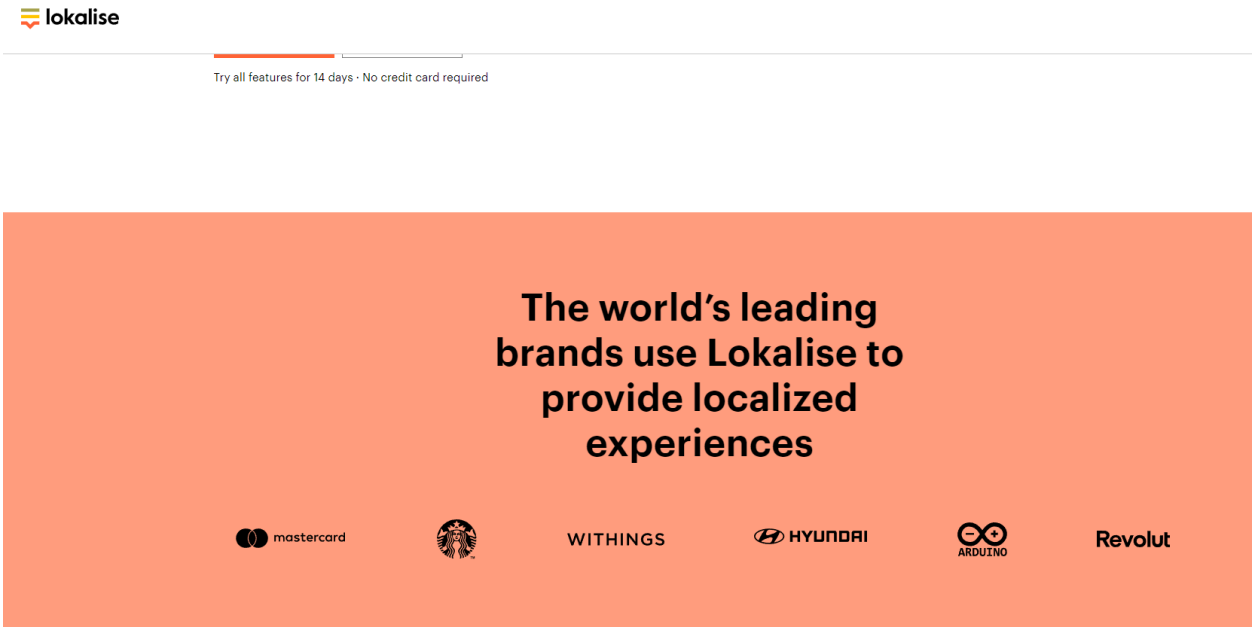

Рисунок 1.15 - компанії, що співпрацюють з сервісом

Згідно рекламного слогану Lokalise це кращий спосіб для компаній, які прагнуть до зростання, розширити свої мобільні програми, ігри, програмне забезпечення чи цифровий вміст кількома мовами. [7]

Weblate – сервіс для локалізації проектів з відкритим вихідним кодом. Має безкоштовний план для невеликих проектів [8]. Сервіс має інтеграції з GitHub, GitLab, Bitbucket, Jira, Slack. Підтримує багато форматів файлу(див. рис. 1.16)

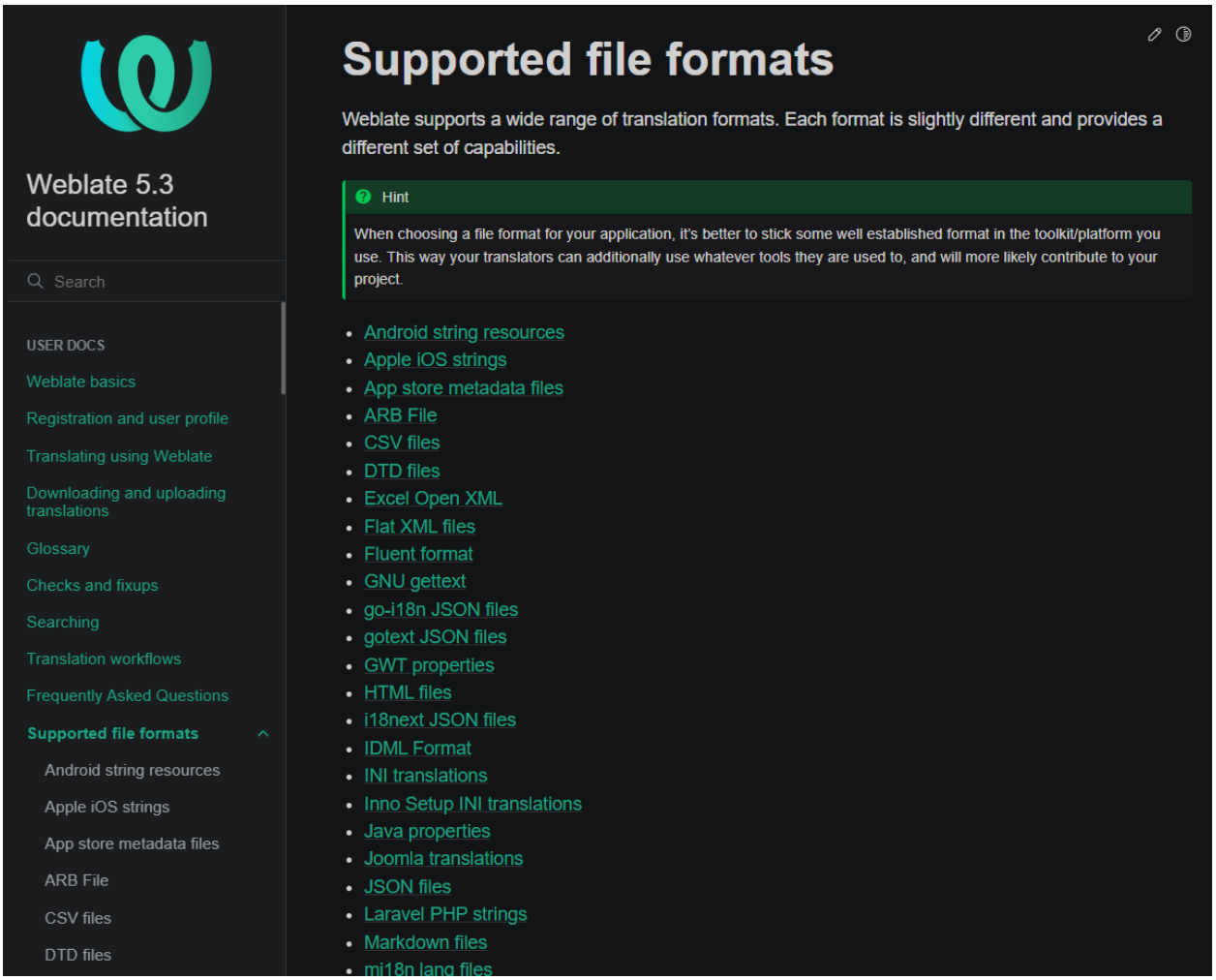

Рисунок 1.16 - формат файлу локалізації

Було зібрано ключові дані кожного сервісу та занесено до таблиці (див. табл. 1.1)

Таблиця 1.1 Таблиця порівнянь сервісів

Сервіс	Мови, які підтримуються	Формати файлів	Онлайн редагування	API	Ціна
Crowdin	300+	Gettext, JSON, XML, CSV, YAML	+	+	50 – 450 $
Transifex	170+	Gettext, JSON, XML, CSV, YAML	+	+	Безкоштовний план, але з обмеженням

Сервіс	Мови, які підтримуються	Формати файлів	Онлайн редагування	API	Ціна
POEditor	120+	Gettext, JSON, XML, CSV, YAML	+	+	Безкоштовний план, але для більшого функціоналу треба платити
Lokalise	80+	Gettext, JSON, XML, CSV, YAML	+	+	Безкоштовний план, але з обмеженням
Weblate	200+	Gettext, JSON, XML, CSV, YAML	+	+	Безкоштовний план, але з обмеженням

Аналізуючи данні представлені у таблиці ми можемо виділити ключові особливості, які повинен мати сервіс перекладу. Врахуємо це при постановці задачі. Між тим, зважаючи на цінові політики варто реалізувати власну, гібридну, модель перекладу для зменшення собівартості локалізації.

