МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_  _ проф. Приходько С.Б.

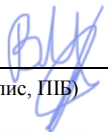
«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення

Виконав: студент групи 4157ст3



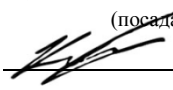
Микулянець В.А.

(підпис, ПІБ)

Керівник роботи:

Доцент, к.т.н., доцент

(посада, науковий ступень вчене звання)



Каіров В.О.

(підпис, ПІБ)

Миколаїв – 2026 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«_08_» ___04___ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Микулянець Василь Андрійович _____
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення _____

_____

Керівник роботи: доц. каф. ПЗАС, к.т.н., доц. Каіров В.О. _____

Затверджені наказом ректора №153 від 08.04.2026 р.

2. Термін подання роботи: 02.06.2026 р._____

3. Вихідні дані по роботі: _____

_____

4. Перелік питань, що належать до розробки (найменування розділів) _____

Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів 1. Титульний слайд; 2. Актуальність розробки. 3. Мета та завдання кваліфікаційної роботи; 4. Аналіз вимог до вебзастосунку; 5. Концептуальна модель та логічна модель вебзастосунку управління перекладами. 6. Архітектура вебзастосунку; 7. Статична модель вебзастосунку; 8. Діаграми діяльності; 9. Динамічна модель; 10. Обґрунтування вибору засобів розробки; 11. Фізична модель даних; 12. Результати розробки; 13. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 08.04.2026 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	31.03.2026	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	07.04.2026	ПП*
3.	Аналіз вимог до ПЗ	14.04.2026	ПП*
4.	Розробка архітектури ПЗ	21.04.2026	
5.	Розробка проекту ПЗ	05.05.2026	
6.	Реалізація та тестування ПЗ	12.05.2026	
7.	Підготовка розділу Результати розробки ПЗ	16.05.2026	
8.	Підготовка розділу з охорони праці	19.05.2026	ПП*
9.	Підготовка розділу Висновки	23.05.2026	
10.	Оформлення списку використаних джерел та додатків	26.05.2026	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	02.06.2026	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
12.	Підготовка презентації та доповіді	06.06.2026	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	09.06.2026	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	16.06.2026	

* - за результатами переддипломної практики (ПП), яка була з 03.02.2026 до 23.03.2026 р.

Студент _____ Микулянець В.А.
(підпис) (ПІБ)

Керівник роботи _____ Каіров В.О.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена вирішенню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення.

Виконано аналіз практичної задачі, огляд аналогів та постановку задачі на розробку. Здійснено аналіз вимог, обрано архітектуру MVC і розроблено проєкт вебзастосунку. Вебзастосунок реалізовано засобами фреймворку CakePHP 5 і СКБД MySQL та протестовано за програмою і методикою випробувань.

Кваліфікаційна робота викладена на 98 сторінках друкованого тексту та містить 25 рисунків, 8 таблиць, 5 додатків та список використаних джерел з 13 найменувань.

ABSTRACT

The qualification work addresses a practical software engineering problem characterized by complexity and uncertainty: the development of a web application for translation management for uninterrupted software localization.

The practical task was analyzed, existing analogues were reviewed and the problem statement was formulated. Requirements were analyzed, the MVC architecture was selected and the web application was designed. The application was implemented with the CakePHP 5 framework and the MySQL database management system, then tested and trialed.

The qualification work is presented on 98 pages and includes 25 figures, 8 tables, 5 appendices, and a list of references with 13 sources.

ЗМІСТ

ВСТУП

Дана кваліфікаційна робота присвячена розв’язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення. Для проєкту з кількома мовами кожна зміна тексту інтерфейсу породжує по одному оновленню ресурсного файлу на кожну мову, які перекладачі мають отримати, перекласти й повернути у репозиторій до релізу [1].

Наразі у командах розробки локалізація часто виконується шляхом ручного обміну файлами ресурсів між розробниками та перекладачами. Кожна зміна тексту інтерфейсу потребує оновлення файлів локалізації, ручної передачі їх перекладачам, отримання оновлених перекладів та повернення їх у кодову базу. Така схема не масштабується для проєктів із великою кількістю фраз і мов: фрази губляться при копіюванні, термінологія між перекладачами розходиться, файли ресурсів пошкоджуються при ручному злитті. Розробка централізованого вебзастосунку із підтримкою імпорту-експорту ресурсних файлів та REST API для експорту перекладів прибере ручне копіювання і дозволить автоматично синхронізувати переклади з кодовою базою.

Мета роботи: розробити вебзастосунок управління перекладами для безперебійної локалізації програмного забезпечення.

Завдання:

1) провести аналіз практичної задачі інженерії програмного забезпечення з розробки вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення;

2) провести аналіз існуючого програмного забезпечення для управління перекладами;

3) виконати постановку задачі на розробку вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення;

4) виконати аналіз вимог до вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення;

5) вибрати архітектуру вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення;

6) розробити проєкт вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення;

7) виконати реалізацію та провести тестування вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення.

Об’єктом кваліфікаційної роботи є процес розробки вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення.

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ВЕБЗАСТОСУНКУ УПРАВЛІННЯ ПЕРЕКЛАДАМИ ДЛЯ БЕЗПЕРЕБІЙНОЇ ЛОКАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Аналіз практичної задачі інженерії програмного забезпечення з розробки вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення

Розробка вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення є практичною задачею інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов. Комплексність зумовлена необхідністю одночасно підтримувати кілька форматів ресурсних файлів (PO, JSON, CSV) із коректною обробкою UTF-8 для багатомовного тексту, надавати REST API для синхронізації з кодовою базою та розмежовувати права доступу.

Наразі процес локалізації у більшості команд розробки виконується у ручному режимі і включає такі етапи:

1) витягання тексту інтерфейсу з кодової бази у файли ресурсів (PO, JSON, CSV);

2) надсилання цих файлів перекладачам електронною поштою або через систему обміну файлами;

3) очікування на оновлені переклади;

4) ручне об’єднання результатів та повернення їх у кодову базу.

На кожному з етапів можливі помилки: пропущені фрази, неактуальні переклади, пошкодження структури файлів ресурсів при ручному злитті, неузгодженість термінології між перекладачами. Чим більший проєкт і чим більше підтримуваних мов, тим частіше такі помилки трапляються.

Розробка вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення прибирає ручне копіювання файлів, зберігає переклади централізовано і дасть змогу автоматично синхронізувати їх з кодовою базою через REST API.

1.2 Аналіз існуючого програмного забезпечення для управління перекладами програмних продуктів

В даний час існує низка інструментів для управління перекладами програмних продуктів. Серед них найвідомішими є Crowdin, POEditor, Transifex та Lokalise.

Crowdin [2] є популярною комерційною хмарною платформою для управління перекладами, що підтримує велику кількість форматів ресурсних файлів (рисунок 1.1). Платформа надає інтерфейс для перекладачів, систему пам’яті перекладу та REST API.

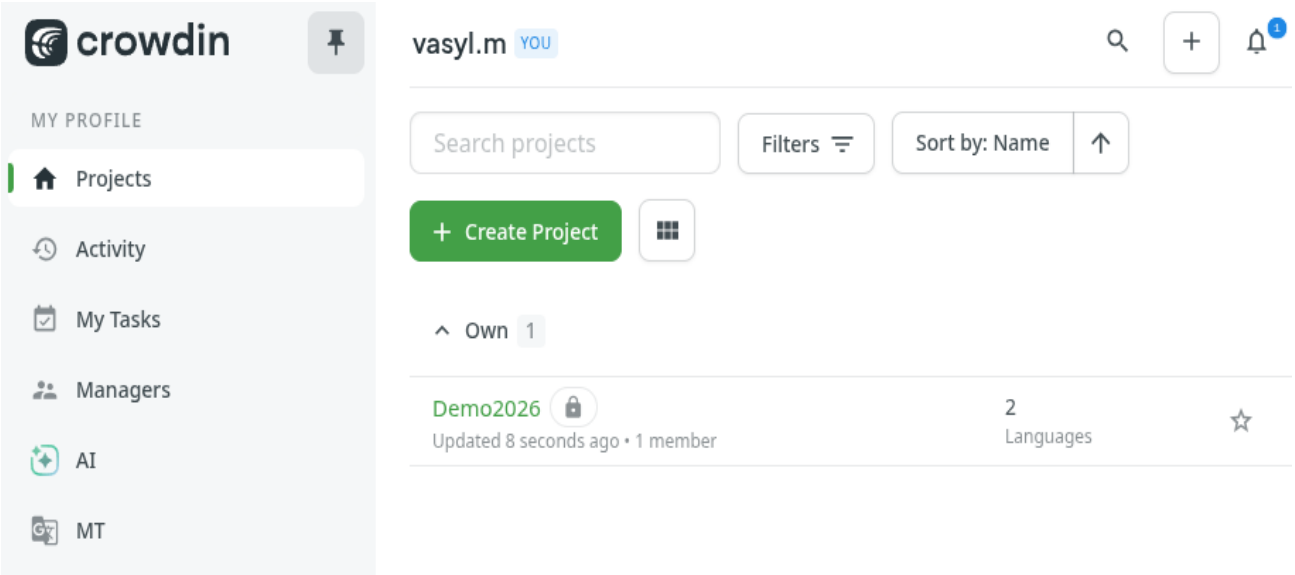


Рисунок 1.1 – Вебзастосунок для управління перекладами програмних продуктів Crowdin

POEditor [3] – це вебзастосунок для управління перекладами програмних продуктів з підтримкою основних форматів ресурсних файлів (PO, JSON, XLIFF) та REST API для автоматичної синхронізації перекладів

(рисунок 1.2). Платформа має безкоштовний план з обмеженням на кількість фраз.

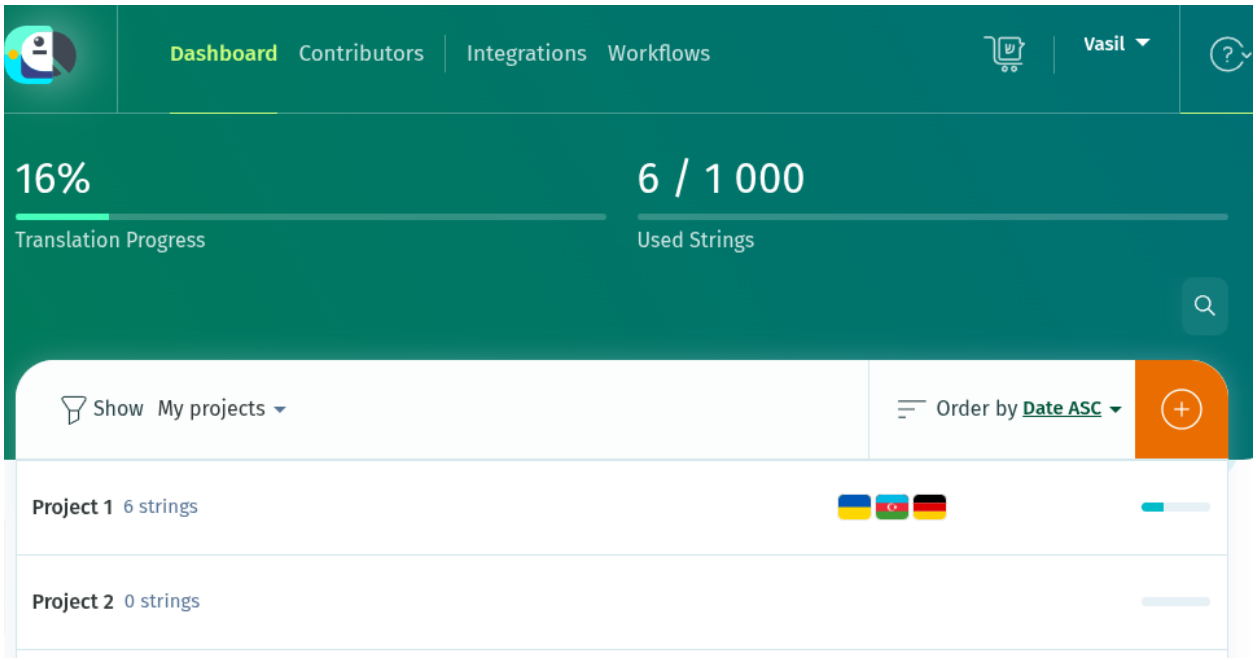


Рисунок 1.2 – Вебзастосунок для управління перекладами програмних продуктів POEditor

POEditor дозволяє створити окремі проєкти локалізації, після чого до них можна додавати мови (рисунок 1.3), фрази (рисунок 1.4) та переклади для доданих фраз (рисунок 1.5).

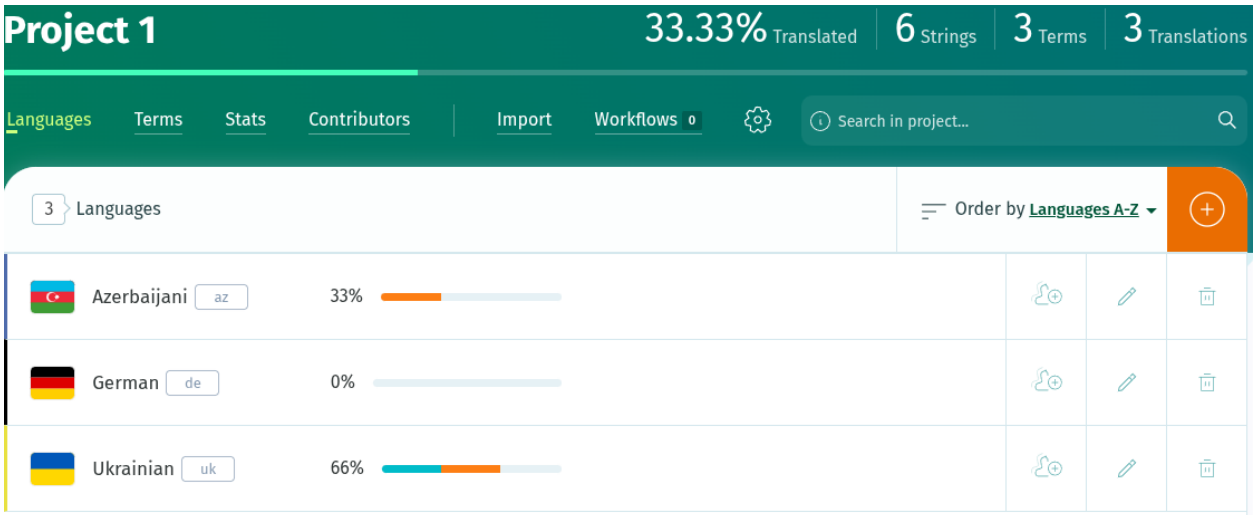


Рисунок 1.3 – Сторінка перегляду списку мов вебзастосунку POEditor

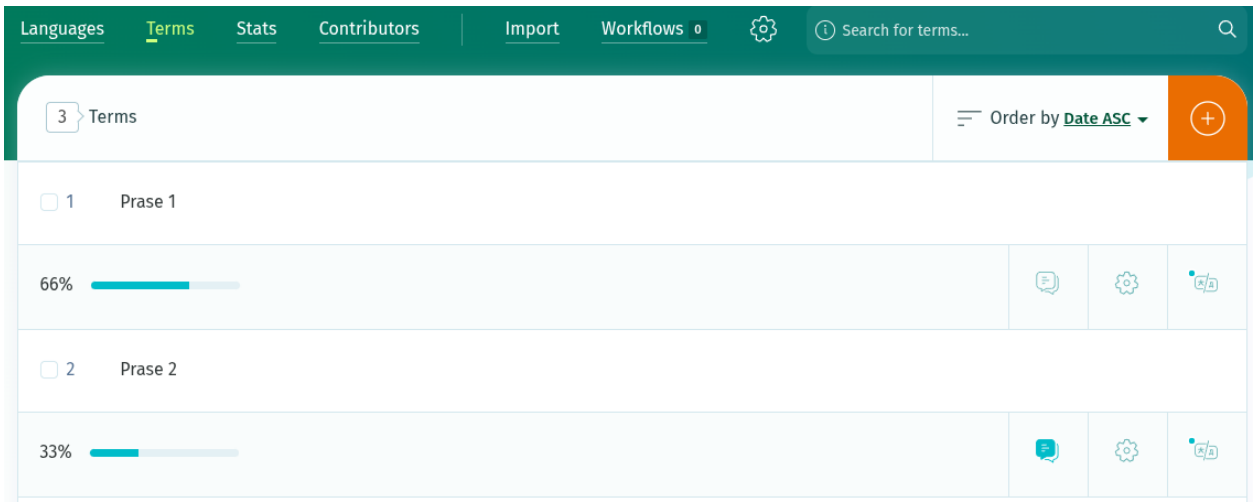


Рисунок 1.4 – Сторінка перегляду списку фраз вебзастосунку POEditor

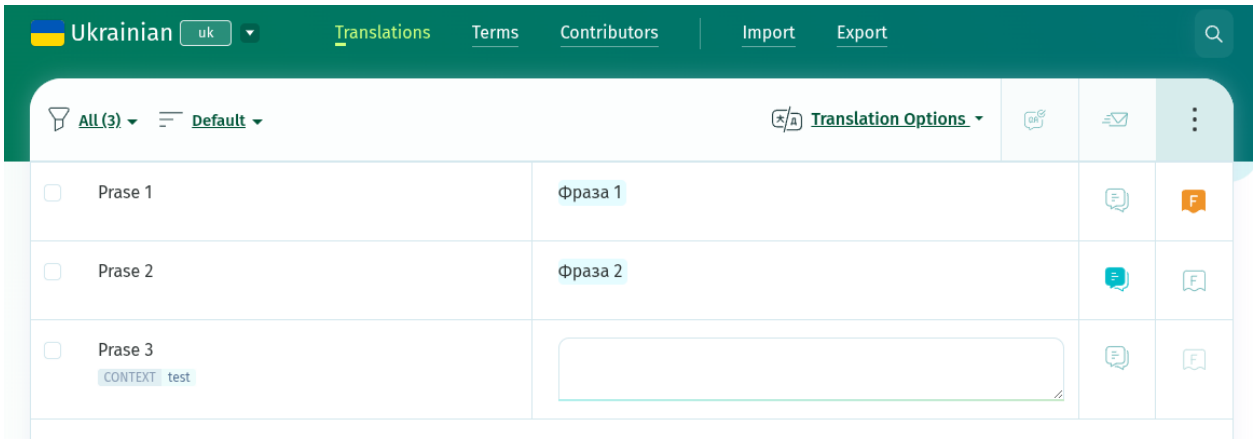


Рисунок 1.5 – Сторінка перегляду списку перекладів вебзастосунку POEditor

Transifex та Lokalise – це комерційні платформи, що мають розширені функції управління командами перекладачів та робочими процесами локалізації.

Спільними недоліками зазначених платформ є висока вартість підписки, обмеження на кількість фраз або мов у безкоштовних планах та зберігання даних на серверах сторонніх постачальників, що не завжди прийнятно для проєктів із підвищеними вимогами до конфіденційності.

Проведений аналіз існуючого програмного забезпечення для управління перекладами програмних продуктів показав, що на ринку відсутній простий інструмент управління перекладами із підтримкою

основних форматів ресурсних файлів та REST API, який дав би команді повний контроль над даними перекладів. Для усунення цього недоліку пропонується розробка вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення.

1.3 Постановка задачі на розробку вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення

Виходячи з виконаного аналізу практичної задачі інженерії програмного забезпечення та аналізу існуючого програмного забезпечення для управління перекладами, постановка задачі є наступною: необхідно розробити вебзастосунок управління перекладами для безперебійної локалізації програмного забезпечення.

Користувачів вебзастосунку поділено на дві групи:

- користувачі (перекладачі, розробники та менеджери проєктів);
- адміністратори.

У межах своєї ролі користувач виконує такі завдання:

1) Перегляд списку проєктів локалізації, у яких користувач є учасником.

2) Перегляд проєкту з оглядом доданих мов та прогресу перекладу по кожній мові.

3) Перегляд списку фраз проєкту з пошуком за текстом оригіналу або перекладу.

4) Інтерактивний переклад фрази на будь-яку з мов проєкту з урахуванням контексту та коментаря.

5) Редагування тексту оригіналу фрази, її контексту та коментаря.

6) Масове видалення фраз проєкту.

7) Імпорт ресурсного файлу локалізації (формати PO, JSON, CSV) для обраної мови проєкту.

8) Експорт перекладів проєкту у потрібному форматі (PO, JSON, XLIFF, CSV).

9) Генерація персонального API-ключа для машинного доступу до перекладів.

10) Отримання перекладів проєкту через REST API за персональним ключем.