1.4. Постановка задачі

Розроблений сервіс супроводу локалізації додатків буде мати наступний функціонал:

1. Реєстрація, авторизація, аутентифікація користувачів.
2. Створення проекту перекладу.
3. Ініціалізація та оновлення проекту даними із фалів у форматі JSON, XML, YAML.
4. Автоматизований переклад проекту із застосуванням гібридної моделі та можливістю ручного корегування.
5. Збереження перекладу у обраному форматі.

2. ВИБІР МЕТОДУ ВИРІШЕННЯ ЗАДАЧІ

2.1. Інформаційна модель додатку

Розробка будь-якого проекту не може обійтися без вибору архітектури програмного забезпечення. Хоча, на перший погляд, можна подумати, що вибір архітектури це просто технічне рішення, але насправді різні інформаційні моделі додатку мають різні рівні продуктивності та рівні складності управління додатком, тому розробник повинен враховувати всі ці ризики при впровадженні того чи іншого варіанта. [9]

Розглянемо три основні типи архітектури:

1) Монолітна архітектура

Монолітна архітектура (Monolithic Architecture) – це один з основних підходів до побудови програмних продуктів. Цей підхід працює таким чином, що весь функціонал, разом з інтерфейсом для користувача, інтегрований в одну програму. [10]

Завдяки централізованій структурі у монолітній архітектурі, процеси розробки, оновлення, запуску, моніторингу та налагодження продукту не потребує дуже великих зусиль.

Проте, звісно ж є й мінуси у цій архітектурі. Через те що весь код додатку знаходиться в одному місці для внесення якихось правок може знадобитися зміна всього коду. Також при збільшені обсягу додатку дуже ймовірні виникнення проблем з масштабованістю та гнучкістю проекту.

2) Мікросервісна архітектура

Мікросервісна архітектура (Microservice Architecture) – підхід, у якому весь додаток розділяють на декілька малих автономних сервісів і кожен з них виконує доручений йому функціонал. Уся логіка взаємодії цих сервісів прописується у API (програмний інтерфейс застосунку). [11]

Перевагами мікросервісної архітектури є те, що розробники можуть працювати паралельно над різними функціями додатку, що надає

змогу швидше закінчити розробку проекту. Також, у разі збою в одному з функцій додатка інший сервіс продовжить свою роботу.

Щодо мінусів, через децентралізовану структуру проекту можуть виникнути проблеми з моніторингом стану та продуктивності, а також з управлінням комунікаціями та мережею.

Нижче можна подивитися таблицю, що підсумовує всю інформацію: (див. табл. 2.1.)

Таблиця 2.1 монолітна та мікросервісна архітектура

	Монолітна архітектура	Мікросервісна архітектура
Опис	Розробляється у вигляді монолітного модуля, у якому всі компоненти об’єднані в один додаток	Розподілення додатку на малі, автономні сервіси, які взаємодіють через мережу
Переваги	Дуже зручно і просто розробляти та розгортати	Гнучке масштабування та розвиток окремих компонентів
Недоліки	Незручно підтримувати додаток при збільшені його розміру	Незручно керувати та розгортати весь додаток

Трирівнева архітектура

Трирівнева архітектура (Multitier architecture) – ще одна поширена архітектура взаємодії в інтернеті. Цей підхід має три компоненти програми: клієнтський застосунок, сервер застосунку та підключення серверу до бази даних. [12]

До переваг можна віднести масштабованість, високий рівень надійності та здатність швидко переконфігурувати систему.

Недоліками є висока складність розробки застосунку, проблеми з моніторингом стану та продуктивності.

Для свого проекту я обрав трирівневу архітектуру і реалізував її в монолітному додатку. Цей підхід є одним з найбільш поширених та ефективних для розробки проектів. Ключовими аспектами у виборі для мене були: модульність (тобто я поділив додаток на три логічні рівні для більш

зручного тестування та роботи з проектом) та гнучкість (легше додавати будь-який новий функціонал). (див. рис. 2.1.)

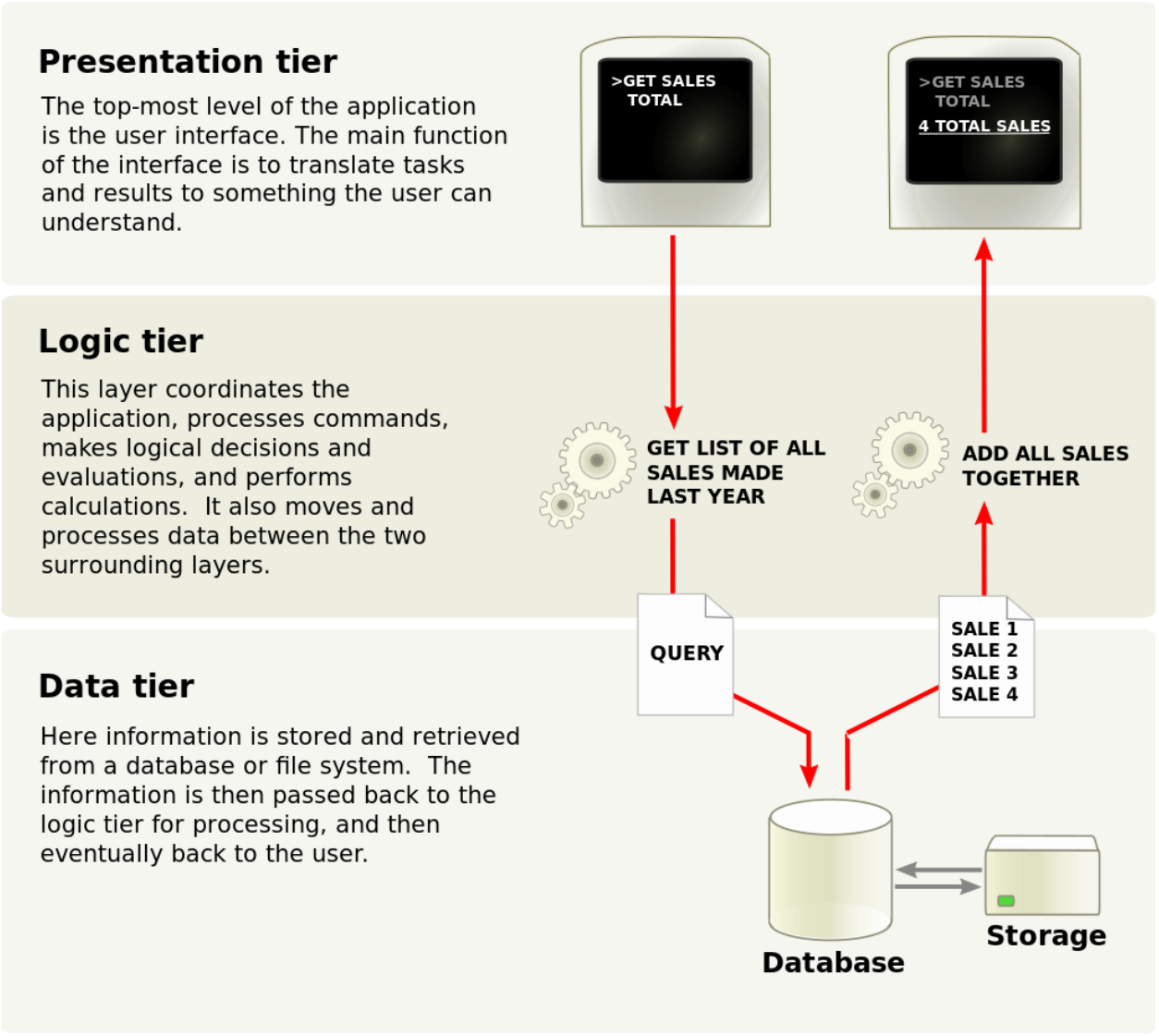

Рисунок 2.1 - приклад трирівневої архітектури [13]