11) Зміна власного пароля та перегляд персональних даних.

Адміністратор має всі можливості користувача, а також додаткові завдання:

1) Додавання нових користувачів (разом зі створенням облікового запису), коригування та видалення даних про наявних користувачів.

2) Перегляд списку всіх користувачів вебзастосунку.

3) Створення нових проєктів локалізації, коригування та видалення наявних проєктів.

4) Додавання мов до проєкту, коригування та видалення мов проєкту.

5) Призначення учасників проєкту з числа зареєстрованих користувачів.

Вебзастосунок повинен забезпечувати виконання таких функцій:

- створення та управління проєктами локалізації;

- імпорт ресурсних файлів локалізації (формати PO, JSON, CSV);

- інтерактивний інтерфейс для перекладу фраз із підтримкою контексту та коментарів;

- підтримка множинних мов у рамках одного проєкту;

- експорт готових локалізацій у потрібних форматах;

- надання REST API для автоматичної синхронізації перекладів із кодовою базою;

- пошук та фільтрація фраз за критеріями: текст оригіналу або перекладу;

- відображення прогресу перекладу по кожній мові;

- автентифікація користувачів із розмежуванням ролей (адміністратор, користувач);

- управління користувачами адміністратором (додавання, редагування, видалення).

Вимоги до складу виконуваних функцій

Нижче наведено перелік функцій вебзастосунку разом з їхніми вхідними та вихідними даними.

1) Авторизація

Вхідні дані: електронна пошта, пароль.

Вихідні дані: повідомлення про успішність авторизації.

2) Додавання нового користувача

Вхідні дані: електронна пошта, ім’я, прізвище, роль (адміністратор або користувач).

Вихідні дані: повідомлення про додавання користувача та згенерований пароль для його облікового запису.

3) Оновлення даних про користувача

Вхідні дані: номер користувача, електронна пошта, ім’я, прізвище, роль.

Вихідні дані: повідомлення про оновлення даних про користувача.

4) Видалення користувача

Вхідні дані: номер користувача.

Вихідні дані: повідомлення про видалення користувача.

5) Зміна пароля

Вхідні дані: поточний пароль, новий пароль, підтвердження нового пароля.

Вихідні дані: повідомлення про зміну пароля.

6) Генерація персонального API-ключа

Вхідні дані: запит на генерацію ключа.

Вихідні дані: згенерований персональний API-ключ користувача.

7) Створення нового проєкту

Вхідні дані: назва проєкту, опис проєкту.

Вихідні дані: повідомлення про створення проєкту.

8) Оновлення даних про проєкт

Вхідні дані: номер проєкту, назва проєкту, опис проєкту.

Вихідні дані: повідомлення про оновлення даних про проєкт.

9) Видалення проєкту

Вхідні дані: номер проєкту.

Вихідні дані: повідомлення про видалення проєкту.

10) Додавання мови до проєкту

Вхідні дані: номер проєкту, мова (код та назва).

Вихідні дані: повідомлення про додавання мови до проєкту.

11) Видалення мови проєкту

Вхідні дані: номер проєкту, номер мови.

Вихідні дані: повідомлення про видалення мови проєкту.

12) Призначення учасників проєкту

Вхідні дані: номер проєкту, перелік користувачів.

Вихідні дані: повідомлення про оновлення складу учасників проєкту.

13) Додавання нової фрази

Вхідні дані: номер проєкту, текст оригіналу, контекст, коментар для перекладача.

Вихідні дані: повідомлення про додавання фрази.

14) Оновлення даних про фразу

Вхідні дані: номер фрази, текст оригіналу, контекст, коментар для перекладача.

Вихідні дані: повідомлення про оновлення даних про фразу.

15) Видалення фраз

Вхідні дані: перелік номерів фраз.

Вихідні дані: повідомлення про видалення фраз.

16) Збереження перекладу фрази

Вхідні дані: номер фрази, номер мови, текст перекладу.

Вихідні дані: повідомлення про збереження перекладу.

17) Імпорт ресурсного файлу локалізації

Вхідні дані: номер проєкту, номер мови, файл формату PO, JSON або CSV.

Вихідні дані: повідомлення про кількість імпортованих фраз та перекладів.

18) Експорт локалізації

Вхідні дані: номер проєкту, номер мови, формат експорту (PO, JSON, XLIFF або CSV).

Вихідні дані: файл локалізації обраного формату.

19) Формування інформації про проєкти

Вхідні дані: запит на перегляд списку проєктів, у яких користувач є учасником, або номер проєкту для перегляду інформації про певний проєкт.

Вихідні дані:

– список проєктів: номер проєкту, назва, перелік доданих мов;

– інформація про певний проєкт: назва, опис, перелік мов, прогрес перекладу по кожній мові, перелік учасників.

20) Формування інформації про фрази

Вхідні дані: номер проєкту, опціональні критерії пошуку за текстом оригіналу або перекладу.

Вихідні дані: список фраз проєкту: текст оригіналу, контекст, наявність перекладу для кожної мови.

21) Формування інформації про прогрес перекладу

Вхідні дані: номер проєкту.

Вихідні дані: кількість перекладених фраз та частка перекладених фраз для кожної мови проєкту.

22) Отримання перекладів через REST API

Вхідні дані: GET-запит до точки доступу експорту з персональним API-ключем у заголовку Authorization, номер проєкту.

Вихідні дані: переклади проєкту у форматі JSON.

Вимоги до організації вхідних та вихідних даних

Вхідні дані надходять із двох джерел: ресурсних файлів локалізації, що завантажуються на сервер, і полів та форм вебінтерфейсу, які заповнює користувач. Тип поля визначає характер значення – текст, число або вибір зі списку готових варіантів (наприклад, мова чи формат експорту). Окремим джерелом вхідних даних є запити зовнішніх систем до REST API.

Вхідні дані:

- файли ресурсів локалізації у форматах PO (Portable Object), JSON, CSV;

- текстові дані перекладів, що вводяться перекладачами через вебінтерфейс;

- запити до REST API від зовнішніх систем.

Вихідні дані – це результати виконаних операцій: екранні форми та повідомлення в інтерфейсі, згенеровані файли локалізації, а також відповіді REST API. Вихідні дані вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення:

- експортовані файли локалізації у форматах PO, JSON, XLIFF та CSV;

- відповіді REST API у форматі JSON.

Вимоги до складу та параметрів технічних засобів, до інформаційної та програмної сумісності, а також до надійності наведено у технічному завданні (додаток А).

2 РОЗРОБКА ВЕБЗАСТОСУНКУ УПРАВЛІННЯ ПЕРЕКЛАДАМИ ДЛЯ БЕЗПЕРЕБІЙНОЇ ЛОКАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз вимог до вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення

Дії з вебзастосунком виконуються двома типами акторів – адміністратором та користувачем (перекладачем, розробником або менеджером проєкту). Варіанти використання поділяються на дві групи.

Для адміністратора: керування користувачами (додавання, редагування, видалення), керування проєктами локалізації, керування мовами проєкту, призначення учасників проєкту, імпорт ресурсних файлів локалізації, експорт перекладів.

Для користувача: перегляд проєктів, у яких користувач є учасником, перегляд та редагування фраз і перекладів, експорт перекладів, звернення до REST API за персональним ключем.

Основні варіанти використання – «Створити проєкт», «Імпортувати ресурсний файл», «Перекласти фразу», «Експортувати локалізацію» та «Отримати переклади через REST API». Авторизація користувачів виконується за електронною поштою і паролем; для машинного доступу через REST API передбачено персональний API-ключ, що генерується для кожного користувача.

Діаграму варіантів використання вебзастосунку зображено на рисунку 2.1.

Рисунок 2.1 – Діаграма варіантів використання вебзастосунку управління перекладами

Специфікації основних варіантів використання наведено у таблицях 2.1–2.6.

Таблиця 2.1 – Специфікація варіанта використання «Створити проєкт»

Складова	**Опис**
Назва прецеденту	Створити проєкт
Опис прецеденту	Адміністратор створює новий проєкт локалізації програмного забезпечення
Актори	Адміністратор
Передумови	Користувач автентифікований у системі та має роль «адміністратор»
Базовий потік	Адміністратор відкриває сторінку списку проєктів. Натискає посилання «Add». Вводить назву та опис проєкту. Натискає кнопку «Add».
Альтернативний потік	Якщо назва порожня або перевищує 255 символів – відображається повідомлення про помилку.
Післяумови	Новий проєкт додано в базу даних із унікальним ідентифікатором

Таблиця 2.2 – Специфікація варіанта використання «Імпортувати ресурсний файл»

Складова	**Опис**
Назва прецеденту	Імпортувати ресурсний файл
Опис прецеденту	Користувач завантажує ресурсний файл локалізації програмного забезпечення
Актори	Адміністратор
Передумови	Існує проєкт із доданою мовою
Базовий потік	Користувач відкриває сторінку проєкту. Обирає мову. Натискає посилання «Import». Обирає файл (PO, JSON або CSV). Натискає кнопку «Import».
Альтернативний потік	Якщо формат файлу не підтримується або структура пошкоджена – відображається опис помилки.
Післяумови	Фрази та переклади з файлу імпортовано до проєкту

Таблиця 2.3 – Специфікація варіанта використання «Перекласти фразу»

Складова	**Опис**
Назва прецеденту	Перекласти фразу
Опис прецеденту	Користувач вводить переклад фрази на одну з мов проєкту
Актори	Користувач, адміністратор
Передумови	Користувач є учасником проєкту, у проєкті існує фраза та додано хоча б одну мову
Базовий потік	Користувач відкриває сторінку списку фраз. Обирає фразу. Вводить текст перекладу для потрібної мови. Натискає кнопку «Save».
Альтернативний потік	Якщо текст перекладу порожній – переклад видаляється; якщо перевищує максимальний розмір – відображається помилка валідації.
Післяумови	Переклад збережено у базі даних

Таблиця 2.4 – Специфікація варіанта використання «Експортувати локалізацію»

Складова	**Опис**
1	2
Назва прецеденту	Експортувати локалізацію
Опис прецеденту	Користувач завантажує переклади проєкту у потрібному форматі
Актори	Користувач, адміністратор

Кінець таблиці 2.4

1	2
Передумови	Користувач є учасником проєкту, у проєкті є хоча б один переклад
Базовий потік	Користувач відкриває сторінку проєкту. Обирає мову та формат експорту. Натискає кнопку «Export».
Післяумови	Файл локалізації обраного формату завантажено клієнту

Таблиця 2.5 – Специфікація варіанта використання «Отримати переклади через REST API»

Складова	**Опис**
Назва прецеденту	Отримати переклади через REST API
Опис прецеденту	Зовнішня система отримує переклади у форматі JSON через HTTP-запит
Актори	Зовнішня система (від імені користувача)
Передумови	Користувач має API-ключ та є учасником проєкту
Базовий потік	Зовнішня система виконує GET-запит до точки доступу /api/projects/export з заголовком Authorization. Сервер повертає JSON з перекладами.
Альтернативний потік	Якщо API-ключ невалідний – повертається HTTP-код 401.
Післяумови	Зовнішня система отримала актуальні переклади

Таблиця 2.6 – Специфікація варіанта використання «Керувати персональним API-ключем»

Складова	**Опис**
Назва прецеденту	Керувати персональним API-ключем
Опис прецеденту	Користувач генерує новий персональний API-ключ для машинного доступу до перекладів
Актори	Користувач, адміністратор
Передумови	Користувач автентифікований у системі
Базовий потік	Користувач відкриває сторінку персональних ключів. Натискає кнопку «Generate». Система генерує новий API-ключ і відображає його користувачеві.
Альтернативний потік	Якщо ключ уже існує – попередній ключ стає недійсним після генерації нового.
Післяумови	Новий API-ключ збережено та може використовуватись для звернень до REST API

Вимоги до складу та параметрів технічних засобів, до інформаційної та програмної сумісності, а також до надійності наведено у технічному завданні (додаток А).

2.2 Архітектура вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення

Для розробки вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення обрано архітектурний шаблон MVC (Model-View-Controller), який поділяє ПЗ на три компоненти [4]:

1) Model (модель): відповідає за управління даними та бізнес-логіку застосунку. Цей шар інкапсулює доступ до бази даних, валідацію введених значень і правила предметної області, тож контролери та вигляди не звертаються до бази напряму [4]. Модель може включати класи двох типів: Table-класи описують роботу з таблицею загалом – пошук записів, збереження, валідацію та зв'язки між таблицями, а Entity-класи представляють окремий запис таблиці у вигляді об'єкта з його полями. Завдяки такому поділу вся логіка доступу до даних зосереджена в одному шарі;

2) View (вигляд): відповідає за відображення даних користувачу через HTML-шаблони, що дає змогу змінювати зовнішній вигляд застосунку, не торкаючись логіки роботи з даними [4];

3) Controller (контролер): обробляє HTTP-запити користувача, отримує дані з моделі та передає їх у вигляд [4]. Контролер приймає запит, звертається до потрібних Table-класів моделі за даними, виконує над ними необхідні дії та передає результат у вигляд для відображення. Кожен контролер відповідає за окрему групу дій.

Взаємодію компонентів за шаблоном MVC зображено на рисунку 2.2.

Такий розподіл відокремлює доступ до даних від їх відображення, що дає змогу тестувати моделі окремо від інтерфейсу та змінювати один із шарів, не зачіпаючи інші [5].

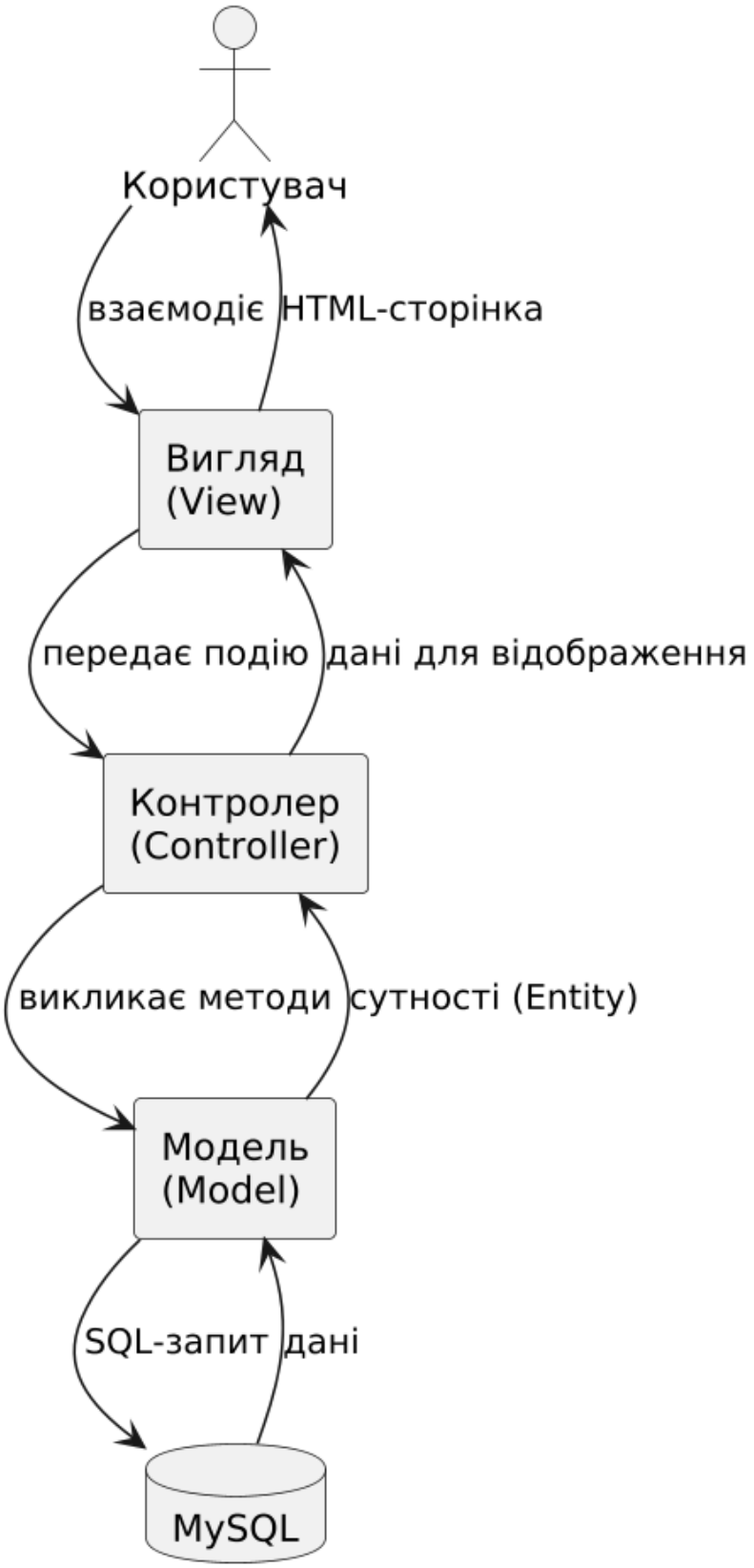


Рисунок 2.2 – Діаграма взаємодії архітектури MVC

2.3 Проєкт вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення

2.3.1 Ескізний проєкт вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення

В ескізному проєкті вебзастосунку управління перекладами визначено основні сценарії взаємодії користувача з вебзастосунком, концептуальну модель даних та прототип інтерфейсу користувача.

Сутностями предметної області є:

- користувач (User),
- проєкт локалізації (Project),
- мова (Locale),
- фраза (Term)
- переклад (Translation).

Між сутностями встановлено такі зв’язки:

- проєкт містить багато фраз (Project 1-N Term);
- фраза має по одному перекладу на кожну мову проєкту (Term 1-N Translation);
- проєкт пов’язаний із мовами через таблицю-зв’язок locales_projects (Project N-M Locale);
- користувач пов’язаний із проєктами через таблицю-зв’язок projects_users (User N-M Project); ролі «адміністратор» і «користувач» зберігаються у полі users.role.

Концептуальну модель даних предметної області зображено на рисунку 2.3.

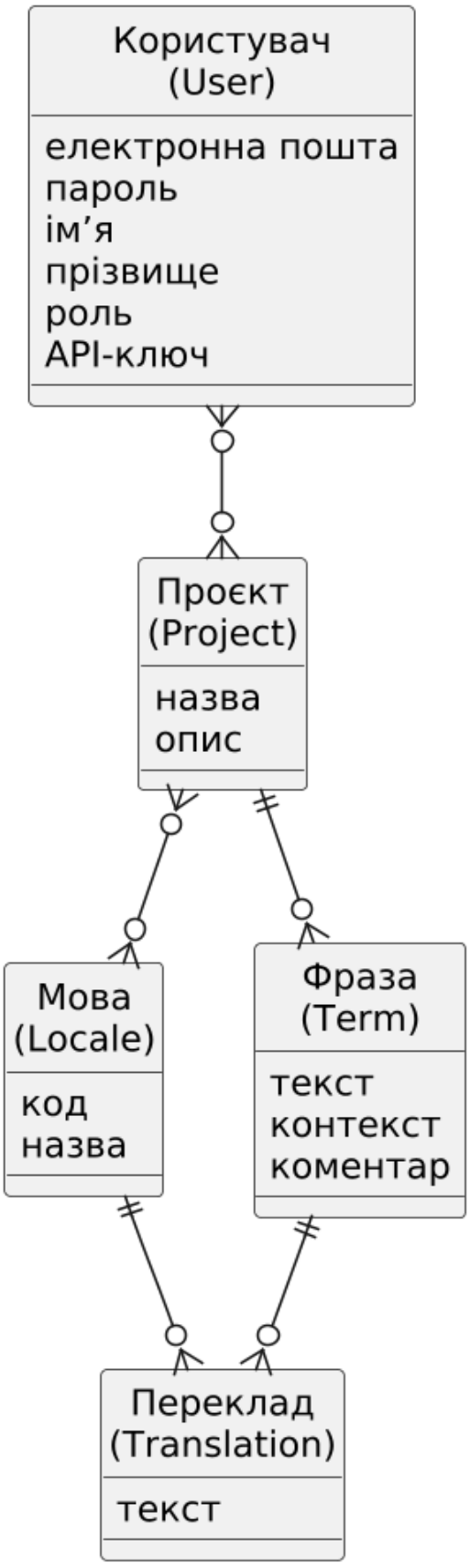


Рисунок 2.3 – Концептуальна модель даних вебзастосунку

Прототип інтерфейсу користувача складається з:

- сторінки авторизації (рисунок 2.4);
- сторінки списку проєктів (рисунок 2.5);
- сторінки додавання проєкту (рисунок 2.6);
- сторінки проєкту з оглядом мов (рисунок 2.7);
- сторінки зі списком фраз проєкту (рисунок 2.8);
- сторінки інтерактивного перекладу фрази (рисунок 2.9);
- сторінки управління персональним API-ключем (рисунок 2.10).

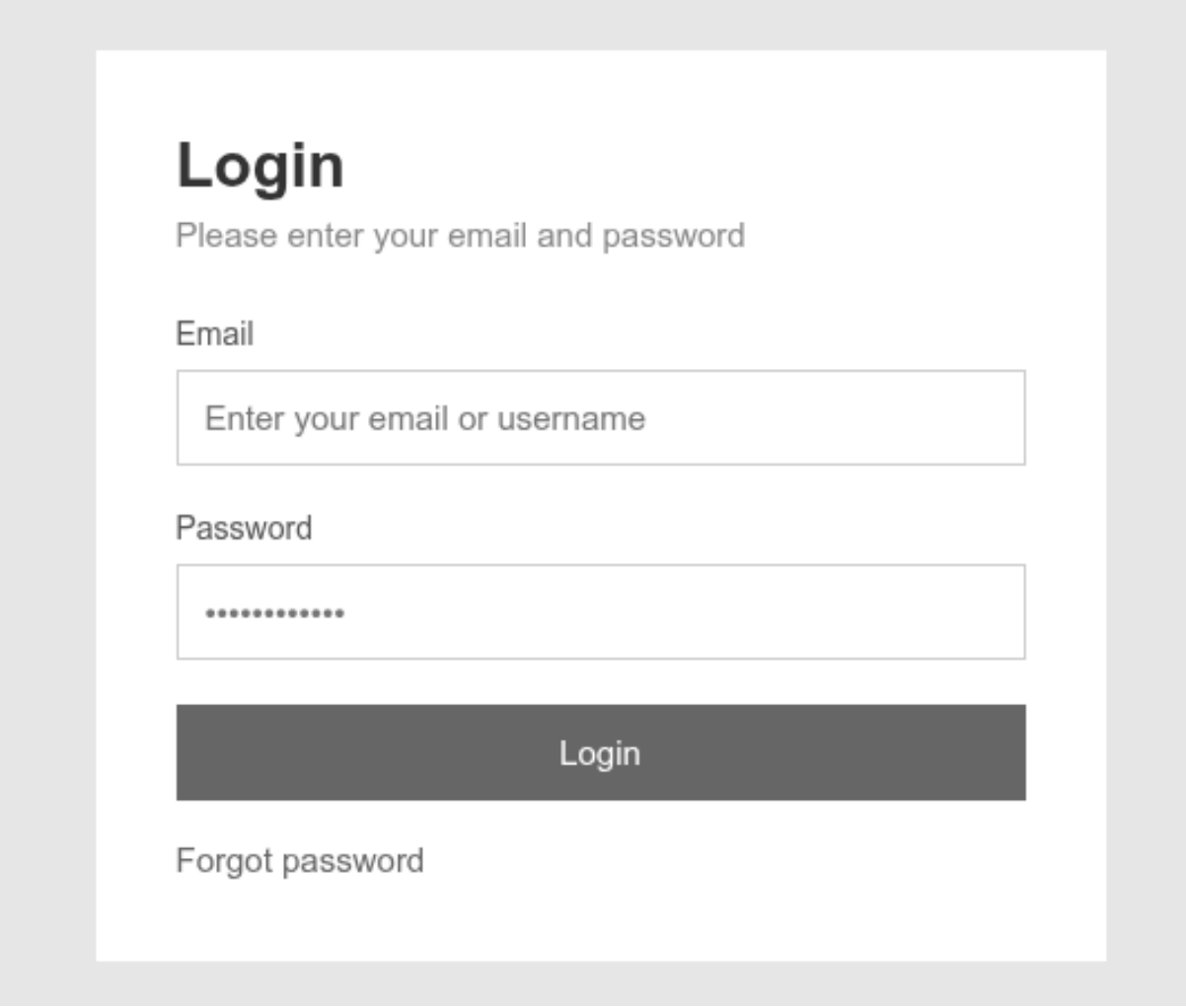


Рисунок 2.4 – Прототип сторінки авторизації користувача

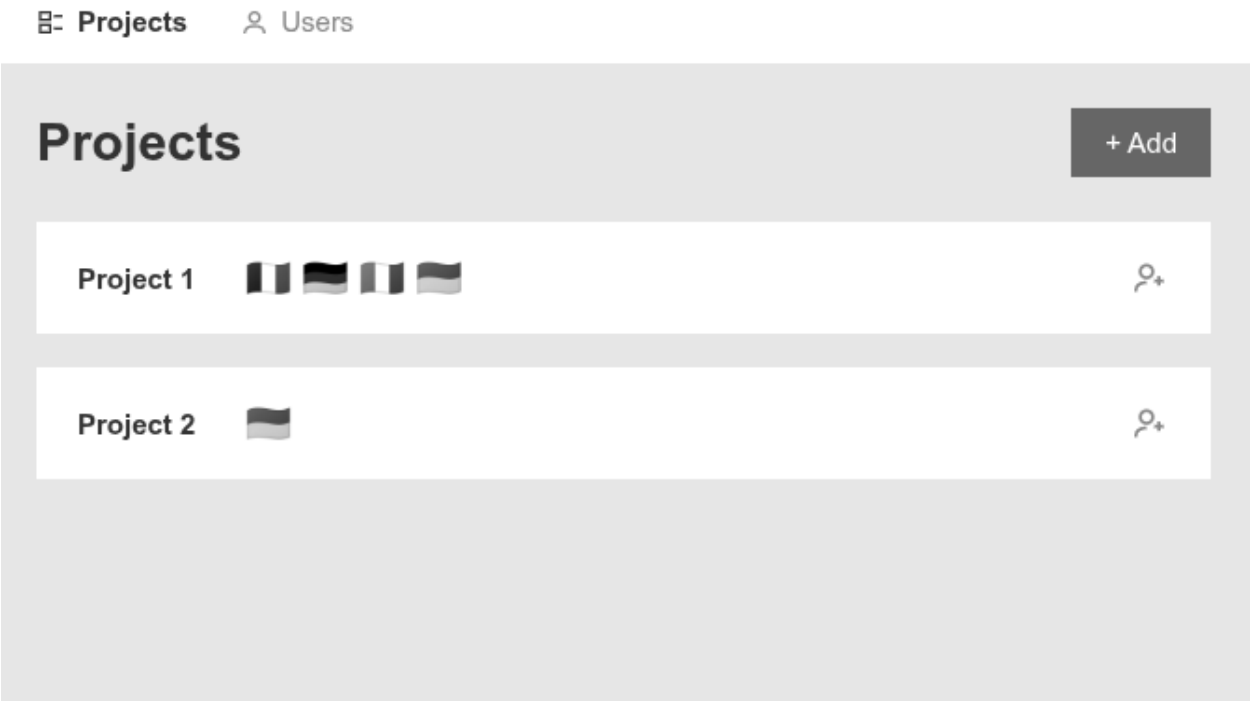


Рисунок 2.5 – Прототип сторінки списку проєктів локалізації програмного забезпечення

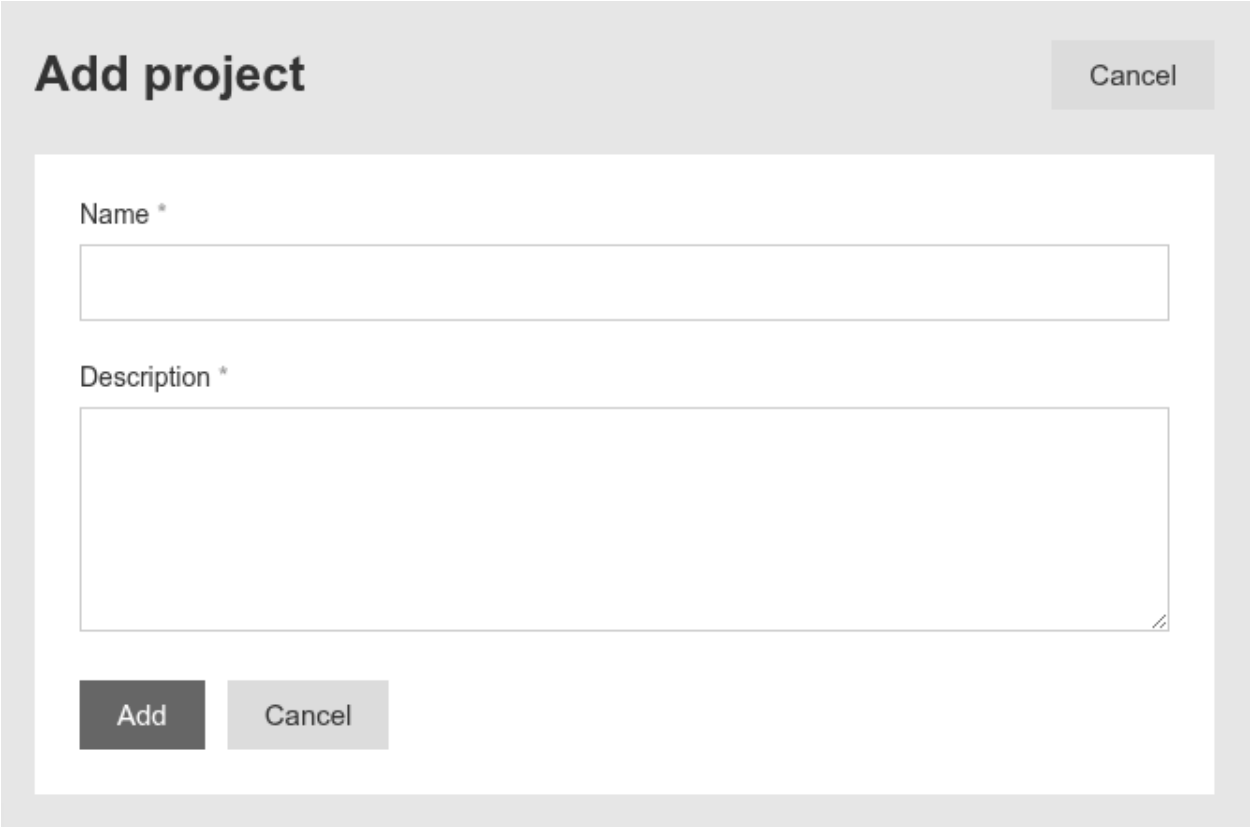


Рисунок 2.6 – Прототип сторінки додавання проєкту локалізації програмного забезпечення

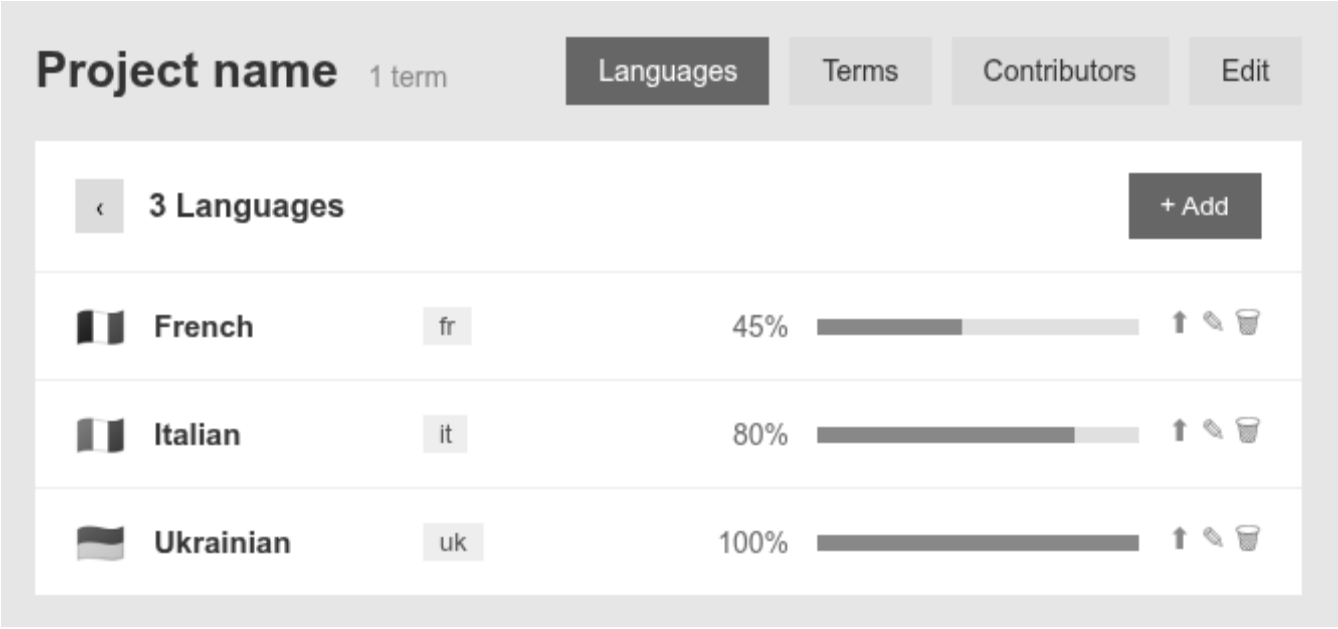


Рисунок 2.7 – Прототип сторінки проєкту з оглядом мов проєкту

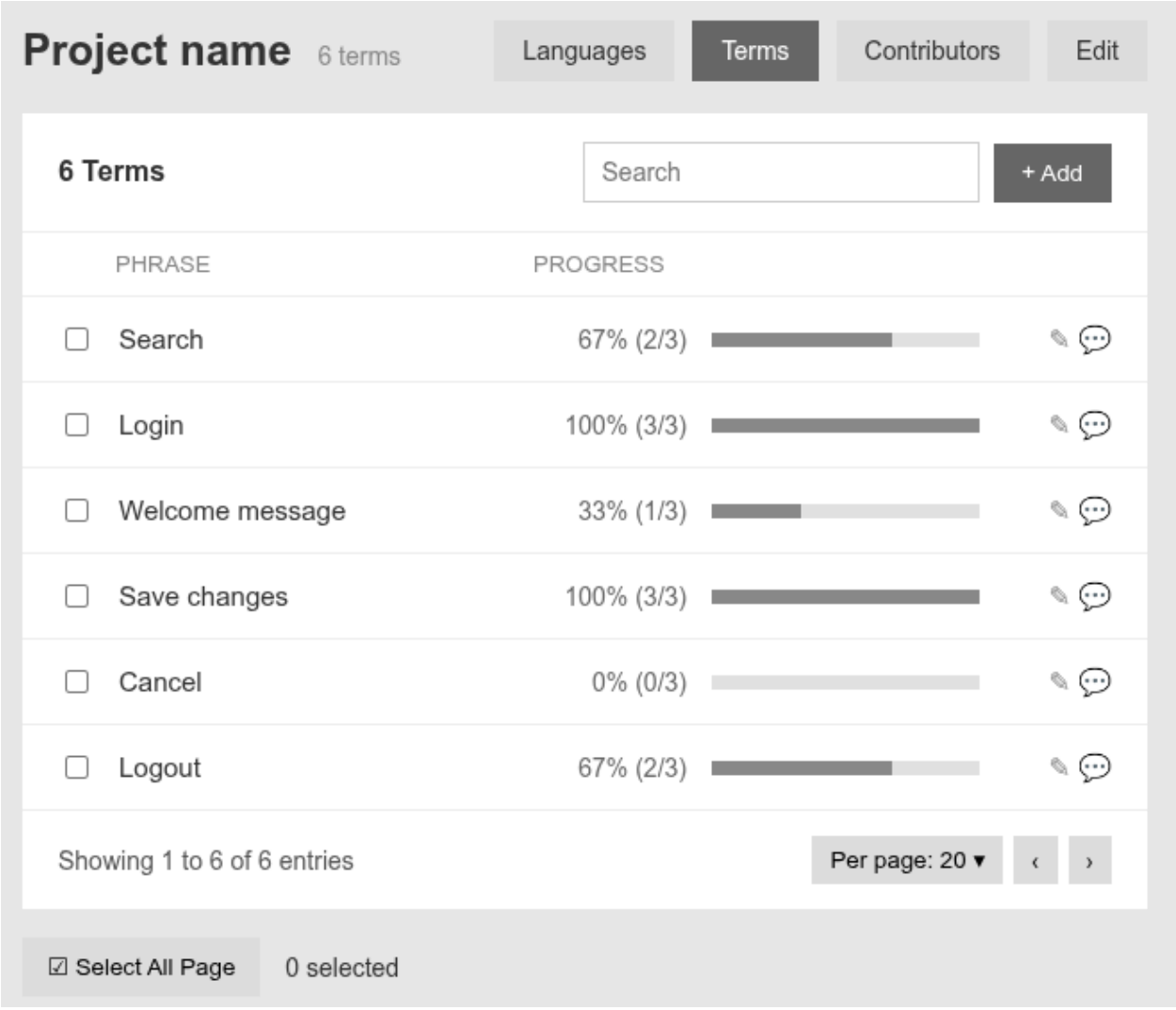


Рисунок 2.8 – Прототип сторінки зі списком фраз проєкту

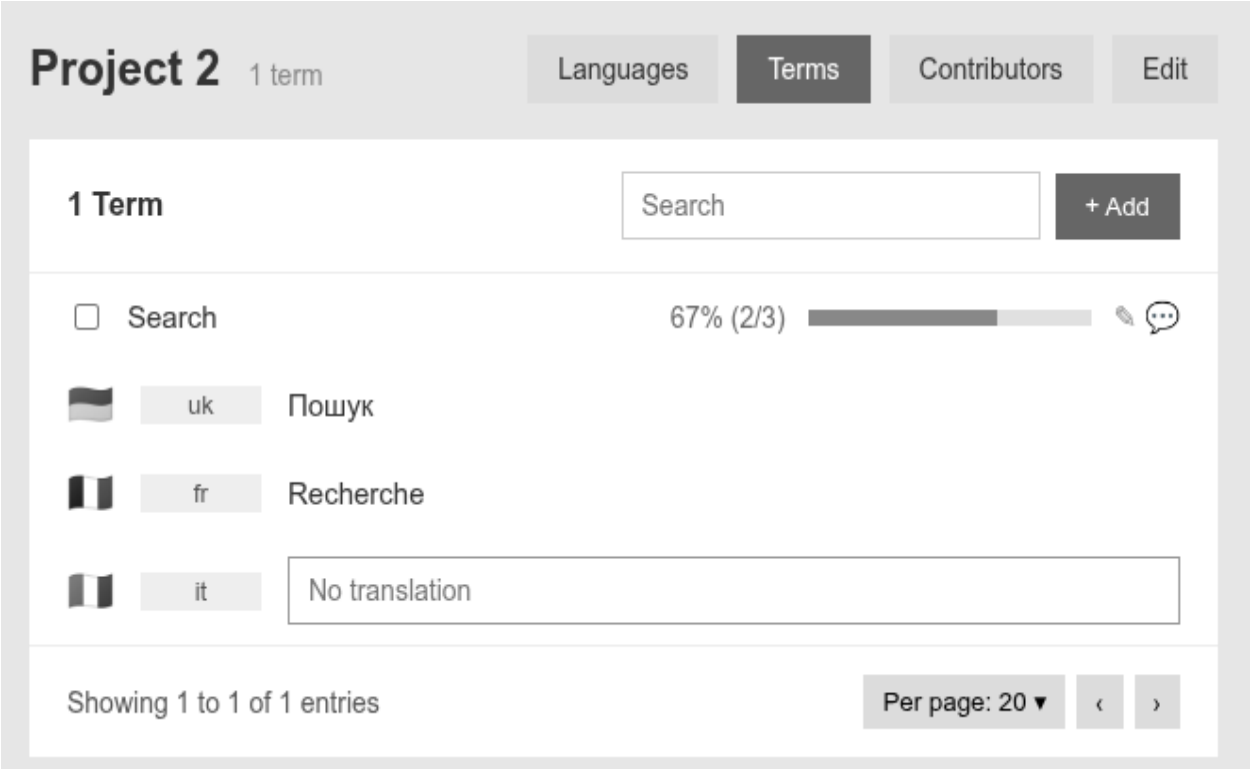


Рисунок 2.9 – Прототип сторінки інтерактивного перекладу фрази

Рисунок 2.10 – Прототип сторінки управління персональним API-ключем

Діаграму діяльності для варіанта використання «Імпортувати ресурсний файл» зображено на рисунку 2.11.

Діаграму діяльності для варіанта використання «Експортувати ресурсний файл» зображено на рисунку 2.12.

Рисунок 2.11 – Діаграма діяльності для варіанта використання «Імпортувати ресурсний файл»

Рисунок 2.12 – Діаграма діяльності для варіанта використання «Експортувати локалізацію»

2.3.2 Технічний проєкт вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення

Технічний проєкт вебзастосунку управління перекладами деталізує статичну та динамічну моделі, а також логічну модель даних, які є основою для реалізації програмного забезпечення.