2.2. Вибір платформи для розробки серверного шару

При розробці серверного шару я зіштовхнувся з проблемою вибору мови програмування, веб-фреймворку та бібліотек з інструментами. Треба було обрати зручні та надійні платформи для розробки проекту. Нижче я написав що саме обрав та причини цього вибору:

Мова програмування. Мовою програмування для створення серверного шару обрано Java. Java — це мова програмування та обчислювальна платформа, вперше випущена компанією Sun Microsystems у 1995 році. Ця мова програмування займає значну частину сучасного цифрового світу. На цій платформі побудовано побудовано багато проектів. [14]

Для того щоб почати розробку на мові Java спочатку необхідно завантажити **JDK.** JDK - це пакет для розробки програм для платформи Java [15]. До нього входять такі інструменти для розробки, як: компілятор коду, налагоджувач, архіватор та генератор документації. Для мого проекту я обрав **JDK 17.**

Веб-фреймворк. Для прощення розробки веб-сервісу обрано фреймворк Javalin 6. Javalin – це один з найпопулярніших веб-фреймворків для мови програмування Java або Kotlin [16]. До основних переваг Javalin можна віднести:

- Simple – Javalin дуже простий та зручний, має дуже мало концепцій, які потрібно вивчити.
- OpenAPI – Javalin підтримує OpenAPI (включаючи Swagger UI і ReDoc).
- Flexible – Javalin створений з урахуванням сумісності, додатки створюються однаково для Java або Kotlin, навідміну від інших веб-фреймворків.

Бібліотеки та інструменти. Для роботи з базою даних PostgreSQL використовується JDBC (Java Database Connectivity). [17] Java Database Connectivity (JDBC) — це API для взаємодії з базами даних із додатків, написаних мовою програмування Java. JDBC забезпечує універсальний спосіб доступу до різних систем управління базами даних (СУБД) і дозволяє виконувати SQL-запити, отримувати та оновлювати дані.

Також для функціоналу перекладу було використано DeeplAPI [18].

DeeplAPI – це один з найточніших у світі машинний перекладач. При використанні безкоштовного плану розробник може перекласти до 500 000 символів у місяць. Для використання цього API треба зареєструватися на сайті https://www.deepl.com/ та отримати ключ. [19]

2.3. Вибір платформи для розробки клієнтського шару

У ході розробки цього проекту для клієнтського шару було використано HTML, CSS та JavaScript. Вони підтримують багато платформ, доступні на різних пристроях та операційних системах, а також зручні у використанні для веб-розробника.

HTML (HyperText Mark-up Language) – це мова розмітки, яка використовується при розробці сайтів в інтернеті. HTML визначає з яких саме елементів та в якій черзі буде збиратися веб-сторінка, а також вказує де брати стилі елементів і скрипти. [20]

CSS (Cascading Style Sheets) – мова візуального представлення вмісту HTML-документа. Він включає в себе додавання кольору, редагування відступів, шрифтів, розташування кожного блоку на сторінці. Тобто, CSS файл містить правила оформлення всіх елементів на сторінці. Це робить код менш повторюваним та більш читабельним. [21]

JavaScript – це мова програмування з динамічною типізацією. JavaScript додає анімацію, спливаючі повідомлення, оновлює контент та валідує форми. Ця мова програмування робить сторінки функціональними. Перевагами JavaScript є простота вивчення та універсальність. Ця мова легко взаємодіє з HTML і CSS, а також прекрасно працює на будь-яких пристроях. [22]

2.4. Структура бази даних

У ході розробки сервісу локалізації додатків необхідно було використати бази даних для зберігання потрібної інформації. Я обрав PostgreSQL для збереження та управління даними. [23]

PostgreSQL – це популярна система управління базами даних, яка

використовує структуровану модель даних і SQL для взаємодії з ними. Ця СУБД є масштабованою та гнучкою, а також підтримує велику кількість типів даних. [24]

Процес взаємодії користувача з сервісом зображено на діаграмах (див. рис. 2.2 – 2.3)

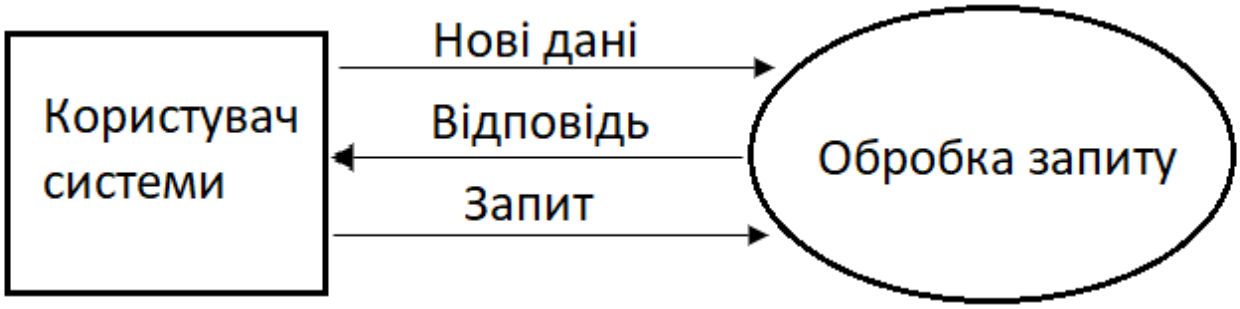

Рисунок 2.2 - DFD 0-го рівня для інформаціних систем

Для зберігання даних було створено 2 таблиці: USERS, USER_TRANSLATIONS.