Статична модель – це набір класів відповідно до архітектурного шаблону MVC:

- моделі (Table-класи: UsersTable, ProjectsTable, LocalesTable, TermsTable, TranslationsTable, ProjectsUsersTable; Entity-класи: User, Project, Locale, Term, Translation);
- контролери (UsersController, ProjectsController, TermsController, TranslationsController, ApiController);
- допоміжні класи (Component\AuthComponent, View\AppView).

Діаграму класів вебзастосунку зображено на рисунку 2.13.

Специфікація класу ProjectsController

Призначення: забезпечує дії над проєктами локалізації.

Методи:

index() – список проєктів, доступних поточному користувачеві;

view($id) – перегляд проєкту із оглядом мов та прогресу перекладу;

add(), edit($id) – створення та редагування проєкту;

addLocale($id), editLocale($id, $localeId), deleteLocale($id, $localeId) – керування мовами проєкту;

addContributors($id), contributors($id) – керування учасниками проєкту;

delete($id) – видалення проєкту.

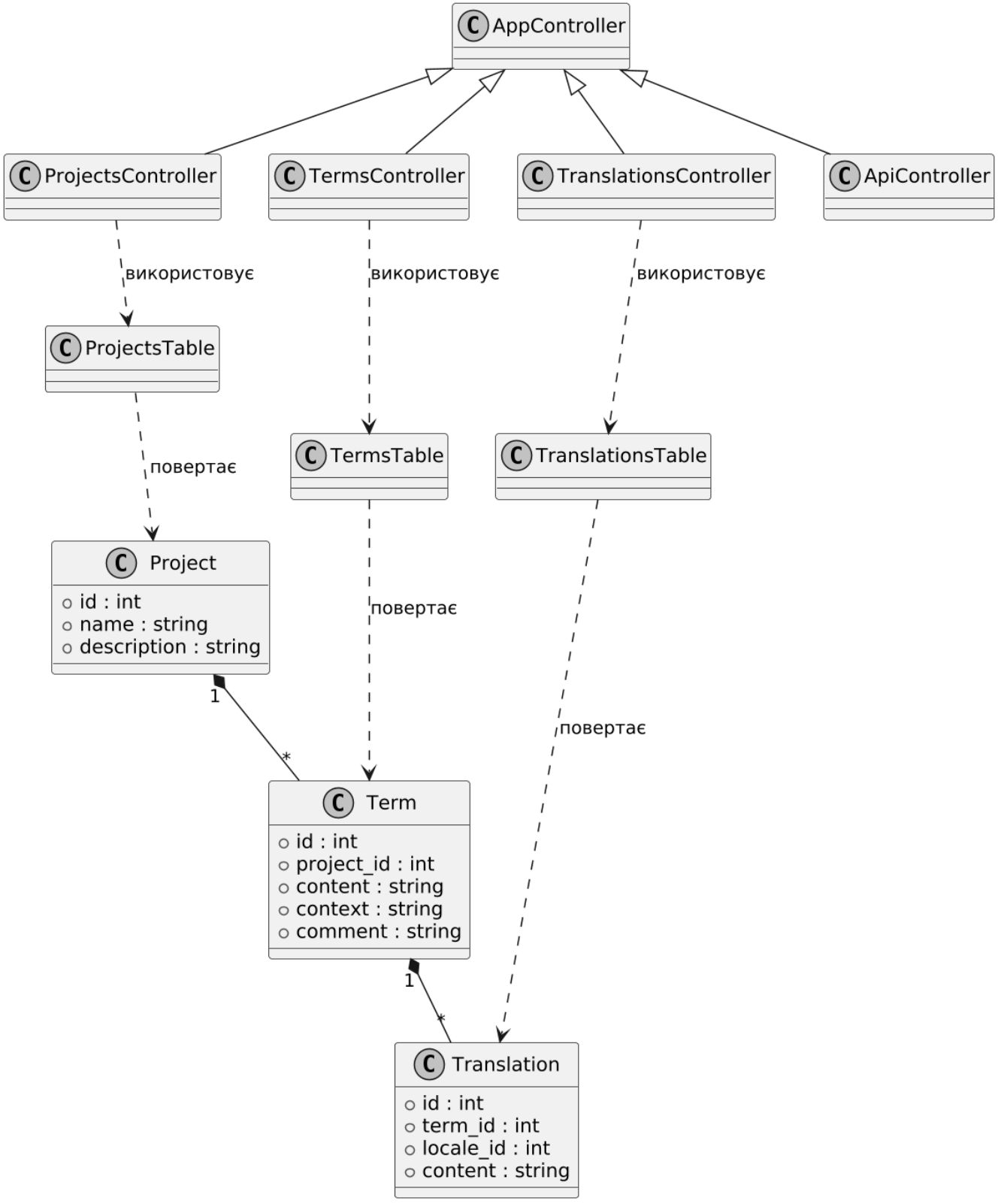


Рисунок 2.13 – Діаграма класів вебзастосунку

Специфікація класу TermsController

Призначення: забезпечує управління фразами проєкту.

Методи:

index($projectId) – список фраз із фільтрацією за наявністю перекладу та текстом;

add($projectId) – додавання нової фрази;

translate($id) – інтерактивний переклад фрази;

saveTerm($id) – збереження редагування фрази;

saveTranslation() – збереження перекладу;

deleteMany() – масове видалення фраз.

Специфікація класу TranslationsController

Призначення: реалізує імпорт-експорт ресурсних файлів локалізації.

Методи:

index($projectId, $localeCode) – список перекладів проєкту для обраної мови;

import($projectId, $localeCode) – імпорт ресурсного файлу формату PO, JSON або CSV;

export($projectId, $localeCode) – експорт перекладів у обраному форматі;

saveTranslation() – збереження окремого перекладу;

deleteMany() – масове видалення перекладів.

Специфікація класу UsersController

Призначення: забезпечує автентифікацію та керування персональними даними користувача.

Методи: login(), logout(), changePassword(), profile(), keys() (генерація API-ключа).

Специфікація класу ApiController

Призначення: забезпечує REST API для зовнішніх систем.

Методи: projectsExport() – повертає переклади у форматі JSON для зовнішніх систем за умови автентифікації за персональним API-ключем.

Специфікація класу Project (Entity)

Атрибути: id: int – ідентифікатор; name: string – назва; description: string – короткий опис проєкту; created, modified: DateTime. Зв’язки: terms: Term[], locales: Locale[], users: User[].

Специфікація класу Term (Entity)

Атрибути: id: int; project_id: int; content: string – текст оригінальної фрази (до 10000 символів); context: string – контекст використання; comment: string – коментар для перекладача; created, modified: DateTime.

Специфікація класу Translation (Entity)

Атрибути: id: int; term_id: int; locale_id: int; content: string – текст перекладу (до 10000 символів); created, modified: DateTime.

Динамічна модель представлена діаграмами послідовності для основних прецедентів вебзастосунку. Діаграму послідовності для прецеденту збереження перекладу зображено на рисунку 2.14, для імпорту ресурсного файлу – на рисунку 2.15, а для експорту локалізації – на рисунку 2.16.

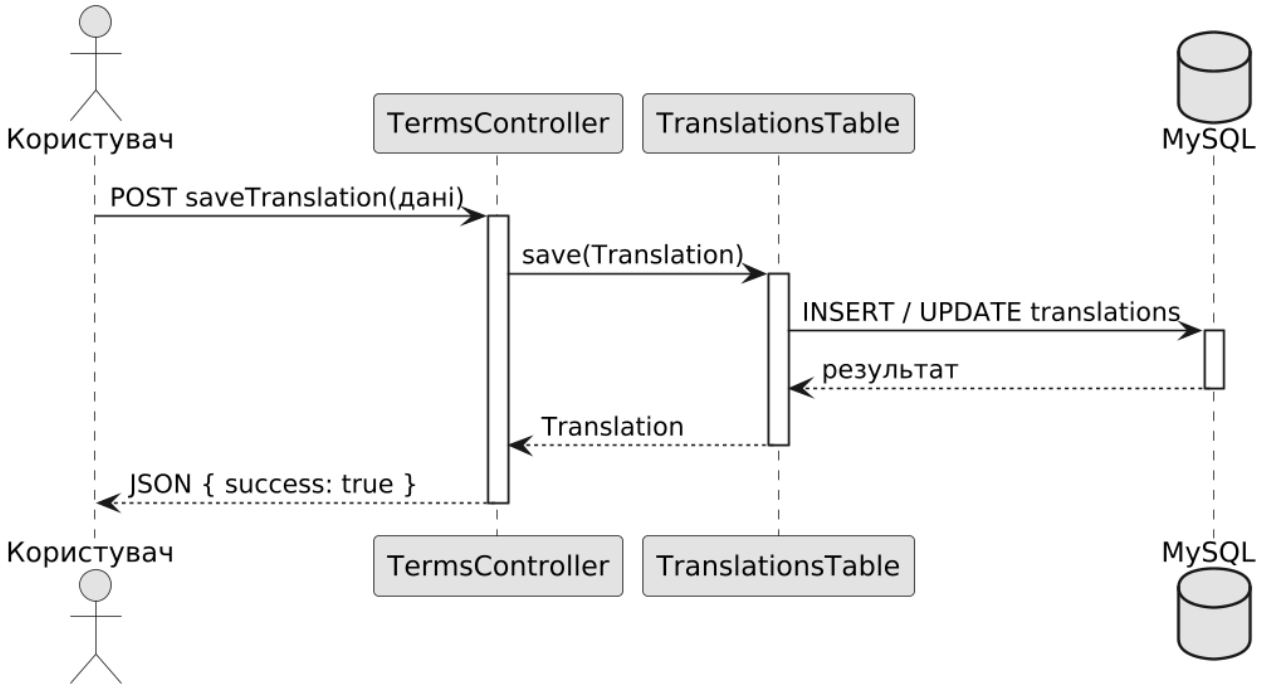


Рисунок 2.14 – Діаграма послідовності для прецеденту «Перекласти фразу»

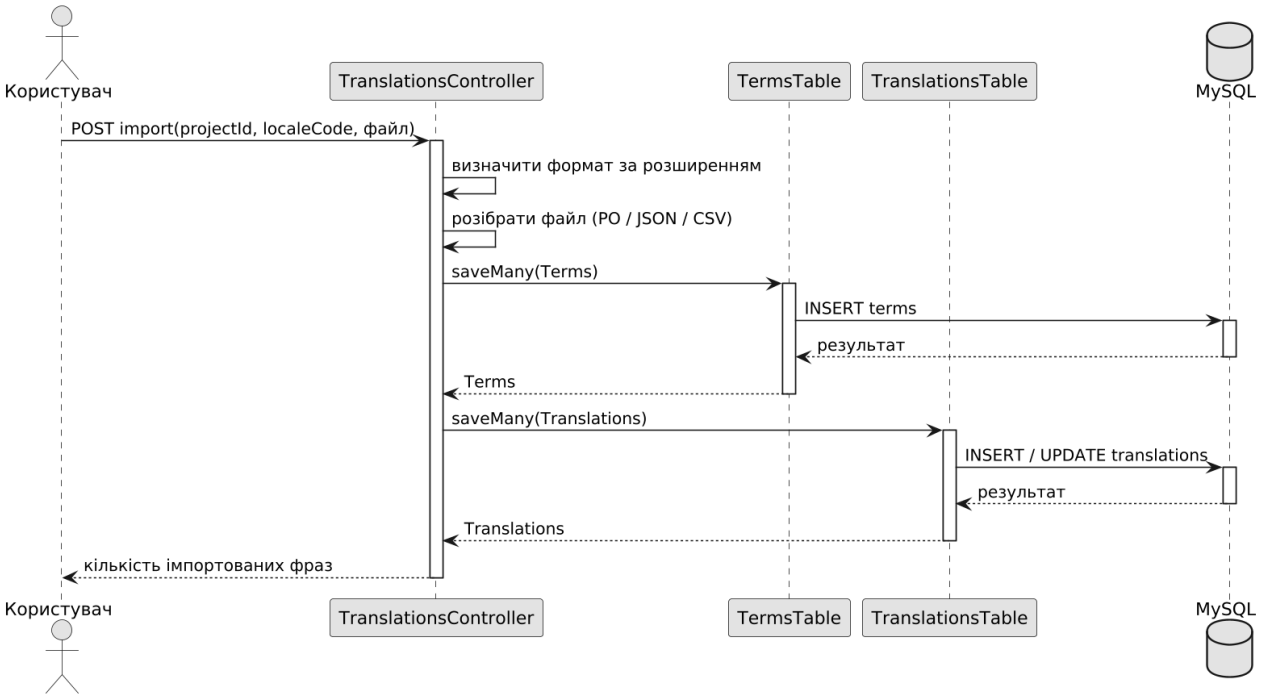


Рисунок 2.15 – Діаграма послідовності для прецеденту «Імпортувати ресурсний файл»

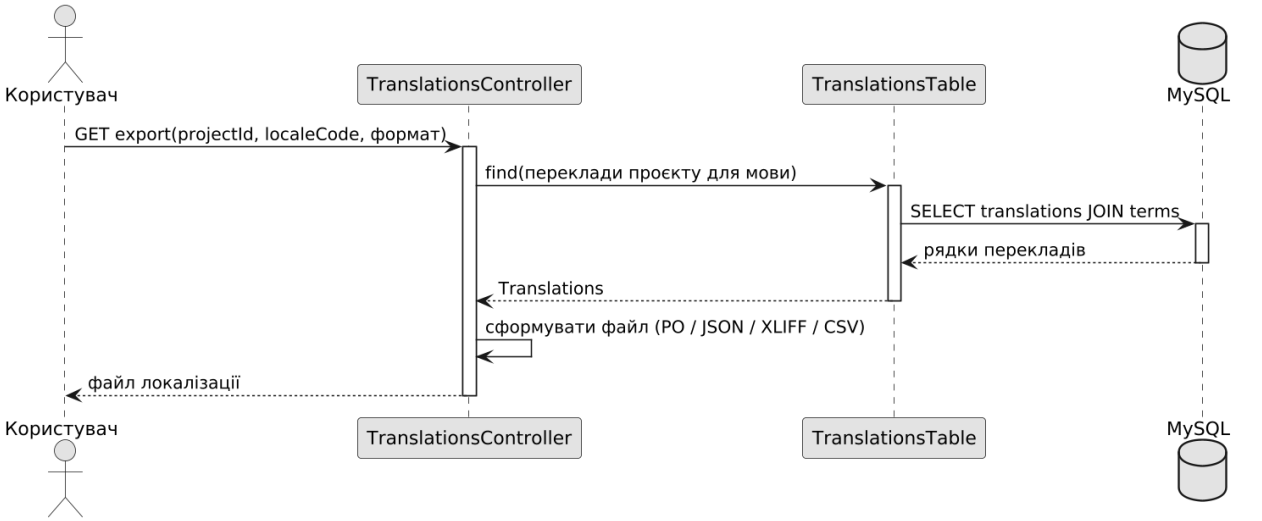


Рисунок 2.16 – Діаграма послідовності для прецеденту «Експортувати локалізацію»

Логічна модель даних включає сутності User, Project, Locale, Term, Translation із атрибутами та зв’язками, що відповідають предметній області, і подана на рисунку 2.17.

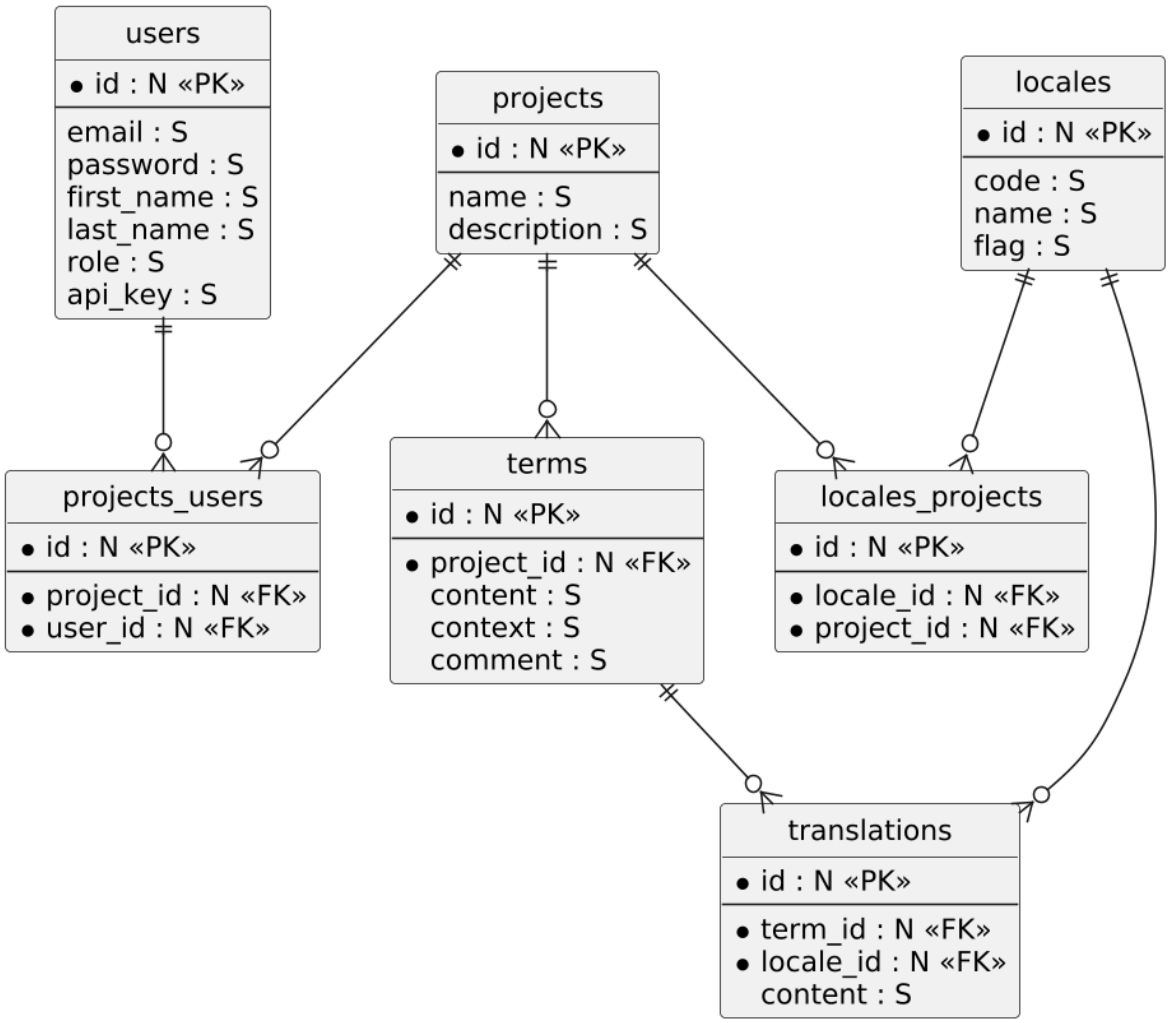


Рисунок 2.17 – Логічна модель даних вебзастосунку

2.3.3 Робочий проект вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення

Обґрунтування вибору засобів розробки

Засоби розробки обрано з урахуванням того, що програмне забезпечення є вебзастосунком: серверна частина має працювати на серверах під управлінням Linux, а клієнтська – у будь-якому сучасному вебоглядачі.

Мова програмування PHP [6] обрана через її поширеність у веброзробці, наявність розвиненої екосистеми бібліотек і фреймворків та офіційну підтримку у більшості хостинг-постачальників [7].

Фреймворк CakePHP 5 обрано як зрілий MVC-фреймворк для PHP із готовими засобами для роботи з базою даних (ORM), маршрутизацією, формами, валідацією, автентифікацією, шаблонами вигляду та REST API [8]. CakePHP 5 потребує PHP 8.1+, що дає змогу використати сувору типізацію та перерахування (enum) для значень фіксованого набору (зокрема ролей користувача).

Систему керування базами даних (СКБД) MySQL обрано як безкоштовну реляційну СКБД, що добре інтегрується з PHP та CakePHP, підтримує транзакції та кодування UTF-8 для багатомовного тексту [9].

Клієнтська частина реалізована на HTML5, з стилями CSS та Javascript із бібліотекою jQuery; для уніфікованого зовнішнього вигляду інтерфейсу використано готовий шаблон Sneat (Bootstrap 5) [10]. REST API працює по протоколу HTTP [11], з використанням формату JSON для передачі даних.