USERS: Таблиця USERS зберігає дані про користувача і має такі поля:

- id – унікальний ідентифікатор, є первинним ключем
- email – поле, що зберігає електрону пошту, або логін
- password – пароль користувача

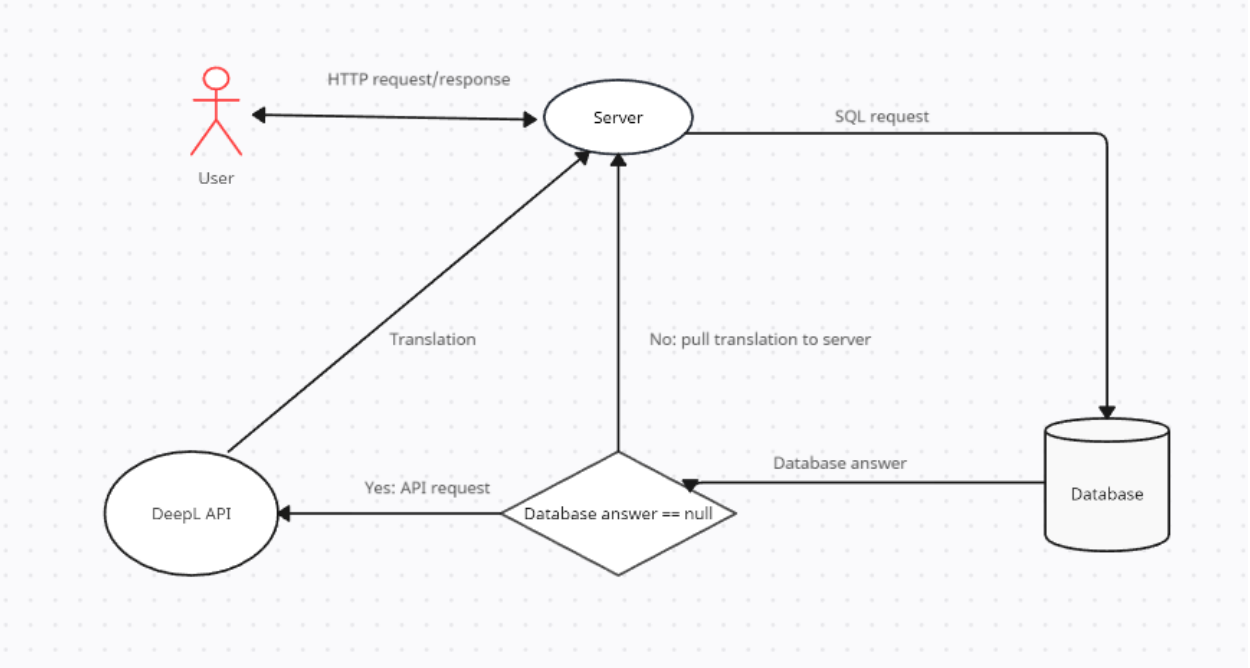

Рисунок 2.2 – UML діаграма послідовностей

USER_TRANSLATIONS: Ця таблиця зберігає дані користувацького перекладу ключів. Приймає зовнішній ключ користувача.

- id – ідентифікатор користувача
- project_name - назва проекту
- language – мова для локалізації
- key – ключ
- translation – переклад ключа

Зобразимо таблиці та їх відношення на ER-діаграмі (див. рисунок 2.3.)

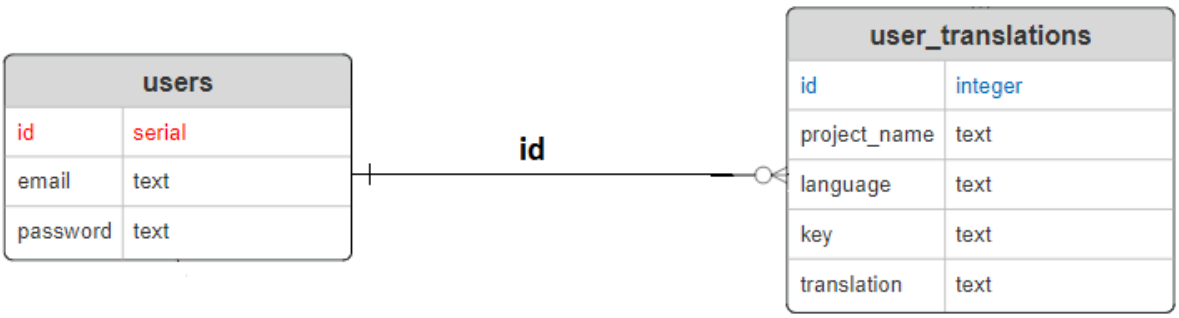

Рисунок 2.3 - ERD діаграма таблиць бази даних

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Структура бекенду додатку

Структура бекенду проєкта поділяється на декілька класів: (рис. 3.1.)

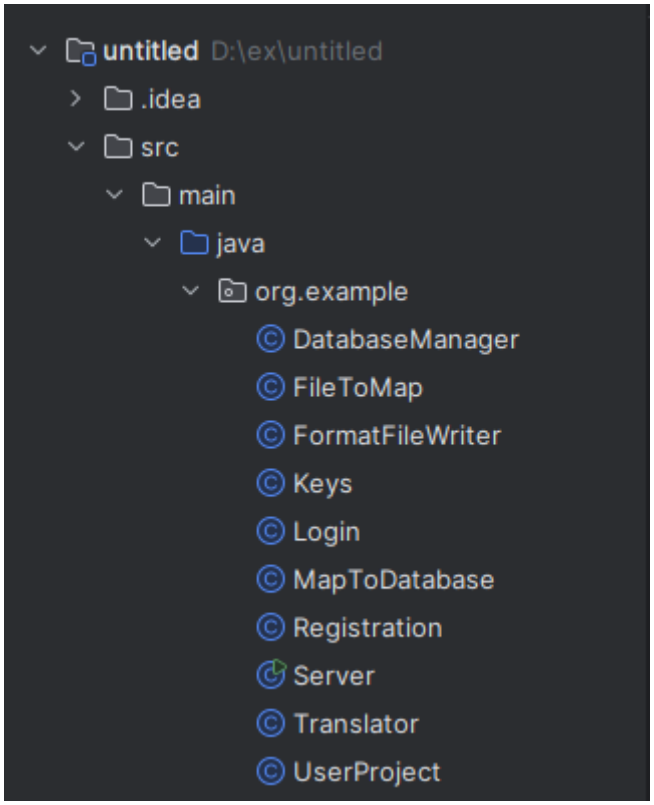

Рисунок 3.1 - структура бекенду додатку

- Клас «DatabaseManager»

Клас «DatabaseManager» створений для перекладу слів або виразів за допомогою таблиці community перекладів збережених у базі даних. Цей клас має два методи: 1) saveTranslation – метод що зберігає переклад до таблиці, 2) getTranslation – метод що виводить переклад з таблиці баз даних.[25]

- Клас «FileToMap»

Клас «FileToMap» створений для обробки файлів типу XML, YAML та JSON. Він читає файл, а потім перетворює вхідну інформацію у формат «ключ» - «значення», а далі за допомогою класу «MapToDatabase» зберігає у таблицю баз даних з користувацьким перекладом.

- Клас «FormatFileWriter»

Клас «FormatFileWriter» створений для того, щоб хеш-мапу з «ключ» - «значення», яка містить переклади, перетворити та записати у файл обраного типу XML, YAML або JSON.

- Клас «Keys»

Клас «Keys» реалізує підключення файлу «configuration.properties», який містить ключ для підключення до DeepL API та бази даних PostgreSQL.

- Клас «Login»

Клас «Login» створений для авторизації та входу користувача.

- Клас «MapToDatabase»

Клас «MapToDatabase» створений для перекладу слів або виразів за допомогою таблиці «користувацького перекладу». Клас має два методи, що зберігають переклад та знаходять та виводять його з таблиці.

- Клас «Registration»

Клас «Registration» створений для створення нового користувача.

- Клас «Server»

Клас «Server» містить основну логіку серверу та обробляє вхідні HTTP запити і повертає відповіді користувачам.

- Клас «Translator»

Клас «Translator» створений для перекладу слова або виразу за допомогою машинного перекладу DeepL API.

- Клас «UserProject»

Клас «UserProject» дістає з баз даних усю інформацію по проектам та мовам користувача

3.2. Структура фронтенду додатка

Фронтенд додатку включає в себе HTML розмітку для сторінок веб-інтерфейсу додатка, CSS відповідає за стилі кнопок, полів та таблиць(рис. 3.2.)