Розробка фізичної моделі даних

Фізична модель даних для СКБД MySQL розроблена на основі відповідної логічної моделі. Фізична модель включає сім основних таблиць: users (користувачі), projects (проєкти локалізації), locales (мови), terms (фрази проєкту), translations (переклади фраз), projects_users (зв’язок користувачів з проєктами для розмежування доступу) та locales_projects (зв’язок мов з проєктами).

Фізичну модель даних зображено на рисунку 2.18.

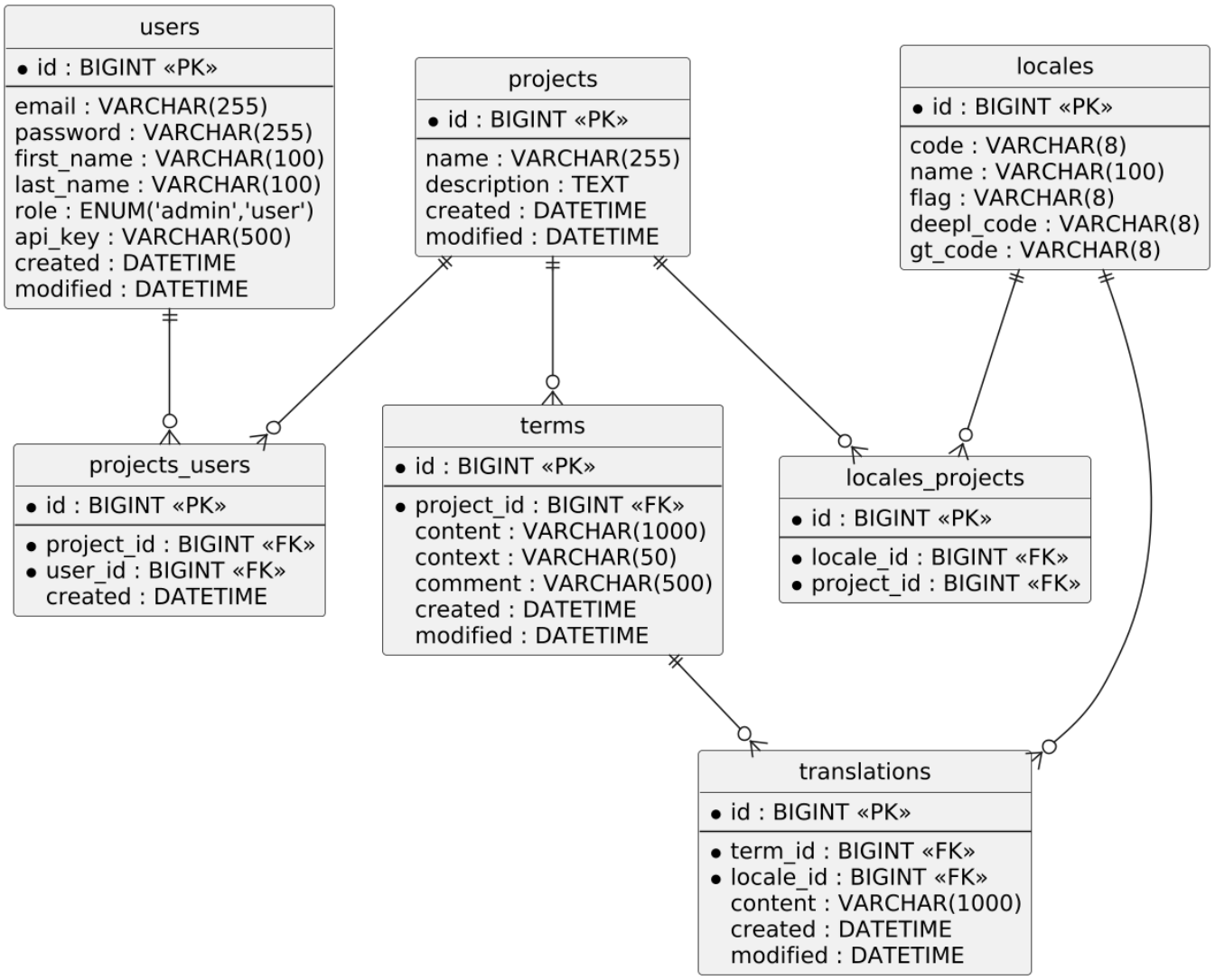


Рисунок 2.18 – Фізична модель даних вебзастосунку для СКБД MySQL

Реалізація вебзастосунку

Серверну частину вебзастосунку реалізовано на мові PHP засобами фреймворку CakePHP 5. Бізнес-логіка розподілена між контролерами (5 класів: ProjectsController, TermsController, TranslationsController, UsersController, ApiController), моделями (Entity- та Table-класами для семи таблиць бази даних) та допоміжними класами (компонентом автентифікації та класом View\AppView).

Імпорт-експорт ресурсних файлів реалізовано у класі TranslationsController методами import та export. Метод import розпізнає формат файлу за розширенням і викликає відповідний обробник для PO,

JSON або CSV. Метод export будує файл потрібного формату з поточних перекладів проєкту та віддає його клієнту.

REST API реалізовано у класі ApiController. Автентифікація запитів виконується за заголовком Authorization із персональним ключем користувача (поле api_key таблиці users). Метод projectsExport віддає переклади у форматі JSON.

Тестування

Тестування виконувалось методом чорної скриньки. За цим методом програму перевіряють лише за її зовнішньою поведінкою: на вхід подають дані й порівнюють отриманий результат з очікуваним, не звертаючись до внутрішньої структури коду [12]. Щоб скоротити кількість перевірок, вхідні дані поділяють на класи еквівалентності – групи значень, які програма обробляє однаково, тож достатньо перевірити по одному представнику з кожного класу.

Для кожної функції вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення визначено правильні та неправильні класи еквівалентності. Класи еквівалентності для основних функцій вебзастосунку наведено у таблиці 2.7.

Таблиця 2.7 – Класи еквівалентності для тестування основних функцій вебзастосунку

Функція	**Правильні класи еквівалентності**	**Неправильні класи еквівалентності**	**Очікуваний результат**
1	2	3	4
Авторизація	Зареєстрована пошта і правильний пароль	Неіснуюча пошта; неправильний пароль; порожні поля	Вхід у систему; повідомлення про помилку авторизації

Кінець таблиці 2.7

1	2	3	4
Створення проєкту	Назва завдовжки від 1 до 255 символів	Порожня назва; назва понад 255 символів	Проєкт збережено; повідомлення про помилку валідації
Збереження перекладу	Текст перекладу завдовжки від 1 до 10000 символів	Текст перекладу понад 10000 символів; порожній текст	Переклад збережено; помилка валідації; порожній переклад видаляється
Імпорт файлу формату PO	Коректний файл PO з валідними записами msgid/msgstr	Пошкоджена структура PO; непідтримуване розширення файлу	Фрази та переклади імпортовано; опис помилки
Імпорт файлу формату JSON	Коректний файл JSON з парами «ключ – переклад»	Невалідний JSON; неочікувана структура; непідтримуване розширення файлу	Фрази та переклади імпортовано; опис помилки
Імпорт файлу формату CSV	Коректний файл CSV з колонками оригіналу та перекладу	Пошкоджена структура CSV; неправильний роздільник; непідтримуване розширення файлу	Фрази та переклади імпортовано; опис помилки
Отримання перекладів через REST API	Дійсний персональний API-ключ	Невалідний або відсутній API-ключ	Переклади у форматі JSON; HTTP-код 401

Результати тестування показали відповідність вебзастосунку функціональним вимогам. Також вебзастосунок адекватно обробляє некоректні вхідні дані й пошкоджені файли ресурсів.

Випробування

Випробування виконувалися відповідно до програми та методики випробувань (додаток Д). За результатами випробувань кожної функції вебзастосунку отримано очікувані результати, що підтверджує відповідність розробленого вебзастосунку вимогам технічного завдання.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ВЕБЗАСТОСУНКУ УПРАВЛІННЯ ПЕРЕКЛАДАМИ ДЛЯ БЕЗПЕРЕБІЙНОЇ ЛОКАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

На основі проєкту було розроблено вебзастосунок управління перекладами для безперебійної локалізації програмного забезпечення.

Вебзастосунок реалізовано на мові PHP засобами фреймворку CakePHP 5 та СКБД MySQL і протестовано за програмою та методикою випробувань. Підтримуються імпорт-експорт чотирьох форматів ресурсних файлів локалізації (PO, JSON, XLIFF, CSV) та REST API для синхронізації перекладів із кодовою базою.

Сторінку зі списком проєктів локалізації, яку бачить користувач після входу, показано на рисунку 3.1.

Рисунок 3.1 – Знімок екрану сторінки зі списком проєктів локалізації

Знімок екрану сторінки проєкту після додавання нової мови показано на рисунку 3.2.

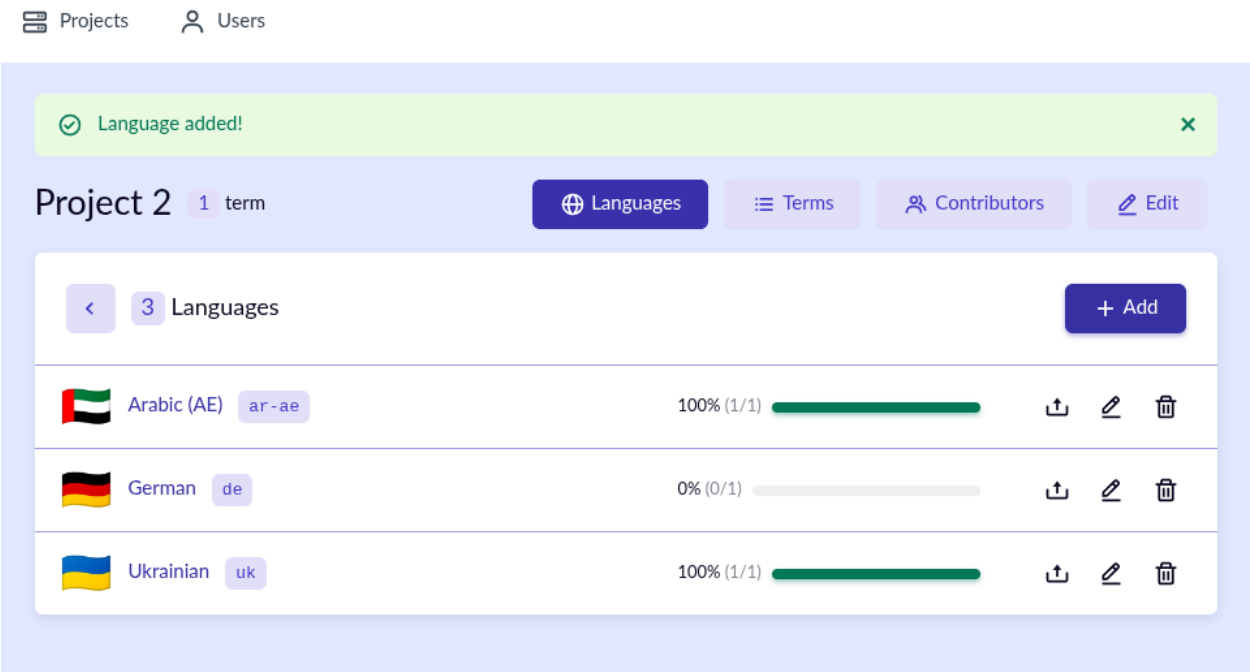


Рисунок 3.2 – Знімок екрану сторінки проєкту після додавання нової мови

Розмір серверної частини вебзастосунку – близько 2 350 рядків коду на мові PHP, розподілених між контролерами, моделями (Entity- та Table-класами) та допоміжними класами.

Клієнтська частина (шаблони HTML/CSS/JavaScript) має близько 1 800 рядків коду.

4 ОХОРОНА ПРАЦІ

Робота з вебзастосунком управління перекладами може виконуватись за комп’ютером протягом тривалого часу, тому особливу увагу слід приділяти безпеці та умовам праці. Охорона праці запобігає професійним ризикам та підтримує здоров’я працівників [13].

4.1 Організація робочого місця

Робоче місце працівника, який виконує завдання за комп’ютером, має бути організоване відповідно до санітарних, гігієнічних і ергономічних норм, щоб мінімізувати фізичне навантаження та забезпечити комфорт. Основні вимоги до робочого місця включають:

Ергономічне обладнання: використання регульованих крісел із підтримкою попереку та спини, столів із можливістю налаштування висоти, а також моніторів, розташованих на відстані 50–70 см від очей і на їхньому рівні. Клавіатура та миша повинні розміщуватися так, щоб руки перебували в природному положенні, а лікті утворювали кут 90°.

Освітлення: забезпечення достатнього природного або штучного освітлення для зниження зорового напруження. Світло має бути рівномірним, без відблисків на екрані монітора.

Вентиляція та мікроклімат: у приміщенні підтримується температура 18–22 °C і вологість 40–60%. Свіже повітря забезпечується регулярним провітрюванням або системами кондиціонування. Слід уникати протягів і різких перепадів температури.

Шумовий фон: рівень шуму в робочому приміщенні повинен бути мінімальним, щоб не відволікати працівника та не створювати додаткового стресу. Для цього рекомендується використовувати шумопоглинаючі матеріали або гарнітури з шумозаглушенням.

Організація простору: робоче місце має бути вільним від зайвих предметів, а кабелі й дроти повинні бути акуратно закріплені, щоб уникнути випадкового травмування або пошкодження обладнання.

4.2 Безпека при роботі з комп’ютерною технікою

Тривала робота за комп’ютером може призводити до професійних ризиків, таких як зорове напруження, захворювання опорно-рухового апарату (наприклад, синдром зап’ястного каналу) або порушення постави. Для їх мінімізації необхідно дотримуватися таких заходів:

Дотримання режиму праці та відпочинку: рекомендуються короткі перерви тривалістю 5–10 хвилин кожні 45–60 хвилин роботи. Під час перерв працівники повинні виконувати вправи для очей (наприклад, фокусування погляду на далеких і близьких об’єктах) і легку фізичну розминку для розслаблення м’язів шиї, плечей і спини.

Захист зору: використання моніторів із технологією зменшення синього світла, налаштування яскравості та контрастності екрана відповідно до умов освітлення.

Електробезпека: комп’ютерне обладнання має бути справним і відповідати стандартам безпеки. Необхідно регулярно перевіряти стан кабелів, розеток і заземлення, а також уникати використання пошкоджених пристроїв.

Правильна постава: працівники повинні сидіти прямо, не сутулитися, з опорою на спинку крісла. Ноги мають стояти на підлозі або на спеціальній підставці, щоб уникнути порушення кровообігу.

Профілактика травм: уникнення тривалого статичного положення рук під час роботи з клавіатурою або мишею. Рекомендується використовувати ергономічні клавіатури та миші, а також виконувати вправи для зап’ясть.

4.3 Навчання з охорони праці

Перед початком роботи працівники повинні пройти інструктаж з охорони праці, який охоплює такі аспекти: правила безпечної експлуатації комп’ютерної техніки, зокрема правильне підключення обладнання та дії в разі несправностей; принципи організації робочого місця; методи профілактики професійних захворювань; виконання вправ для очей і тіла під час перерв; дії в надзвичайних ситуаціях – коротке замикання, задимлення, евакуація з приміщення.

Окрім первинного інструктажу, проводяться періодичні тренінги та перевірки знань з охорони праці.

ВИСНОВКИ

В кваліфікаційній роботі була розв’язана практична задача інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов: розроблено вебзастосунок управління перекладами для безперебійної локалізації програмного забезпечення.

Завдання, що були вирішені для досягнення мети:

1) проаналізовано практичну задачу інженерії програмного забезпечення з розробки вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення;

2) проведено аналіз існуючого програмного забезпечення для управління перекладами;

3) виконано постановку задачі на розробку вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення;

4) проаналізовано вимоги до вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення;

5) зроблено вибір архітектури вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення;

6) виконано розробку проєкту вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення;

7) виконано реалізацію вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення.

Тестування та випробування підтвердили працездатність вебзастосунку управління перекладами для безперебійної локалізації програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Esselink B. A Practical Guide to Localization. Amsterdam : John Benjamins Publishing, 2000. 488 p. https://doi.org/10.1075/liwd.4

2. Crowdin: Localization Management Platform. URL: https://crowdin.com (дата звернення: 11.04.2026).

3. POEditor: Translation Management Platform. URL: https://poeditor.com (дата звернення: 11.04.2026).

4. Фрімен Е., Робсон Е., Бейтс Б., Сієрра К. Head First. Патерни проєктування : пер. з англ. Харків : Фабула, 2020. 672 с.

5. Мартін Р. Чиста архітектура. Мистецтво розробки програмного забезпечення : пер. з англ. Київ : Фабула, 2018. 416 с.

6. PHP: Hypertext Preprocessor : офіційний сайт. URL: https://www.php.net (дата звернення: 11.04.2026).

7. Tatroe K., MacIntyre P. Programming PHP. 4th ed. Sebastopol : O'Reilly Media, 2020. 540 p.

8. CakePHP Cookbook. URL: https://book.cakephp.org/5/en/index.html (дата звернення: 11.04.2026).

9. Schwartz B., Zaitsev P., Tkachenko V. High Performance MySQL. 4th ed. Sebastopol : O'Reilly Media, 2021. 826 p.

10. Sneat – Bootstrap 5 HTML Admin Template : офіційний сайт. URL: https://themeselection.com/item/sneat-bootstrap-html-admin-template/ (дата звернення: 11.04.2026).

11. RFC 7231 – Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. URL: https://datatracker.ietf.org/doc/html/rfc7231 (дата звернення: 11.04.2026).

12. Авраменко А.С., Авраменко В.С., Косенюк Г.В. Тестування програмного забезпечення : навчальний посібник. Черкаси : ЧНУ імені Богдана Хмельницького, 2017. 284 с.

13. Про охорону праці : Закон України від 14.10.1992 р. № 2694-XII. Дата оновлення: 04.04.2026. URL: https://zakon.rada.gov.ua/laws/main/2694-12 (дата звернення: 11.04.2026).

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ВЕБЗАСТОСУНКУ УПРАВЛІННЯ ПЕРЕКЛАДАМИ ДЛЯ БЕЗПЕРЕБІЙНОЇ ЛОКАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1 Вступ

1.1 Назва програмного забезпечення

Вебзастосунок управління перекладами для безперебійної локалізації програмного забезпечення (Translation Management Web Application)

1.2 Сфера застосування

Вебзастосунок призначений для використання командами розробників та перекладачів, які займаються локалізацією програмних продуктів. Застосунок забезпечує централізоване управління перекладами інтерфейсів програм із подальшою автоматичною синхронізацією з кодовою базою проєкту.

2 Підстава для розробки програмного забезпечення

Розробка вебзастосунку виконується на підставі завдання на кваліфікаційну роботу, виданого кафедрою ПЗАС НУК імені адмірала Макарова.

3 Призначення розробки

3.1 Функціональне призначення

Функціональним призначенням вебзастосунку є автоматизація процесів локалізації програмного забезпечення шляхом створення централізованого середовища для управління перекладами інтерфейсів програм. Застосунок

оптимізує робочий цикл локалізації: імпорт ресурсних файлів (PO, JSON, CSV), доступ через API, надання інтерактивного інтерфейсу для перекладу та експорту готових локалізацій у потрібних форматах, що виключає ризик пропуску перекладів або пошкодження структури файлів.

3.2 Експлуатаційне призначення

Вебзастосунок призначений для експлуатації командами локалізації програмних продуктів, зокрема розробниками, менеджерами проєктів та перекладачами. Застосунок замінює традиційний підхід до локалізації, що базується на ручному обміні файлами ресурсів між розробниками та перекладачами, на автоматизований процес із централізованим контролем.

4 Технічні вимоги до програми

4.1 Вимоги до функціональних характеристик

4.1.1 Вимоги до переліку виконуваних функцій

Вебзастосунок повинен забезпечувати виконання таких функцій:

створення та управління проєктами локалізації;

- імпорт ресурсних файлів локалізації (формати PO, JSON, CSV);
- інтерактивний інтерфейс для перекладу фраз із підтримкою контексту та коментарів;
- підтримка множинних мов у рамках одного проєкту;
- експорт готових локалізацій у потрібних форматах;
- надання REST API для автоматичної синхронізації перекладів із кодовою базою;
- пошук та фільтрація фраз за критеріями: текст оригіналу або перекладу;
- відображення прогресу перекладу по кожній мові;
- автентифікація користувачів із розмежуванням ролей (адміністратор, користувач);

- управління користувачами адміністратором (додавання, редагування, видалення).

4.1.2 Вимоги до організації вхідних та вихідних даних

Вхідні дані:

- файли ресурсів локалізації у форматах PO (Portable Object), JSON, CSV;

- текстові дані перекладів, що вводяться перекладачами через вебінтерфейс;

- запити до REST API від зовнішніх систем.

Вихідні дані:

- експортовані файли локалізації у форматах PO, JSON та CSV;

- відповіді REST API у форматі JSON.

4.2 Вимоги до надійності

Вебзастосунок повинен забезпечувати:

- коректну обробку помилкових дій користувача з відображенням відповідних повідомлень;

- збереження цілісності даних при одночасній роботі кількох користувачів;

- валідацію вхідних даних на стороні сервера;

- стійкість до некоректних форматів імпортованих файлів із відповідним інформуванням користувача.

4.3 Умови експлуатації

4.3.1 Кліматичні умови експлуатації

Вебзастосунок експлуатується в закритих приміщеннях з нормальними кліматичними умовами для роботи технічного обладнання.

4.3.2 Вимоги до чисельності та кваліфікації користувачів

Кількість користувачів не обмежена. Користувачі повинні мати базові навички роботи з вебзастосунками та вебоглядачами. Для роботи з

функціями API користувач повинен мати базові знання з інтеграції програмного забезпечення.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Вимоги до користувацького обладнання:

- процесор з тактовою частотою не менше 1 ГГц;
- оперативна пам'ять не менше 2 ГБ;
- наявність підключення до мережі Інтернет;
- монітор з роздільною здатністю не менше 1280×720 пікселів.

4.4.2 Вимоги до серверного обладнання:

- процесор з тактовою частотою не менше 4 ГГц;
- оперативна пам'ять не менше 4 ГБ;
- вільне місце на жорсткому диску не менше 1 ГБ;
- наявність стабільного підключення до мережі Інтернет.

4.5 Вимоги до інформаційної та програмної сумісності

Клієнт повинен мати в наявності наступні програмні засоби:

- операційна система: Windows 10/11, macOS 10.15+, Linux або мобільна ОС (Android 10+, iOS 14+);
- вебоглядач: Google Chrome 90+, Mozilla Firefox 90+, Safari 14+, Microsoft Edge 90+ або інший сучасний браузер з підтримкою JavaScript ES6+.

Сервер повинен відповідати наступним вимогам до програмної сумісності:

- операційна система: Linux (Ubuntu 20.04+ або аналог);
- мова програмування: PHP 8.1+;
- система управління базами даних: MySQL 8.0+;
- вебсервер: Apache 2.4+ з модулем mod_rewrite.

4.6 Вимоги до маркування та упаковки

Вимоги до маркування та упаковки відсутні.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання відсутні.

4.8 Спеціальні вимоги

Спеціальні вимоги відсутні.

5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- текст програми;
- керівництво користувача;
- опис програми;
- програму і методику випробувань.

6 Техніко-економічні показники

У кваліфікаційній роботі техніко-економічні показники не розраховуються.

7 Стадії та етапи розробки

Етапи розробки програмного забезпечення наведено в таблиці А.1.

Таблиця А.1 - Етапи розробки програмного забезпечення для безперебійної локалізації програмного забезпечення

Номер	Назва етапу розробки	Терміни виконання
1.	Розробка архітектури ПЗ	20.04.2026
2.	Розробка проекту ПЗ	04.05.2026
3.	Реалізація та тестування ПЗ	11.05.2026

8 Порядок контролю і прийомки

Контроль здійснюється керівником кваліфікаційної роботи на кожному етапі розробки програмного забезпечення. Приймання здійснюється за програмою та методикою випробувань. Випробування проводиться в зазначений термін.

ДОДАТОК Б – ТЕКСТ ВЕБЗАСТОСУНКУ УПРАВЛІННЯ ПЕРЕКЛАДАМИ ДЛЯ БЕЗПЕРЕБІЙНОЇ ЛОКАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Текст файлу ProjectsController.php

```
ProjectsController extends AppController {

 private const ADMIN_ACTIONS = [
    'add', 'edit', 'delete',
    'addLocale', 'editLocale', 'deleteLocale',
    'contributors', 'addContributors',
 ];

 public function initialize(): void {
    parent::initialize();
 }

 public function beforeFilter(EventInterface $event): void {
    parent::beforeFilter($event);
    $action = $this->getRequest()->getParam('action');
    if (in_array($action, static::ADMIN_ACTIONS, true) && !$this->isAdmin())
{
      $this->Flash->error(__('Access denied.'));
      $this->redirect(['action' => 'index']);
    }
 }

 public function index(): void {
    $Query = $this->Projects->find()
      ->contain(['Locales']);

    if (UserRole::Admin !== $this->Auth->user('role')) {
      $userId = (int)$this->Auth->user('id');
      $Query->matching('Users', static fn(Query $q): Query => $q-
>where(['Users.id' => $userId]));
    }

    /** @var \App\Model\Entity\Project[] $Entities */
    $Entities = $Query->all();

    $this->set(compact('Entities'));
 }

 public function view(?string $id = null): void {
    /** @var \App\Model\Entity\Project $Entity */
    $Entity = $this->Projects->findById($id)
      ->contain(['Locales' => static fn(Query $q): Query => $q-
>orderBy(['Locales.name' => 'ASC'])])
      ->firstOrFail();

    $totalTerms = $this->Terms->find()
```

```
        ->where(['project_id' => $Entity->id])
        ->count();

    $translationsByLocale = [];
    if (0 < $totalTerms) {
        $Translations = $this->fetchTable('Translations');
        $rows = $Translations->find()
            ->innerJoinWith('Terms', static fn(Query $q): Query => $q-
>where(['Terms.project_id' => $Entity->id]))
            ->select(['locale_id' => 'Translations.locale_id', 'cnt' =>
$Translations->find()->func()->count('Translations.id')])
            ->groupBy(['Translations.locale_id'])
            ->disableHydration()
            ->all();
        foreach ($rows as $row) {
            $translationsByLocale[(int)$row['locale_id']] = (int)$row['cnt'];
        }
    }

    $this->set(compact('Entity', 'totalTerms', 'translationsByLocale'));
 }

 public function add(): void {
    $Entity = $this->Projects->newEmptyEntity();

    if ($this->projectAddEdit($Entity)) {
        $this->Flash->success(__('Created!'));

        $this->redirect(['action' => 'index']);

        return;
    }
 }

 public function edit(?string $id = null): void {
    /** @var \App\Model\Entity\Project $Entity */
    $Entity = $this->Projects->findById($id)
        ->firstOrFail();

    if ($this->projectAddEdit($Entity)) {
        $this->Flash->success(__('Updated!'));

        $this->redirect(['action' => 'index']);

        return;
    }
 }

 private function projectAddEdit(Project $Entity): bool {
    $validator = [$this->Projects, 'validationDefault'];

    $Request = $this->getRequest();
    if ($Request->is(['post', 'put'])) {
        $fields = ['name', 'description'];
        $this->Projects->patchEntity($Entity, $Request->getData(), ['fields'
=> $fields, 'validate' => $validator]);
```

```
        if ($this->Projects->save($Entity)) {
            return true;
        }

        $this->Flash->error(__('Item not saved'));
    }

    $this->set(compact('Entity', 'validator'));

    return false;
 }

 public function addLocale(?string $id = null): void {
    /** @var \App\Model\Entity\Project $Entity */
    $Entity = $this->Projects->findById($id)
        ->contain(['Locales'])
        ->firstOrFail();

    $attachedLocaleIds = array_map(static fn($Locale) => $Locale->id,
$Entity->locales);

    $Request = $this->getRequest();
    if ($Request->is(['post', 'put'])) {
        $localeId = (int)$Request->getData('locale_id');
        if (0 === $localeId || in_array($localeId, $attachedLocaleIds, true))
{
            $this->Flash->error(__('Please select a language that is not
already added'));
        } else {
            /** @var \App\Model\Entity\Locale $Locale */
            $Locale = $this->Locales->findById($localeId)->firstOrFail();
            $this->Projects->Locales->link($Entity, [$Locale]);
            $this->Flash->success(__('Language added!'));

            $this->redirect(['action' => 'view', $Entity->id]);

            return;
        }
    }

    $LocalesQuery = $this->Locales->find()->orderBy(['name' => 'ASC']);
    if (0 < count($attachedLocaleIds)) {
        $LocalesQuery->where(['id NOT IN' => $attachedLocaleIds]);
    }
    $localeOptions = $LocalesQuery->all()->combine('id', static fn(Locale
$Locale): string => "{$Locale->name} {$Locale->flag}")->toArray();

    $this->set(compact('Entity', 'localeOptions'));
 }

 public function editLocale(?string $id = null, ?string $localeId = null):
void {
    /** @var \App\Model\Entity\Project $Entity */
    $Entity = $this->Projects->findById($id)
        ->contain(['Locales'])
        ->firstOrFail();
```

```
    /** @var \App\Model\Entity\Locale $CurrentLocale */
    $CurrentLocale = $this->Locales->findById($localeId)->firstOrFail();

    $attachedLocaleIds = array_map(static fn(Locale $Locale): int =>
$Locale->id, $Entity->locales);

    $Request = $this->getRequest();
    if ($Request->is(['post', 'put'])) {
      $newLocaleId = (int)$Request->getData('locale_id');
      if (0 === $newLocaleId) {
          $this->Flash->error(__('Please select a language'));
      } elseif ($newLocaleId !== $CurrentLocale->id &&
in_array($newLocaleId, $attachedLocaleIds, true)) {
          $this->Flash->error(__('Please select a language that is not
already added'));
      } else {
          if ($newLocaleId !== $CurrentLocale->id) {
           /** @var \App\Model\Entity\Locale $NewLocale */
           $NewLocale = $this->Locales->findById($newLocaleId)-
>firstOrFail();
           $this->Projects->Locales->unlink($Entity, [$CurrentLocale]);
           $this->Projects->Locales->link($Entity, [$NewLocale]);

           $termIdsQuery = $this->Terms->find()
                 ->select(['id'])
                 ->where(['project_id' => $Entity->id]);
           $this->Translations->updateAll(
                 ['locale_id' => $NewLocale->id],
                 ['locale_id' => $CurrentLocale->id, 'term_id IN' =>
$termIdsQuery],
           );
          }
          $this->Flash->success(__('Language updated!'));

          $this->redirect(['action' => 'view', $Entity->id]);

          return;
      }
    }

    $excludeIds = array_values(array_diff($attachedLocaleIds,
[$CurrentLocale->id]));
    $LocalesQuery = $this->Locales->find()->orderBy(['name' => 'ASC']);
    if (0 < count($excludeIds)) {
      $LocalesQuery->where(['id NOT IN' => $excludeIds]);
    }
    $localeOptions = $LocalesQuery->all()->combine('id', static fn(Locale
$Locale): string => "{$Locale->name} {$Locale->flag}")->toArray();

    $this->set(compact('Entity', 'CurrentLocale', 'localeOptions'));
 }

 public function deleteLocale(?string $id = null, ?string $localeId = null):
void {
    $this->getRequest()->allowMethod(['post', 'delete']);
```

```
    /** @var \App\Model\Entity\Project $Entity */
    $Entity = $this->Projects->findById($id)->firstOrFail();
    /** @var \App\Model\Entity\Locale $Locale */
    $Locale = $this->Locales->findById($localeId)->firstOrFail();

    $this->Projects->Locales->unlink($Entity, [$Locale]);
    $this->Flash->success(__('Language removed'));

    $this->redirect(['action' => 'view', $Entity->id]);
 }

 public function addContributors(?string $id = null): void {
    /** @var \App\Model\Entity\Project $Entity */
    $Entity = $this->Projects->findById($id)
      ->contain(['Users'])
      ->firstOrFail();

    $Request = $this->getRequest();
    if ($Request->is(['post', 'put'])) {
      $selectedIds = array_values(array_filter(array_map('intval',
(array)$Request->getData('user_ids'))));

      $SelectedUsers = 0 < count($selectedIds)
          ? $this->Users->find()->where(['id IN' => $selectedIds, 'role' =>
UserRole::User])->toArray()
          : [];
      $this->Projects->Users->replaceLinks($Entity, $SelectedUsers);

      $this->Flash->success(__('Contributors updated!'));

      $this->redirect(['action' => 'contributors', $Entity->id]);

      return;
    }

    $Users = $this->Users->find()
      ->where(['role' => UserRole::User])
      ->orderBy(['first_name' => 'ASC', 'last_name' => 'ASC'])
      ->toArray();
    $attachedUserIds = array_map(static fn($User) => $User->id, $Entity-
>users);

    $this->set(compact('Entity', 'Users', 'attachedUserIds'));
 }

 public function contributors(?string $id = null): void {
    /** @var \App\Model\Entity\Project $Entity */
    $Entity = $this->Projects->findById($id)
      ->contain(['Users' => static fn(Query $q): Query => $q-
>orderBy(['Users.first_name' => 'ASC', 'Users.last_name' => 'ASC'])])
      ->firstOrFail();

    $isAdmin = UserRole::Admin === $this->Auth->user('role');

    $Request = $this->getRequest();
```

```
    if ($Request->is(['post', 'put'])) {
      if (!$isAdmin) {
          $this->Flash->error(__('Access denied.'));
          $this->redirect(['action' => 'contributors', $Entity->id]);

          return;
      }

      $selectedIds = array_values(array_filter(array_map('intval',
(array)$Request->getData('user_ids'))));
      if (0 < count($selectedIds)) {
          $ToUnlink = array_filter($Entity->users, static fn($User) =>
in_array($User->id, $selectedIds, true));
          if (0 < count($ToUnlink)) {
           $this->Projects->Users->unlink($Entity, array_values($ToUnlink));
           $this->Flash->success(__('Contributors removed'));
          }
      }

      $this->redirect(['action' => 'contributors', $Entity->id]);

      return;
    }

    $this->set(compact('Entity', 'isAdmin'));
 }

 public function delete(string $id = null): void {
    $this->getRequest()->allowMethod('delete');
    /** @var \App\Model\Entity\Project $Entity */
    $Entity = $this->Projects->findById($id)->firstOrFail();
    $this->Projects->deleteOrFail($Entity);
    $this->Flash->success(__('Item deleted'));

    $this->redirect(['action' => 'index']);
 }
}
```

Текст файлу TermsController.php

```
class TermsController extends AppController {

 public function initialize(): void {
    parent::initialize();
 }

 public function beforeFilter(EventInterface $event): void {
    parent::beforeFilter($event);

    if (!$this->isAdmin()) {
      $Request = $this->getRequest();
      if ($Request->is('ajax')) {
          throw new \Cake\Http\Exception\ForbiddenException('Admins only');
      }
      $this->Flash->error(__('Access denied.'));
```

```
            $this->redirect(['controller' => 'Projects', 'action' => 'index']);
        }
    }

    public function index(?string $projectId): void {
        $Request = $this->getRequest();
        /** @var \App\Model\Entity\Project $Project */
        $Project = $this->Projects->findById($projectId)
            ->contain(['Locales'])
            ->firstOrFail();

        $Locales = array_column($Project->locales, null, 'id');

        $Query = $this->Terms->find()
            ->where(['Terms.project_id' => $Project->id])
            ->contain(['Translations'])
            ->orderBy(['Terms.id' => 'ASC']);

        $q = trim((string)$Request->getQuery('q'));
        if ('' !== $q) {
            $pattern = '%' . mb_strtolower($q) . '%';
            $Query->where(function ($exp, $query) use ($pattern) {
                $content = $query->func()->lower(['Terms.content' =>
'identifier']);
                $context = $query->func()->lower(['Terms.context' =>
'identifier']);
                $comment = $query->func()->lower(['Terms.comment' =>
'identifier']);

                return $exp->or(static fn($or) => $or
                    ->like($content, $pattern)
                    ->like($context, $pattern)
                    ->like($comment, $pattern));
            });
        }

        /** @var \App\Model\Entity\Term[] $Entities */
        $Entities = $this->paginate($Query);

        $csrfToken = (string)$Request->getAttribute('csrfToken');
        $hasTranslationEngine = ''
            !== (string)Configure::read('Integrations.Deepl.apiKey')
            || '' !== (string)Configure::read('Integrations.Google.apiKey');

        $this->set(compact('Project', 'Entities', 'Locales', 'csrfToken', 'q',
'hasTranslationEngine'));
    }

    public function saveTranslation(): Response {
        $Request = $this->getRequest();
        $Request->allowMethod(['post']);

        $data = $Request->getData();
        $termId = (int)($data['term_id'] ?? 0);
        $localeId = (int)($data['locale_id'] ?? 0);
        $content = (string)($data['content'] ?? '');
```

```
    if (0 === $termId || 0 === $localeId) {
      throw new BadRequestException('term_id and locale_id are required');
    }

    $Translation = $this->Translations->find()
      ->where(['term_id' => $termId, 'locale_id' => $localeId])
      ->first();

    if ('' === trim($content)) {
      if (null !== $Translation) {
         $this->Translations->deleteOrFail($Translation);
      }

      return $this->jsonResponse([‘success’ => true, 'deleted' => true]);
    }

    if (null === $Translation) {
      $Translation = $this->Translations->newEmptyEntity();
      $Translation->term_id = $termId;
      $Translation->locale_id = $localeId;
    }
    $Translation->content = $content;

    $saved = (bool)$this->Translations->save($Translation);

    return $this->jsonResponse([‘success’ => $saved, 'id' => $Translation-
>id]);
 }

 public function deleteMany(): Response {
    $Request = $this->getRequest();
    $Request->allowMethod(['post']);

    $ids = (array)$Request->getData('term_ids');
    $ids = array_values(array_filter(array_map('intval', $ids)));
    if (0 === count($ids)) {
      throw new BadRequestException('No term ids provided');
    }

    $Entities = $this->Terms->find()->where(['id IN' => $ids])->all();
    $this->Terms->deleteManyOrFail($Entities);

    return $this->jsonResponse([‘success’ => true, 'deleted' =>
count($Entities)]);
 }

 public function saveTerm(?string $id = null): Response {
    $Request = $this->getRequest();
    $Request->allowMethod(['post']);

    /** @var \App\Model\Entity\Term $Term */
    $Term = $this->Terms->findById($id)->firstOrFail();

    $data = $Request->getData();
    $allowedFields = ['content', 'context', 'comment'];
```

```
    $payload = array_intersect_key($data, array_flip($allowedFields));
    if (0 === count($payload)) {
      throw new BadRequestException('No editable fields provided');
    }

    $this->Terms->patchEntity($Term, $payload, ['fields' => $allowedFields,
'validate' => [$this->Terms, 'validationDefault']]);
    $saved = (bool)$this->Terms->save($Term);

    return $this->jsonResponse([‘success’ => $saved, 'id' => $Term->id]);
 }

 public function add(?string $projectId): void {
    $Project = $this->Projects->findById($projectId)->firstOrFail();
    $Entity = $this->Terms->newEmptyEntity();
    $Entity->project = $Project;

    if ($this->termAddEdit($Entity)) {
      $this->Flash->success(__('Created!'));

      $this->redirect(['action' => 'index', $Project->id]);

      return;
    }
 }

 private function termAddEdit(Term $Entity): bool {
    $validator = [$this->Terms, 'validationDefault'];
    $Project = $Entity->project;

    $Request = $this->getRequest();
    if ($Request->is(['post', 'put'])) {
      $fields = ['content', 'context', 'comment'];
      $this->Terms->patchEntity($Entity, $Request->getData(), ['fields' =>
$fields, 'validate' => $validator]);
      if ($this->Terms->save($Entity)) {
         return true;
      }

      $this->Flash->error(__('Item not saved'));
    }

    $this->set(compact('Project', 'Entity', 'validator'));

    return false;
 }

 public function translate(?string $id = null): void {
    /** @var \App\Model\Entity\Term $Term */
    $Term = $this->Terms->findById($id)
      ->contain(['Projects' => ['Locales'], 'Translations'])
      ->firstOrFail();
    $Project = $Term->project;
    $translatedLocaleIds = array_map(static fn($T) => (int)$T->locale_id,
$Term->translations);
```

```
    $engines = [];
    if ('' !== (string)Configure::read('Integrations.Deepl.apiKey')) {
      $engines['deepl'] = __('DeepL');
    }
    if ('' !== (string)Configure::read('Integrations.Google.apiKey')) {
      $engines['google'] = __('Google Translate');
    }

    $Request = $this->getRequest();
    if ($Request->is(['post', 'put'])) {
      $engine = (string)$Request->getData('engine');
      $targetIds = array_values(array_filter(array_map('intval',
(array)$Request->getData('locale_ids'))));

      if (!array_key_exists($engine, $engines)) {
          $this->Flash->error(__('Please select a translation engine.'));
      } elseif (0 === count($targetIds)) {
          $this->Flash->error(__('Please select at least one language.'));
      } else {
          $created = $updated = $skipped = 0;
          foreach ($Project->locales as $Locale) {
           if (!in_array($Locale->id, $targetIds, true)) {
                continue;
           }
           $text = $this->machineTranslate($engine, $Term->content,
$Locale);
           if ('' === $text) {
                $skipped++;
                continue;
           }
           /** @var \App\Model\Entity\Translation|null $Translation */
           $Translation = $this->Translations->find()
                ->where(['term_id' => $Term->id, 'locale_id' => $Locale-
>id])
                ->first();
           if (null === $Translation) {
                $Translation = $this->Translations->newEmptyEntity();
                $Translation->term_id = $Term->id;
                $Translation->locale_id = $Locale->id;
                $created++;
           } else {
                $updated++;
           }

           $Translation->content = $text;
           $this->Translations->saveOrFail($Translation);
          }

          $this->Flash->success(__('Translated via {0}: {1} created, {2}
updated, {3} skipped', $engines[$engine], $created, $updated, $skipped));
          $redirectUrl = $this->Auth->getLoginRedirect() ?: ['action' =>
'index', $Project->id];
          $this->redirect($redirectUrl);

          return;
      }
```

```
    }

    $this->set(compact('Term', 'Project', 'engines',
'translatedLocaleIds'));
 }

 public function delete(string $id = null): void {
    $this->getRequest()->allowMethod('delete');
    /** @var \App\Model\Entity\Term $Entity */
    $Entity = $this->Terms->findById($id)
      ->contain(['Projects'])
      ->firstOrFail();
    $Project = $Entity->project;

    $this->Terms->deleteOrFail($Entity);
    $this->Flash->success(__('Item deleted'));

    $this->redirect(['action' => 'index', $Project->id]);
 }
}
```