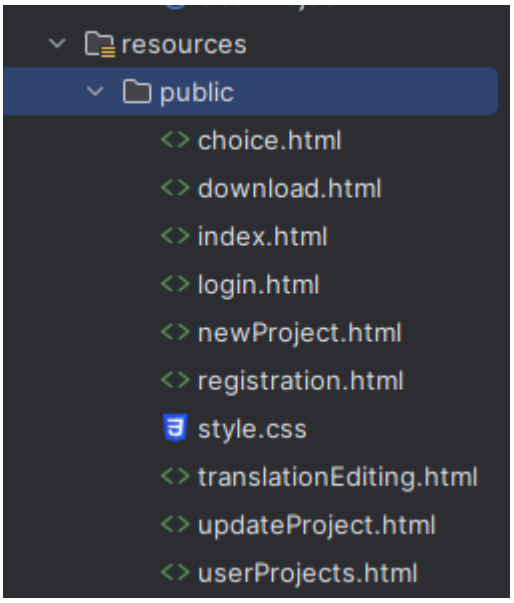

Рисунок 3.2 - структура фронтенду додатка

- «index.html»

Розмітка першої сторінки сервісу. На ній у користувача є можливість перейти на сторінку реєстрації, або на сторінку входу.

- «login.html»

Розмітка сторінки входу. Тут створений інтерфейс для входу користувача в додаток.

- «registration.html»

Розмітка сторінки для реєстрації користувача в додатку.(рис. 3.3)

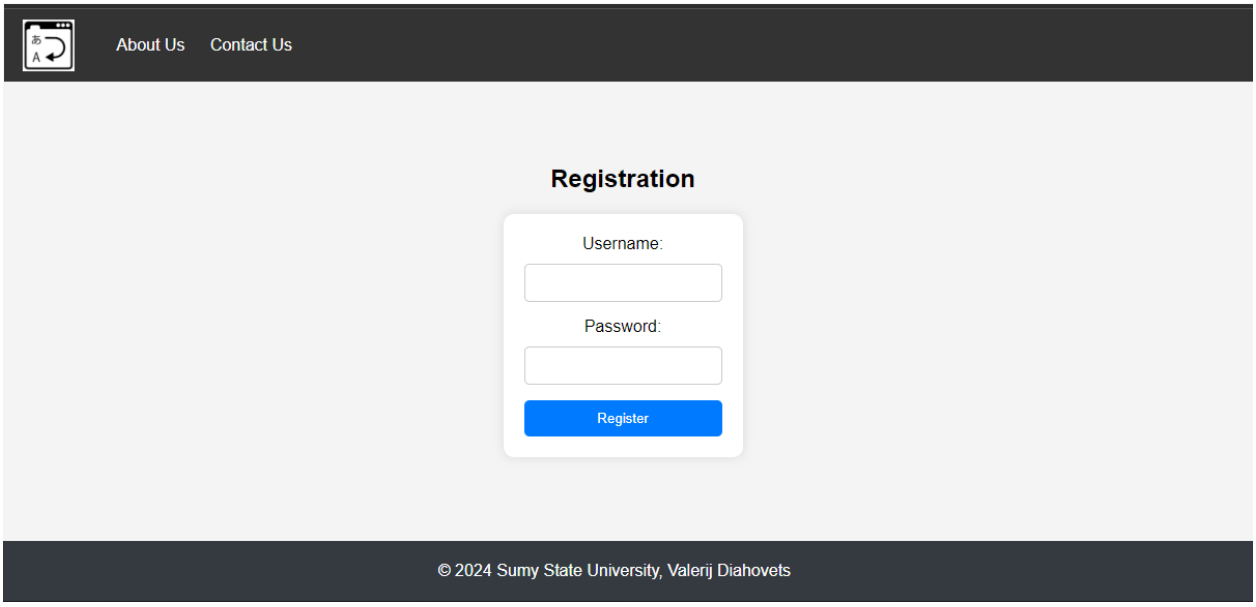

Рисунок 3.3 - сторінка реєстрації

- «userProjects.html»

Розмітка сторінки, що дозволяє користувачу переглянути всі його проекти й додати новий. (рис. 3.4)

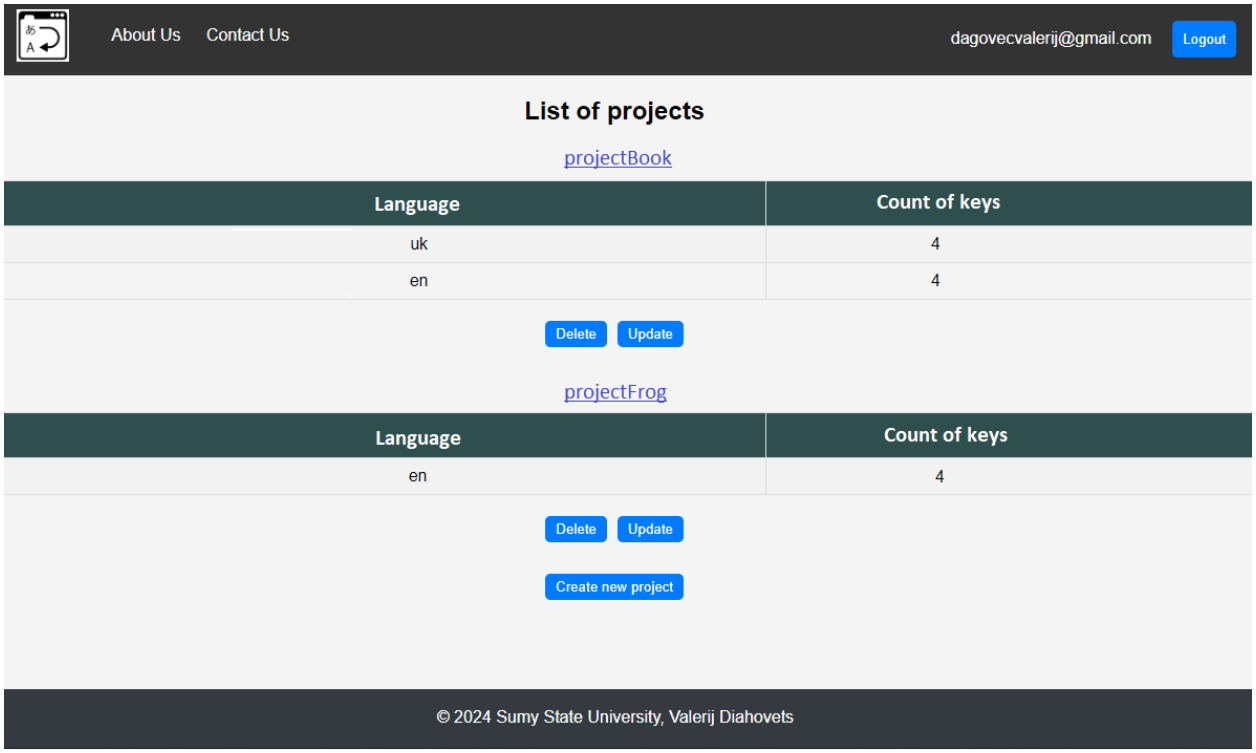

Рисунок 3.4 - сторінка вибору проекту

- «newProject.html»

Розмітка сторінки створення нового проекту користувача. Тут користувач має змогу написати назву нового проекту, завантажити файл та обрати його мову.

- «updateProject.html»

Розмітка сторінки оновлення вже існуючого проекту. Тут користувач може завантажити оновлений файл для вже існуючого проекту.

- «choise.html»

Розмітка сторінки на якій користувач маж змогу написати назву файлу, у який буде записано локалізацію, обрати мову для переводу та формат файлу. (рис. 3.5)

- «translationEditing.html»

Розмітка сторінки редагування перекладу користувачем. Тут користувач може редагувати переклад кожного ключа файлу.

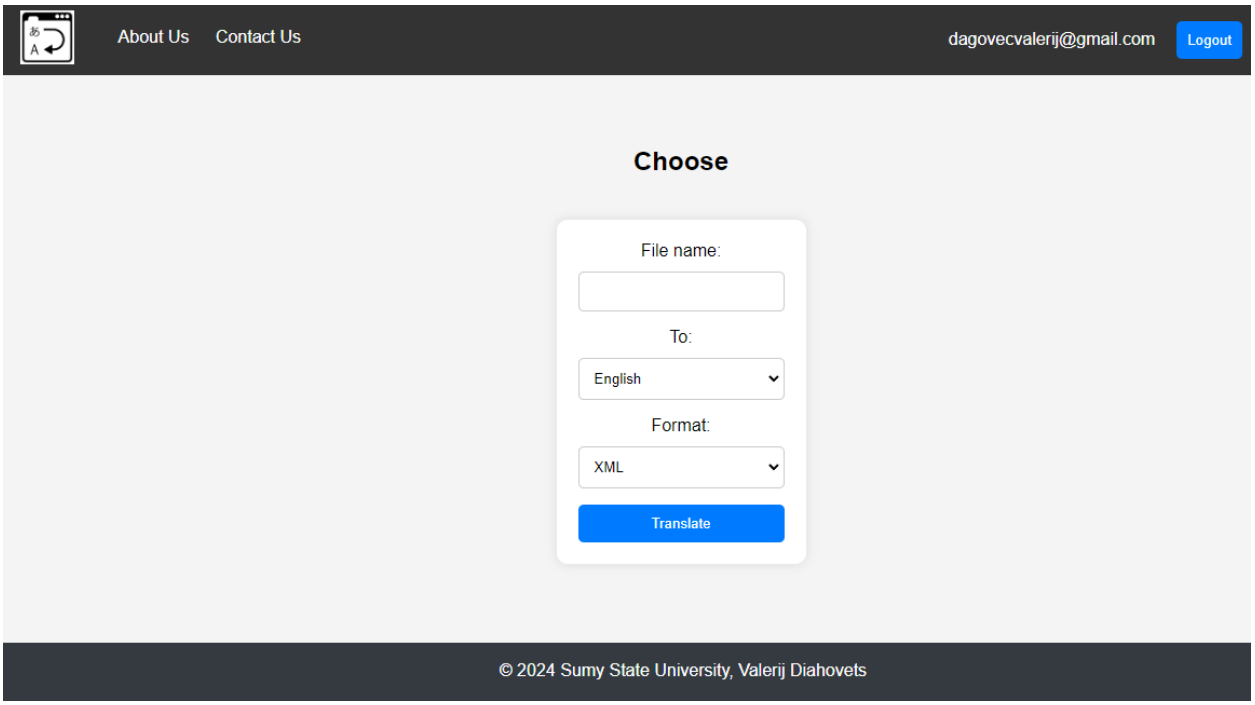

Рисунок 3.5 - сторінка вибору назви та формату файлу та мови перекладу

- «download.html»

Розмітка сторінки для завантаження файлу з перекладом

- «style.css»

Стилі для сторінок веб-інтерфейсу додатку

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було реалізовано сервіс, що дозволяє створювати і підтримувати переклади мовних файлів у форматах YAML, JSON, XML. Для оптимізації витрат було реалізовано гібридну модель перекладу, де першою ланкою отримуються результати отримані завдяки колаборативному підходу. За відсутності результатів застосовується машинний переклад із використанням DeepL API. Сервіс, також, дозволяє редагувати отримані результати та отримані результати та зберігати їх у різних форматах.

Для розробки додатку було виконано пошук та аналіз доступних інформацій моделей, СУБД, фреймворків та API для машинного перекладу.

За результатами роботи інтелектуальної системи, можна зробити висновок, що застосування розробленого сервісу зменшить витрати часу та ресурсів на локалізацію додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. XML для новачків - Підтримка від Microsoft. *Microsoft Support*. URL: https://support.microsoft.com/uk-ua/office/xml-для-новачків-a87d234d-4c2e-4409-9cbc-45e4eb857d44 (дата звернення: 09.05.2024).
2. YAML Syntax – Ansible Community Documentation. *Ansible Documentation*. URL: https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html (date of access: 09.05.2024).
3. JSON - JavaScript | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON (date of access: 09.05.2024).
4. Crowdin | Localization Management Platform for agile teams. *Crowdin*. URL: https://crowdin.com/ (date of access: 09.05.2024).
5. Pricing New. *Transifex*. URL: https://www.transifex.com/pricing/ (date of access: 09.05.2024).
6. Localization tools for the entire team - POEditor features. *POEditor*. URL: https://poeditor.com/features/ (date of access: 09.05.2024).
7. A Localization and Translation Software Tool | Lokalise. *lokalise.com*. URL: https://lokalise.com/ (date of access: 09.05.2024).
8. Weblate – мережева локалізація. *Weblate*. URL: https://weblate.org/uk/ (дата звернення: 09.05.2024).
9. Архітектура програмного забезпечення | Wezom. *IT-компания полного цикла разработки программных продуктов WEZOM - Киев, Украина*. URL: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya (дата звернення: 09.05.2024).
10. Монолітна архітектура ПЗ. - QALight. *QALight*. URL: https://qalight.ua/baza-znaniy/shho-take-monolitna-arhitektura/ (дата звернення: 09.05.2024).

11. GlobalLogic Ukraine. *GlobalLogic Ukraine*. URL: https://www.globallogic.com/ua/insights/blogs/microservices-architecture-for-beginners-part-one/ (date of access: 09.05.2024).
12. Java Professional *JavaRush*. URL: https://javarush.com/ua/quests/lectures/ua.questservlets.level14.lecture01 (date of access: 09.05.2024).
13. Weiterleitungshinweis. *Google*. URL: https://www.google.com/url?sa=i&url=https://uk.wikipedia.org/ (date of access: 10.05.2024).
14. Oracle | Cloud Applications and Cloud Platform. *Oracle | Cloud Applications and Cloud Platform*. URL: https://www.oracle.com/ (date of access: 09.05.2024).
15. Java. JVM Основи. JVM, JRE та JDK. Уроки для початківців. W3Schools українською. *W3Schools українською. Безплатні уроки онлайн для початківців, школярів та студентів*. URL: https://w3schoolsua.github.io/hyperskill/jvm-jre-jdk.html#google_vignette (дата звернення: 09.05.2024).
16. Javalin - A lightweight Java and Kotlin web framework. Create REST APIs in Java or Kotlin easily. *Javalin*. URL: https://javalin.io/ (date of access: 09.05.2024).
17. Informix Servers 12.10. *IBM in Deutschland, Österreich und der Schweiz*. URL: https://www.ibm.com/docs/en/informix-servers/12.10?topic=started-what-is-jdbc (date of access: 09.05.2024).
18. DeepL Translate API | Machine Translation Technology. *DeepL Translate: The world's most accurate translator*. URL: https://www.deepl.com/en/pro-api (date of access: 09.05.2024).
19. Java SE | Oracle Technology Network | Oracle. *Oracle | Cloud Applications and Cloud Platform*. URL: https://www.oracle.com/java/technologies/java-se-glance.html (date of access: 09.05.2024).