Текст файлу TranslationsController.php

```
class TranslationsController extends AppController {

 public function initialize(): void {
    parent::initialize();
 }

 public function index(?string $projectId = null, ?string $localeCode =
null): void {
    $Request = $this->getRequest();
    /** @var \App\Model\Entity\Project $Project */
    $Project = $this->Projects->findById($projectId)
      ->contain(['Locales'])
      ->firstOrFail();

    /** @var \App\Model\Entity\Locale $Locale */
    $Locale = $this->Locales->find()->where(['code' => $localeCode])-
>firstOrFail();

    $Query = $this->Terms->find()
      ->where(['Terms.project_id' => $Project->id])
      ->contain(['Translations' => static fn(Query $Query): Query => $Query-
>where(['Translations.locale_id' => $Locale->id])])
      ->orderBy(['Terms.id' => 'ASC']);

    $q = trim((string)$Request->getQuery('q'));
    if ('' !== $q) {
      $Query
          ->leftJoinWith('Translations', static fn(Query $sub): Query =>
$sub->where(['Translations.locale_id' => $Locale->id]))
          ->where([
           'OR' => [
                'Terms.content LIKE' => "%{$q}%",
```

```
                'Terms.context LIKE' => "%{$q}%",
                'Translations.content LIKE' => "%{$q}%",
            ],
          ])
          ->groupBy(['Terms.id']);
    }

    /** @var \App\Model\Entity\Term[] $Entities */
    $Entities = $this->paginate($Query);

    $csrfToken = (string)$Request->getAttribute('csrfToken');

    $this->set(compact('Project', 'Locale', 'Entities', 'csrfToken', 'q'));
 }

 public function saveTranslation(): Response {
    $Request = $this->getRequest();
    $Request->allowMethod(['post']);

    $data = $Request->getData();
    $termId = (int)($data['term_id'] ?? 0);
    $localeId = (int)($data['locale_id'] ?? 0);
    $content = (string)($data['content'] ?? '');

    if (0 === $termId || 0 === $localeId) {
      throw new BadRequestException('term_id and locale_id are required');
    }

    $Translation = $this->Translations->find()
      ->where(['term_id' => $termId, 'locale_id' => $localeId])
      ->first();

    if ('' === trim($content)) {
      if (null !== $Translation) {
          $this->Translations->deleteOrFail($Translation);
      }

      return $this->jsonResponse([‘success’ => true, 'deleted' => true]);
    }

    if (null === $Translation) {
      $Translation = $this->Translations->newEmptyEntity();
      $Translation->term_id = $termId;
      $Translation->locale_id = $localeId;
    }
    $Translation->content = $content;

    $saved = (bool)$this->Translations->save($Translation);

    return $this->jsonResponse([‘success’ => $saved, 'id' => $Translation-
>id]);
 }

 public function deleteMany(): Response {
    $Request = $this->getRequest();
    $Request->allowMethod(['post']);
```

```
    $ids = (array)$Request->getData('translation_ids');
    $ids = array_values(array_filter(array_map('intval', $ids)));
    if (0 === count($ids)) {
      throw new BadRequestException('No translation ids provided');
    }

    $deleted = $this->Translations->deleteAll(['id IN' => $ids]);

    return $this->jsonResponse([‘success’ => true, 'deleted' => $deleted]);
 }

 public function export(?string $projectId = null, ?string $localeCode =
null): ?Response {
    /** @var \App\Model\Entity\Project $Project */
    $Project = $this->Projects->findById($projectId)->firstOrFail();
    /** @var \App\Model\Entity\Locale $Locale */
    $Locale = $this->Locales->find()->where(['code' => $localeCode])-
>firstOrFail();

    $Request = $this->getRequest();
    if ($Request->is(['post', 'put'])) {
      $format = (string)$Request->getData('format');
      $onlyTranslated = (bool)$Request->getData('only_translated');

      if (!in_array($format, ['po', 'json', 'key_value_json', 'xliff',
'csv'], true)) {
          $this->Flash->error(__('Unsupported format.'));

          return null;
      }

      $Terms = $this->Terms->find()
          ->where(['Terms.project_id' => $Project->id])
          ->contain(['Translations' => static fn(Query $q): Query => $q-
>where(['Translations.locale_id' => $Locale->id])])
          ->orderBy(['Terms.id' => 'ASC'])
          ->all();

      if ($onlyTranslated) {
          $Terms = $Terms->filter(static fn(\App\Model\Entity\Term $Term):
bool => isset($Term->translations[0]) && '' !== trim((string)$Term-
>translations[0]->content));
      }

      [$content, $ext, $contentType] = $this->renderExport($format, $Terms,
$Locale, $Project);

      $filename = sprintf('%s_%s.%s', $this->slugify($Project->name),
$Locale->code, $ext);

      return $this->getResponse()
          ->withType($contentType)
          ->withHeader('Content-Disposition', 'attachment; filename="' .
$filename . '"')
          ->withStringBody($content);
```

```
    }

    $this->set(compact('Project', 'Locale'));

    return null;
 }

 /**
  * @param iterable<\App\Model\Entity\Term> $Terms
  *
  * @return array{0: string, 1: string, 2: string}
  */
 private function renderExport(string $type, iterable $Terms,
\App\Model\Entity\Locale $Locale, \App\Model\Entity\Project $Project): array
{
    switch ($type) {
       case 'json':
           $out = [];
           foreach ($Terms as $Term) {
            $Translation = $Term->translations[0] ?? null;
            $out[] = [
                  'term' => $Term->content,
                  'context' => $Term->context,
                  'comment' => $Term->comment,
                  'translation' => ['content' => null !== $Translation ?
$Translation->content : ''],
            ];
           }

           return [json_encode($out, JSON_UNESCAPED_UNICODE |
JSON_PRETTY_PRINT), 'json', 'application/json'];

       case 'key_value_json':
           $out = [];
           foreach ($Terms as $Term) {
            $Translation = $Term->translations[0] ?? null;
            $out[$Term->content] = null !== $Translation ? $Translation-
>content : '';
           }

           return [json_encode($out, JSON_UNESCAPED_UNICODE |
JSON_PRETTY_PRINT), 'json', 'application/json'];

       case 'csv':
           $lines = [];
           $lines[] = $this->csvRow(['term', 'context', 'comment',
'translation']);
           foreach ($Terms as $Term) {
            $Translation = $Term->translations[0] ?? null;
            $lines[] = $this->csvRow([
                  $Term->content,
                  $Term->context,
                  $Term->comment,
                  null !== $Translation ? (string)$Translation->content : '',
            ]);
           }
```

```
            return [implode("\n", $lines), 'csv', 'text/csv'];

        case 'xliff':
            $xml = new \DOMDocument('1.0', 'UTF-8');
            $xml->formatOutput = true;
            $xliff = $xml->createElement('xliff');
            $xliff->setAttribute('version', '1.2');
            $xliff->setAttribute('xmlns',
'urn:oasis:names:tc:xliff:document:1.2');
            $xml->appendChild($xliff);

            $file = $xml->createElement('file');
            $file->setAttribute('original', $Project->name);
            $file->setAttribute('source-language', 'en');
            $file->setAttribute('target-language', $Locale->code);
            $file->setAttribute('datatype', 'plaintext');
            $xliff->appendChild($file);

            $body = $xml->createElement('body');
            $file->appendChild($body);

            $index = 0;
            foreach ($Terms as $Term) {
             $Translation = $Term->translations[0] ?? null;
             $index++;
             $unit = $xml->createElement('trans-unit');
             $unit->setAttribute('id', (string)$Term->id);
             if ('' !== (string)$Term->context) {
                   $unit->setAttribute('resname', (string)$Term->context);
             }
             $source = $xml->createElement('source');
             $source->appendChild($xml->createTextNode((string)$Term-
>content));
             $unit->appendChild($source);
             $target = $xml->createElement('target');
             $target->appendChild($xml->createTextNode(null !== $Translation ?
(string)$Translation->content : ''));
             $unit->appendChild($target);
             if ('' !== (string)$Term->comment) {
                   $note = $xml->createElement('note');
                   $note->appendChild($xml->createTextNode((string)$Term-
>comment));
                   $unit->appendChild($note);
             }
             $body->appendChild($unit);
            }

            return [(string)$xml->saveXML(), 'xliff', 'application/x-
xliff+xml'];

        case 'po':
        default:
            $lines = [];
            $lines[] = 'msgid ""';
            $lines[] = 'msgstr ""';
```

```
            $lines[] = '"MIME-Version: 1.0\n"';
            $lines[] = '"Content-Type: text/plain; charset=UTF-8\n"';
            $lines[] = '"Content-Transfer-Encoding: 8bit\n"';
            $lines[] = '"Project-Id-Version: ' . $this->poEscape($Project-
>name) . '\n"';
            $lines[] = '"Language: ' . $Locale->code . '\n"';
            $lines[] = '';
            foreach ($Terms as $Term) {
             $Translation = $Term->translations[0] ?? null;
             if ('' !== (string)$Term->comment) {
                   $lines[] = '#: ' . str_replace("\n", "\n# ", $Term-
>comment);
             }
             if ('' !== (string)$Term->context) {
                   $lines = array_merge($lines, $this-
>poFormatString('msgctxt', $Term->context));
             }
             $lines = array_merge($lines, $this->poFormatString('msgid',
$Term->content));
             $lines = array_merge($lines, $this->poFormatString('msgstr', null
!== $Translation ? $Translation->content : ''));
             $lines[] = '';
            }

           return [implode("\n", $lines), 'po', 'text/x-gettext-
translation'];
     }
 }

 private function poEscape(string $value): string {
     return str_replace(["\\", "\"", "\n"], ["\\\\", "\\\"", "\\n"], $value);
 }

 /**
  * @return array<int, string>
  */
 private function poFormatString(string $key, string $value): array {
     $escaped = $this->poEscape($value);
     $segments = [];
     $current = '';
     foreach (preg_split('/(\\\\n)/', $escaped, -1, PREG_SPLIT_DELIM_CAPTURE)
as $part) {
       $current .= $part;
       if ('\\n' === $part) {
           $segments[] = $current;
           $current = '';
       }
     }
     if ('' !== $current) {
       $segments[] = $current;
     } elseif (str_ends_with($escaped, '\\n')) {
       $segments[] = '';
     }
     if (1 >= count($segments)) {
       return [$key . ' "' . $escaped . '"'];
     }
```

```
    $lines = [];
    foreach ($segments as $i => $segment) {
      $lines[] = (0 === $i ? $key . ' ' : '') . '"' . $segment . '"';
    }

    return $lines;
 }

 private function csvRow(array $fields): string {
    return implode(',', array_map(static fn(string $f): string => '"' .
str_replace('"', '""', $f) . '"', $fields));
 }

 private function slugify(string $value): string {
    $value = preg_replace('/[^\pL\d]+/u', '_', $value) ?? '';
    $value = trim($value, '_');

    return '' !== $value ? strtolower($value) : 'export';
 }

 public function import(?string $projectId = null, ?string $localeCode =
null): void {
    /** @var \App\Model\Entity\Project $Project */
    $Project = $this->Projects->findById($projectId)->firstOrFail();
    /** @var \App\Model\Entity\Locale $Locale */
    $Locale = $this->Locales->find()->where(['code' => $localeCode])-
>firstOrFail();

    $Request = $this->getRequest();
    if ($Request->is(['post', 'put'])) {
      /** @var \Psr\Http\Message\UploadedFileInterface|null $upload */
      $upload = $Request->getUploadedFile('file');
      $format = (string)$Request->getData('format');
      $createMissing = $this->isAdmin() && (bool)$Request-
>getData('create_missing');
      $overwrite = (bool)$Request->getData('overwrite');

      if (null === $upload || UPLOAD_ERR_OK !== $upload->getError()) {
          $this->Flash->error(__('Please select a file to upload.'));
      } else {
          $contents = (string)$upload->getStream()->getContents();
          $entries = $this->parseImport($contents, $format);

          $created = $updated = $skipped = 0;
          foreach ($entries as $entry) {
           $termText = (string)($entry['term'] ?? '');
           $translationText = (string)($entry['translation'] ?? '');
           $context = (string)($entry['context'] ?? '');
           $comment = (string)($entry['comment'] ?? '');
           if ('' === trim($termText)) {
                 continue;
           }

           /** @var \App\Model\Entity\Term|null $Term */
```

```
            $Term = $this->Terms->find()->where(['project_id' => $Project-
>id, 'content' => $termText])->first();
            if (null === $Term) {
                  if (!$createMissing) {
                        $skipped++;
                        continue;
                  }
                  $Term = $this->Terms->newEmptyEntity();
                  $Term->project_id = $Project->id;
                  $Term->content = $termText;
                  $Term->context = $context;
                  $Term->comment = $comment;
                  $this->Terms->saveOrFail($Term);
            } elseif ($overwrite) {
                  $dirty = false;
                  if ('' !== $context && $context !== (string)$Term->context)
{
                        $Term->context = $context;
                        $dirty = true;
                  }
                  if ('' !== $comment && $comment !== (string)$Term->comment)
{
                        $Term->comment = $comment;
                        $dirty = true;
                  }
                  if ($dirty) {
                        $this->Terms->saveOrFail($Term);
                  }
            }

            if ('' === trim($translationText)) {
                  continue;
            }

            /** @var \App\Model\Entity\Translation|null $Translation */
            $Translation = $this->Translations->find()->where(['term_id' =>
$Term->id, 'locale_id' => $Locale->id])->first();
            if (null === $Translation) {
                  $Translation = $this->Translations->newEmptyEntity();
                  $Translation->term_id = $Term->id;
                  $Translation->locale_id = $Locale->id;
                  $Translation->content = $translationText;
                  $this->Translations->saveOrFail($Translation);
                  $created++;
            } elseif ($overwrite) {
                  $Translation->content = $translationText;
                  $this->Translations->saveOrFail($Translation);
                  $updated++;
            } else {
                  $skipped++;
            }
          }

          $this->Flash->success(__('Imported: {0} created, {1} updated, {2}
skipped', $created, $updated, $skipped));
```

```
            $this->redirect(['action' => 'index', $Project->id, $Locale-
>code]);

            return;
        }
    }

    $this->set(compact('Project', 'Locale'));
 }

 /**
  * @return array<int, array{term: string, context: string, comment: string,
translation: string}>
  */
 private function parseImport(string $contents, string $format): array {
    switch ($format) {
       case 'json':
           $decoded = json_decode($contents, true);
           if (!is_array($decoded)) {
            return [];
           }
           $entries = [];
           foreach ($decoded as $key => $value) {
            if (is_string($value)) {
                  $entries[] = ['term' => (string)$key, 'context' => '',
'comment' => '', 'translation' => $value];
            } elseif (is_array($value) && array_key_exists('term', $value)) {
                  $entries[] = [
                        'term' => (string)$value['term'],
                        'context' => (string)($value['context'] ?? ''),
                        'comment' => (string)($value['comment'] ?? ''),
                        'translation' =>
(string)($value['translation']['content'] ?? ($value['translation'] ?? '')),
                  ];
            }
           }

           return $entries;

       case 'po':
           $tmp = tempnam(sys_get_temp_dir(), 'po_import_');
           file_put_contents($tmp, $contents);
           try {
            $parsed = (new PoFileParser())->parse($tmp);
           } finally {
            @unlink($tmp);
           }

           $entries = [];
           foreach ($parsed as $msgid => $data) {
            if ('' === $msgid || !is_array($data)) {
                  continue;
            }
            $contexts = $data['_context'] ?? ['' => ''];
            foreach ($contexts as $context => $translation) {
                  if (is_array($translation)) {
```

```
                        $translation = $translation[0] ?? '';
                    }
                    $entries[] = [
                        'term' => (string)$msgid,
                        'context' => (string)$context,
                        'comment' => '',
                        'translation' => (string)$translation,
                    ];
                }
            }

            return $entries;

        case 'csv':
            $entries = [];
            $rows = preg_split('/\r\n|\r|\n/', $contents);
            foreach ($rows as $i => $row) {
                if (0 === $i || '' === trim($row)) {
                    continue;
                }
                $cols = str_getcsv($row);
                if (2 > count($cols)) {
                    continue;
                }
                $term = (string)$cols[0];
                if ('' === $term) {
                    continue;
                }
                $translation = (string)$cols[count($cols) - 1];
                $context = (string)($cols[1] ?? '');
                $comment = 4 <= count($cols) ? (string)$cols[2] : '';
                $entries[] = ['term' => $term, 'context' => $context, 'comment'
=> $comment, 'translation' => $translation];
            }

            return $entries;
    }

    return [];
 }
}
```