20. Що таке HTML? Для чого він необхідний та його перспективи. *MC.today, Media for Creators*. URL: https://mc.today/uk/shho-take-html-ta-yak-za-dopomogoyu-nogo-uvijti-do-it/ (дата звернення: 09.05.2024).
21. Що таке CSS. *Український веб-довідник*. URL: https://css.in.ua/article/shcho-take-html_10 (дата звернення: 09.05.2024).
22. Що таке JavaScript і для чого він потрібен – GoIT Global. *GoIT Global*. URL: https://goit.global/ua/articles/shcho-take-javascript-i-dlia-choho-vin-potriben/ (дата звернення: 09.05.2024).
23. PostgreSQL: about. *PostgreSQL: The world's most advanced open source database*. URL: https://www.postgresql.org/about/ (date of access: 09.05.2024)
24. PostgreSQL. *PostgreSQL*. URL: https://www.postgresql.org/ (date of access: 09.05.2024).
25. How To Connect to PostgreSQL with a JDBC driver • Vertex Academy. *Vertex Academy*. URL: https://vertex-academy.com/tutorials/en/connect-postgresql-jdbc-driver/ (date of access: 09.05.2024).

ДОДАТОК

Клас «DatabaseManager»

```
package org.example;

import org.postgresql.util.PSQLException;

import org.postgresql.util.PSQLState;

import java.io.IOException;

import java.sql.*;

import java.util.Objects;

public class DatabaseManager {

    private final String url = Keys.loadProperty("DB.URL");

    private final String username = Keys.loadProperty("DB.USERNAME");

    private final String password = Keys.loadProperty("DB.PASSWORD");

    private final Translator translator;

    public DatabaseManager(Translator translator) {

        this.translator = translator;

    }

        public void saveTranslation(String sourceLang, String sourceWord, String targetLang, String targetWord) {

            try (Connection connection = DriverManager.getConnection(url, username, password)) {

                String insertTranslation = "INSERT INTO translations (" + sourceLang + ", " + targetLang + ") VALUES (?, ?)";

                String insertTranslationWithEnglish = "INSERT INTO translations (" + sourceLang + ", " + targetLang + ", en) VALUES (?, ?, ?)";

                String updateTranslationWithTwoWords = "UPDATE translations SET " + sourceLang + " = ?, " + targetLang + " = ? WHERE en = ?";

                String insertQuery = insertTranslationWithEnglish;

                if(Objects.equals(sourceLang, "en") || Objects.equals(targetLang, "en")){

                    insertQuery = insertTranslation;
```

```
                }

                try (PreparedStatement insertStatement = connection.prepareStatement(insertQuery)) {

                    if(Objects.equals(sourceLang, "en") || Objects.equals(targetLang, "en")){

                        insertStatement.setString(1, sourceWord);

                        insertStatement.setString(2, targetWord);

                    } else {

                        insertStatement.setString(1, sourceWord);

                        insertStatement.setString(2, targetWord);

                        insertStatement.setString(3, translator.translate(targetWord, targetLang, "en"));

                    }

                    try {

                        insertStatement.executeUpdate();

                    } catch (PSQLException e) {

                        if(Objects.equals(e.getSQLState(), PSQLState.UNIQUE_VIOLATION.getState())){

                            String updateQuery = updateTranslationWithTwoWords;

                            try(PreparedStatement updateStatement = connection.prepareStatement(updateQuery)){

                                updateStatement.setString(1, sourceWord);

                                updateStatement.setString(2, targetWord);

                                updateStatement.setString(3, translator.translate(targetWord, targetLang, "en"));

                               updateStatement.executeUpdate();

                            }

                        } else {

                            System.err.println("Error when saving the translation to the database: " + e.getMessage());

                        }
```

```
                }
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        } catch (SQLException e) {
            System.err.println("Error when saving the translation to the database: " + e.getMessage());
        }
    }
    public String getTranslation(String targetLang, String sourceLang, String word) {
        try (Connection connection = DriverManager.getConnection(url, username, password)) {
            String query = "SELECT " + targetLang + " FROM translations WHERE " + sourceLang + " = ?";
            try (PreparedStatement statement = connection.prepareStatement(query)) {
                statement.setString(1, word);
                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        return resultSet.getString(targetLang);
                    }
                }
            }
        } catch (SQLException e) {
            System.err.println("Error retrieving translation from database: " + e.getMessage());
        }
        return null;
    }
}
```

Клас «FileToMap»

```
package org.example;
import org.json.JSONArray;
import org.json.JSONObject;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import org.yaml.snakeyaml.Yaml;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import java.io.*;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
public class FileToMap {
    private final MapToDatabase mapToDatabase;
    public FileToMap(MapToDatabase mapToDatabase) {
        this.mapToDatabase = mapToDatabase;
    }
    public void fileXML(File file, String language, int id, String projectName) {
        Map<String, String> map = new HashMap<>();
        try {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
```

```
                DocumentBuilder builder = factory.newDocumentBuilder();


                InputStream inputStream = new FileInputStream(file);
                Document document = builder.parse(new InputSource(inputStream));
                Element root = document.getDocumentElement();
                String key = root.getNodeName();
                saveElementXML(map, key, root);
                mapToDatabase.saveUserTranslations(id, projectName, language, map);
            } catch (Exception e) {
                System.err.println("Помилка при обробці XML файлу: " + e.getMessage());
            }
        }

        private void saveElementXML(Map<String, String> map, String key, Element element)
throws IOException {
            NodeList nodeList = element.getChildNodes();
            for (int i = 0; i < nodeList.getLength(); i++) {
                Node node = nodeList.item(i);
                //   key = node.getNodeName();
                if (node.getNodeType() == Node.ELEMENT_NODE) {
                    saveElementXML(map, key+"."+node.getNodeName(), (Element) node);
                } else if (node.getNodeType() == Node.TEXT_NODE) {
                    String text = node.getNodeValue().trim();
                    if (!text.isEmpty()) {
                        map.put(key, text);
                        key = "";
                    }
                }
            }
```

```
    }

    public void fileYAMLandJSON(File file, String language, int id, String projectName){
        Map<String, String> map = new HashMap<>();
        try {
            Yaml yaml = new Yaml();
            InputStream inputStream = new FileInputStream(file);
            Object yamlObject = yaml.load(inputStream);
            String key = "";
            saveElementYAMLandJSON(key, map, yamlObject);
            mapToDatabase.saveUserTranslations(id, projectName, language, map);
        } catch (Exception e) {
            System.err.println("Помилка при обробці YAML файлу: " + e.getMessage());
        }
    }

    private void saveElementYAMLandJSON(String key, Map<String, String> keyMap,
Object yamlObject) throws IOException {
        if (yamlObject instanceof Map) {
            Map<String, Object> map = (Map<String, Object>) yamlObject;
            for (Map.Entry<String, Object> entry : map.entrySet()) {
                String newKey = key + entry.getKey() + ".";
                Object value = entry.getValue();
                if (value instanceof String) {
                    String text = (String) value;
                    newKey = newKey.substring(0, newKey.length() - 1);
                    keyMap.put(newKey, text);
                    map.put(entry.getKey(), text);
                } else {
                    saveElementYAMLandJSON(newKey, keyMap, value);
```

```
                }
            }
        } else if (yamlObject instanceof List) {
            List<Object> list = (List<Object>) yamlObject;
            for (int i = 0; i < list.size(); i++) {
                Object item = list.get(i);
                String newKey = key + "[" + i + "]";
                if (item instanceof String) {
                    String text = (String) item;
                    keyMap.put(newKey, text);
                } else {
                    saveElementYAMLandJSON(newKey, keyMap, item);
                }
            }
        }
    }

}
```

Клас «FormatFileWriter»

```
package org.example;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.dataformat.xml.XmlMapper;
import org.yaml.snakeyaml.DumperOptions;
import org.yaml.snakeyaml.Yaml;
import org.yaml.snakeyaml.representer.Representer;
import java.io.File;
import java.io.FileWriter;
```

```
import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

public class FormatFileWriter {

    public void exportToXML(Map<String, String> localization, String filename) throws
IOException {

        XmlMapper xmlMapper = new XmlMapper();

        Map<String, Object> root = new HashMap<>();

        for (Map.Entry<String, String> entry : localization.entrySet()) {

            String[] keys = entry.getKey().split("\\.");

            addToNestedMap(root, keys, entry.getValue());

        }

        xmlMapper.writeValue(new File(filename), root);

    }

    public void exportToYAML(Map<String, String> localization, String filename) throws
IOException {

        Representer representer = new Representer();

        representer.getPropertyUtils().setSkipMissingProperties(true);

        DumperOptions options = new DumperOptions();

        options.setDefaultFlowStyle(DumperOptions.FlowStyle.BLOCK);

        Yaml yaml = new Yaml(representer, options);

        Map<String, Object> root = new HashMap<>();

        for (Map.Entry<String, String> entry : localization.entrySet()) {

            String[] keys = entry.getKey().split("\\.");

            addToNestedMap(root, keys, entry.getValue());

        }

        yaml.dump(root, new FileWriter(filename));

    }
```

```
    public void exportToJSON(Map<String, String> localization, String filename) throws
IOException {
        ObjectMapper objectMapper = new ObjectMapper();
        Map<String, Object> root = new HashMap<>();
        for (Map.Entry<String, String> entry : localization.entrySet()) {
            String[] keys = entry.getKey().split("\\.");
            addToNestedMap(root, keys, entry.getValue());
        }
        objectMapper.writeValue(new File(filename), root);
    }
    private void addToNestedMap(Map<String, Object> map, String[] keys, String value) {
        Map<String, Object> currentMap = map;
        for (int i = 0; i < keys.length - 1; i++) {
            currentMap = (Map<String, Object>) currentMap.computeIfAbsent(keys[i], k ->
new HashMap<>());
        }
        currentMap.put(keys[keys.length - 1], value);
    }
}
```

Клас «Keys»

```
package org.example;
import java.io.IOException;
import java.util.Properties;
public class Keys {
    public static String loadProperty(final String key) {
        Properties properties = new Properties();
        try {
```

```
properties.load(Keys.class.getClassLoader().getResourceAsStream("configuration.properties"));
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
            return properties.getProperty(key);
        }
}
```

Клас «Login»

```
package org.example;
import java.sql.*;
public class Login {
    private final String url = Keys.loadProperty("DB.URL");
    private final String username = Keys.loadProperty("DB.USERNAME");
    private final String password = Keys.loadProperty("DB.PASSWORD");
    public int loginUser(String user, String userPassword) {
        int id = 0;
        try (Connection connection = DriverManager.getConnection(url, username,
password)) {
            String selectUser = "SELECT id FROM users WHERE email = ? AND password =
?";
            try (PreparedStatement statement = connection.prepareStatement(selectUser)) {
                statement.setString(1, user);
                statement.setString(2, userPassword);
                try (ResultSet resultSet = statement.executeQuery()) {
                    if (resultSet.next()) {
                        id = resultSet.getInt("id");
                    }
```

```
                    return id;
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
    }
}
```

Клас «Registration»

```
package org.example;
import org.postgresql.util.PSQLException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class Registration {
    private final String url = Keys.loadProperty("DB.URL");
    private final String username = Keys.loadProperty("DB.USERNAME");
    private final String password = Keys.loadProperty("DB.PASSWORD");
    public int registrationUser(String user, String userPassword){
        int flag = 0;
        try (Connection connection = DriverManager.getConnection(url, username, password)) {
            String insertUser = "INSERT INTO users (email, password) VALUES (?, ?);";
            try(PreparedStatement insertStatement = connection.prepareStatement(insertUser)){
                insertStatement.setString(1, user);
```

```
                insertStatement.setString(2, userPassword);
                insertStatement.executeUpdate();
            } catch (PSQLException e){
                flag++;
                System.err.println("User has already exist " + e.getMessage());
            }
        } catch (SQLException e) {
            flag++;
            throw new RuntimeException(e);
        }
        return flag;
    }
}
```

Клас «Translator»

```
package org.example;
import okhttp3.*;
import org.json.JSONObject;
import java.io.IOException;
public class Translator {
    private final String apiKey = Keys.loadProperty("DEEPL.AUTH_KEY");
    private final OkHttpClient httpClient;
    public Translator() {
        this.httpClient = new OkHttpClient();
    }
    public String translate(String text, String sourceLanguage, String targetLanguage) throws IOException {
        // Створення запиту до API Deepl
```

```
            HttpUrl.Builder urlBuilder = HttpUrl.parse("https://api-free.deepl.com/v2/translate").newBuilder();
            urlBuilder.addQueryParameter("auth_key", apiKey);
            urlBuilder.addQueryParameter("text", text);
            urlBuilder.addQueryParameter("source_lang", sourceLanguage);
            urlBuilder.addQueryParameter("target_lang", targetLanguage);
            // Виконання HTTP запиту
            Request request = new Request.Builder()
                .url(urlBuilder.build())
                .build();
            try (Response response = httpClient.newCall(request).execute()) {
               if (!response.isSuccessful()) {
                  throw new IOException("Unexpected code " + response);
               }
               // Отримання результату перекладу з відповіді API
               JSONObject jsonResponse = new JSONObject(response.body().string());
               String answer = jsonResponse.getJSONArray("translations").getJSONObject(0).getString("text");
               if(answer.endsWith(".")){
                  answer = answer.substring(0, answer.length() - 1);
               }
               return answer;
            }
         }
}
```

Клас «UserProject»

```
package org.example;
import java.sql.*;
```

```
import java.util.ArrayList;
import java.util.List;
public class UserProject {
    private final String url = Keys.loadProperty("DB.URL");
    private final String username = Keys.loadProperty("DB.USERNAME");
    private final String password = Keys.loadProperty("DB.PASSWORD");

    public List<String> getProjects(int id) {
        List<String> projectNames = new ArrayList<>();
        try (Connection connection = DriverManager.getConnection(url, username, password)) {
            String select = "SELECT DISTINCT project_name FROM user_translations WHERE id = ?";
            try (PreparedStatement statement = connection.prepareStatement(select)) {
                statement.setInt(1, id);
                try (ResultSet resultSet = statement.executeQuery()) {
                    while (resultSet.next()) {
                        String projectName = resultSet.getString("project_name");
                        projectNames.add(projectName);
                    }
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return projectNames;
    }
```

```
    public List<String> getLanguage(int id, String projectName){
        List<String> languages = new ArrayList<>();
        try (Connection connection = DriverManager.getConnection(url, username, password)) {
            String select = "SELECT DISTINCT language FROM user_translations WHERE id = ? AND project_name = ?";
            try (PreparedStatement statement = connection.prepareStatement(select)) {
                statement.setInt(1, id);
                statement.setString(2, projectName);
                try (ResultSet resultSet = statement.executeQuery()) {
                    while (resultSet.next()) {
                        String language = resultSet.getString("language");
                        languages.add(language);
                    }
                }
            }
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return languages;
    }
}
```