Текст файлу ApiController.php

```
class ApiController extends AppController {

 public function initialize(): void {
    parent::initialize();
 }

 public function beforeFilter(EventInterface $event): void {
    parent::beforeFilter($event);
    $this->Auth->allowUnauthenticated(['projectsExport']);
 }
```

```
 public function projectsExport(): Response {
    $Request = $this->getRequest();
    $Request->allowMethod(['post']);

    $User = $this->authenticateApiUser();
    if (null === $User) {
      return $this->apiResponse('fail', 403, 'Invalid api_token');
    }

    $projectId = (int)$Request->getData('id');
    $languageCode = (string)$Request->getData('language');
    $type = (string)$Request->getData('type');

    if (0 === $projectId) {
      return $this->apiResponse('fail', 400, 'Missing project id');
    }
    if ('' === $languageCode) {
      return $this->apiResponse('fail', 400, 'Missing language');
    }
    if (!in_array($type, ['po', 'json', 'key_value_json', 'csv'], true)) {
      return $this->apiResponse('fail', 400, 'Unsupported type');
    }

    /** @var \App\Model\Entity\Project|null $Project */
    $Project = $this->Projects->findById($projectId)->first();
    if (null === $Project) {
      return $this->apiResponse('fail', 404, 'Project not found');
    }

    /** @var \App\Model\Entity\Locale|null $Locale */
    $Locale = $this->Locales->find()->where(['code' => $languageCode])-
>first();
    if (null === $Locale) {
      return $this->apiResponse('fail', 404, 'Language not found');
    }

    $Terms = $this->Terms->find()
      ->where(['Terms.project_id' => $Project->id])
      ->contain(['Translations' => static fn(Query $q): Query => $q-
>where(['Translations.locale_id' => $Locale->id])])
      ->orderBy(['Terms.id' => 'ASC'])
      ->all();

    [$content, $ext, $contentType] = $this->renderExport($type, $Terms,
$Locale, $Project);

    return $this->apiResponse('success', 200, 'OK', [
      'filetype' => $ext,
      'content_type' => $contentType,
      'contents' => $content,
    ]);
 }

 private function authenticateApiUser(): ?\App\Model\Entity\User {
    $token = (string)$this->getRequest()->getData('api_token');
```

```
    if ('' === $token) {
      return null;
    }

    /** @var iterable<\App\Model\Entity\User> $Users */
    $Users = $this->Users->find()->where(['api_key IS NOT' => null])->all();
    foreach ($Users as $User) {
      $decoded = json_decode((string)$User->api_key, true);
      if (!is_array($decoded)) {
          continue;
      }
      if (($decoded['is_enabled'] ?? false) && ($decoded['value'] ?? '') ===
$token) {
          return $User;
      }
    }

    return null;
 }

 /**
  * @param iterable<\App\Model\Entity\Term> $Terms
  *
  * @return array{0: string, 1: string, 2: string}
  */
 private function renderExport(string $type, iterable $Terms,
\App\Model\Entity\Locale $Locale, \App\Model\Entity\Project $Project): array
{
    switch ($type) {
      case 'json':
          $out = [];
          foreach ($Terms as $Term) {
           $Translation = $Term->translations[0] ?? null;
           $out[] = [
                 'term' => $Term->content,
                 'context' => $Term->context,
                 'comment' => $Term->comment,
                 'translation' => ['content' => null !== $Translation ?
$Translation->content : ''],
            ];
          }
          return [json_encode($out,
JSON_UNESCAPED_UNICODE|JSON_PRETTY_PRINT), 'json', 'application/json'];

      case 'key_value_json':
          $out = [];
          foreach ($Terms as $Term) {
           $Translation = $Term->translations[0] ?? null;
           $out[$Term->content] = null !== $Translation ? $Translation-
>content : '';
          }
          return [json_encode($out, JSON_UNESCAPED_UNICODE |
JSON_PRETTY_PRINT), 'json', 'application/json'];

      case 'csv':
          $lines = [];
```

```
            $lines[] = $this->csvRow(['term', 'context', 'comment',
'translation']);
            foreach ($Terms as $Term) {
             $Translation = $Term->translations[0] ?? null;
             $lines[] = $this->csvRow([
                   $Term->content,
                   $Term->context,
                   $Term->comment,
                   null !== $Translation ? (string)$Translation->content : '',
             ]);
            }
            return [implode("\n", $lines), 'csv', 'text/csv'];

        case 'po':
        default:
            $lines = [];
            $lines[] = 'msgid ""';
            $lines[] = 'msgstr ""';
            $lines[] = '"MIME-Version: 1.0\n"';
            $lines[] = '"Content-Type: text/plain; charset=UTF-8\n"';
            $lines[] = '"Content-Transfer-Encoding: 8bit\n"';
            $lines[] = '"Project-Id-Version: ' . $this->poEscape($Project-
>name) . '\n"';
            $lines[] = '"Language: ' . $Locale->code . '\n"';
            $lines[] = '';
            foreach ($Terms as $Term) {
             $Translation = $Term->translations[0] ?? null;
             if ('' !== $Term->comment) {
                   $lines[] = '#: ' . str_replace("\n", "\n# ", $Term-
>comment);
             }
             if ('' !== $Term->context) {
                   $lines = array_merge($lines, $this-
>poFormatString('msgctxt', $Term->context));
             }
             $lines = array_merge($lines, $this->poFormatString('msgid',
$Term->content));
             $lines = array_merge($lines, $this->poFormatString('msgstr', null
!== $Translation ? $Translation->content : ''));
             $lines[] = '';
            }

            return [implode("\n", $lines), 'po', 'text/x-gettext-
translation'];
        }
    }

    private function poEscape(string $value): string {
        return str_replace(["\\", "\"", "\n"], ["\\\\", "\\\"", "\\n"], $value);
    }

    /**
     * Format a PO key/value pair, splitting the value across multiple
     * quoted lines on each `\n` (literal escape) so long messages remain
     * readable. Returns an array of output lines.
     *
```

```
  * @return array<int, string>
  */
 private function poFormatString(string $key, string $value): array {
     $escaped = $this->poEscape($value);
     $segments = [];
     $current = '';
     foreach (preg_split('/(\\\\n)/', $escaped, -1, PREG_SPLIT_DELIM_CAPTURE)
as $part) {
        $current .= $part;
        if ('\\n' === $part) {
            $segments[] = $current;
            $current = '';
        }
     }
     if ('' !== $current) {
        $segments[] = $current;
     } elseif (str_ends_with($escaped, '\\n')) {
        $segments[] = '';
     }
     if (1 >= count($segments)) {
        return [$key . ' "' . $escaped . '"'];
     }

     $lines = [];
     foreach ($segments as $i => $segment) {
        $lines[] = (0 === $i ? $key . ' ' : '') . '"' . $segment . '"';
     }

     return $lines;
 }

 private function csvRow(array $fields): string {
     return implode(',', array_map(static fn(string $f): string => '"' .
str_replace('"', '""', $f) . '"', $fields));
 }

 private function apiResponse(string $status, int $code, string $message,
array $result = []): Response {
     $body = [
        'response' => [
            'status' => $status,
            'code' => $code,
            'message' => $message,
        ],
        'result' => $result,
     ];

     return $this->getResponse()
        ->withType('application/json')
        ->withStringBody((string)json_encode($body, JSON_UNESCAPED_UNICODE));
 }
}
```

Текст файлу Project.php

```
namespace App\Model\Entity;

/**
 * @property int $id
 * @property string $name
 * @property string $description
 * @property \Cake\I18n\DateTime $created
 * @property \Cake\I18n\DateTime $modified
 *
 * @property \App\Model\Entity\Term[] $terms
 * @property \App\Model\Entity\Locale[] $locales
 * @property \App\Model\Entity\User[] $users
 */
class Project extends Entity {

}
```

Текст файлу Locale.php

```
namespace App\Model\Entity;

/**
 * @property int $id
 * @property string $code
 * @property string $name
 * @property string $flag
 * @property string $deepl_code
 * @property string $gt_code
 *
 * @property \App\Model\Entity\Translation[] $translations
 * @property \App\Model\Entity\Project[] $projects
 */
class Locale extends Entity {

}
```

Текст файлу Term.php

```
namespace App\Model\Entity;

/**
 * @property int $id
 * @property int $project_id
 * @property string $content
 * @property string $context
 * @property string $comment
 * @property \Cake\I18n\DateTime $created
 * @property \Cake\I18n\DateTime $modified
 *
 * @property \App\Model\Entity\Project $project
```

```
 * @property \App\Model\Entity\Translation[] $translations
 */
class Term extends Entity {

}
```

Текст файлу Translation.php

```
namespace App\Model\Entity;

/**
 * @property int $id
 * @property int $term_id
 * @property int $locale_id
 * @property string $content
 * @property \Cake\I18n\DateTime $created
 * @property \Cake\I18n\DateTime $modified
 *
 * @property \App\Model\Entity\Term $term
 * @property \App\Model\Entity\Locale $locale
 */
class Translation extends Entity {

}
```

Текст файлу Locale.php

```
namespace App\Model\Entity;

/**
 * @property int $id
 * @property string $code
 * @property string $name
 * @property string $flag
 * @property string $deepl_code
 * @property string $gt_code
 *
 * @property \App\Model\Entity\Translation[] $translations
 * @property \App\Model\Entity\Project[] $projects
 */
class Locale extends Entity {

}
```

ДОДАТОК В – ІНСТРУКЦІЯ КОРИСТУВАЧА ВЕБЗАСТОСУНКУ УПРАВЛІННЯ ПЕРЕКЛАДАМИ ДЛЯ БЕЗПЕРЕБІЙНОЇ ЛОКАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Початок роботи

Для початку роботи відкрийте вебоглядач і перейдіть за URL-адресою вебзастосунку, наданою адміністратором. На сторінці входу введіть свою електронну пошту і пароль та натисніть кнопку «Login» (рисунок В.1).

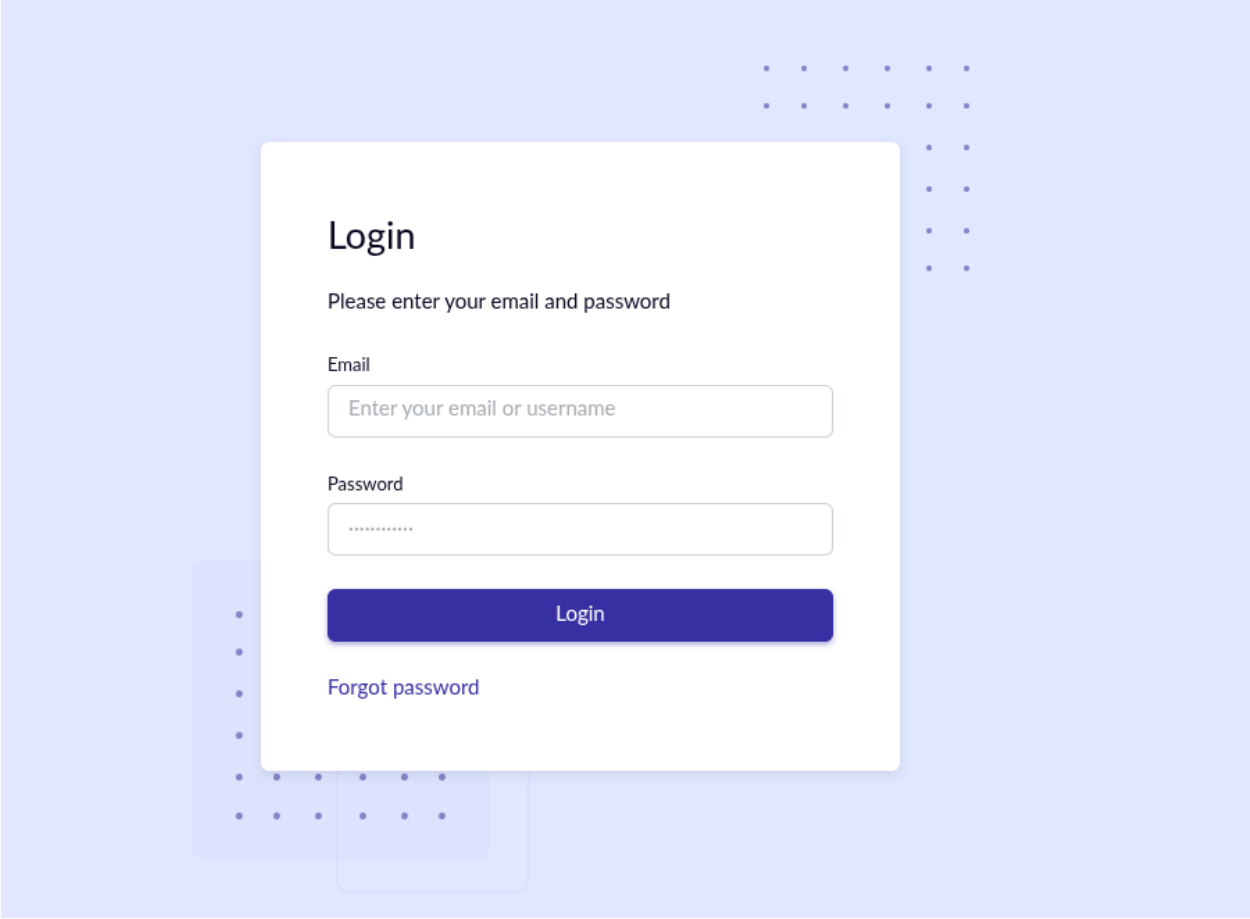


Рисунок В.1 – Знімок екрану сторінки входу

Після успішної автентифікації відкриється сторінка зі списком проєктів локалізації, до яких ви маєте доступ (рисунок В.2).

Створення проєкту

На сторінці списку проєктів адміністратор може створити новий проєкт за допомогою кнопки «Add». Введіть назву та опис проєкту і натисніть «Add» (рисунок В.3).

Рисунок В.2 – Знімок екрану сторінки зі списком проєктів локалізації

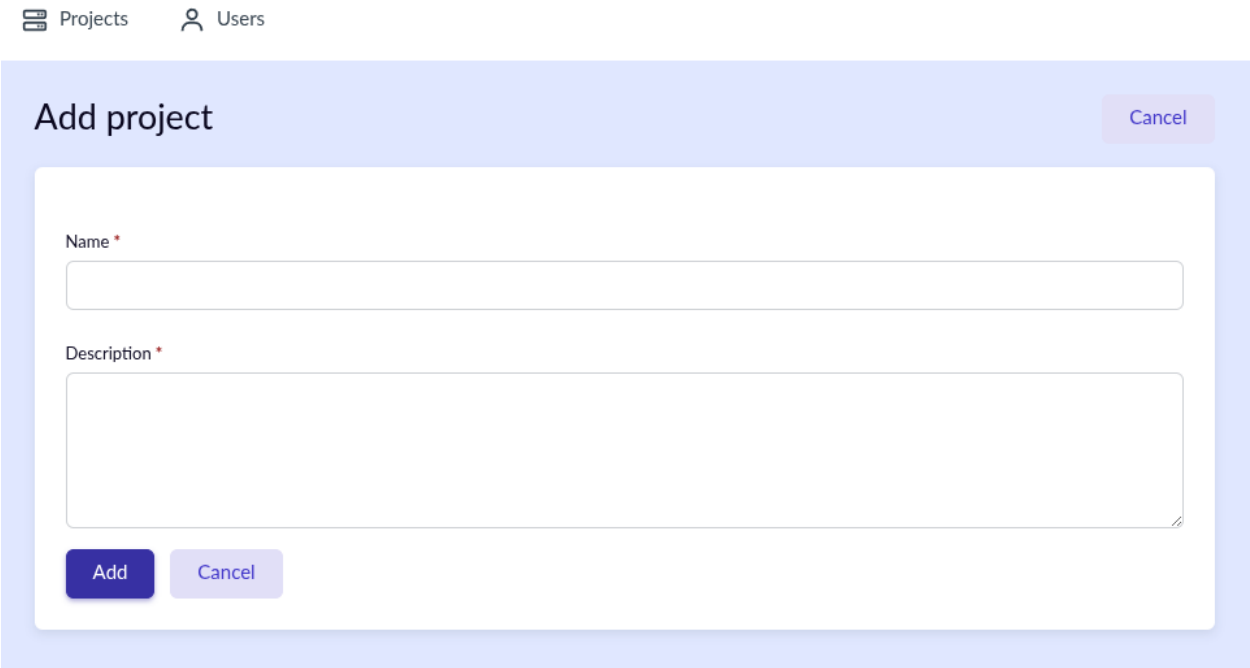


Рисунок В.3 – Знімок екрану сторінки створення нового проєкту локалізації

Створений проєкт з’явиться у загальному списку.

Налаштування мов проєкту

Перейдіть до сторінки проєкту (рисунок В.4) і натисніть «Add». Оберіть мову зі списку та підтвердіть. Додані мови відображаються у вигляді переліку зі станом прогресу перекладу для кожної мови.

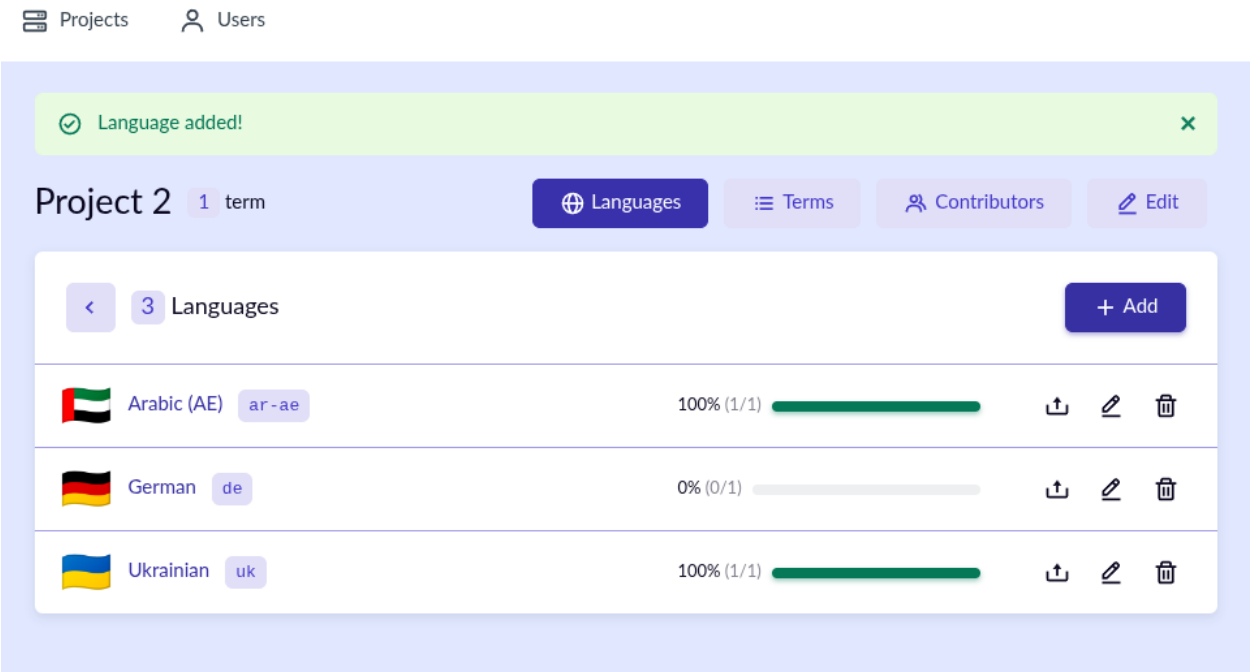


Рисунок В.4 – Знімок екрану сторінки проєкту після додавання нової мови

Робота з фразами та перекладами

На сторінці фраз (рисунок В.5) ви можете додати нову фразу за допомогою посилання «Add», ввести оригінальний текст, контекст та коментар. Для перекладу фрази натисніть на її назву – відкриється додатковий елемент інтерактивного перекладу з полями для всіх мов проєкту. Після введення тексту перекладу натисніть «Save».

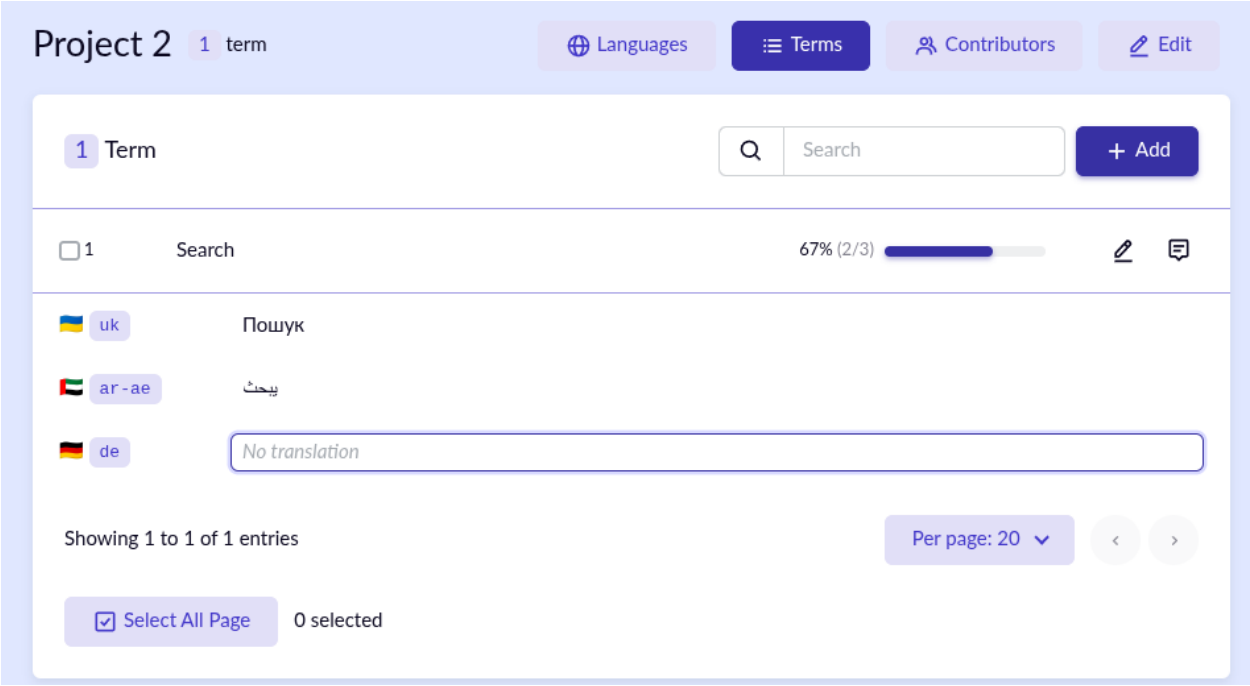


Рисунок В.5 – Знімок екрану сторінки зі списком фраз проєкту

Імпорт та експорт ресурсних файлів

Для імпорту ресурсного файлу натисніть кнопку «Import». На сторінці імпорту ресурсного файлу (рисунок В.6) оберіть мову, для якої виконується імпорт, та файл у форматі PO, JSON або CSV.

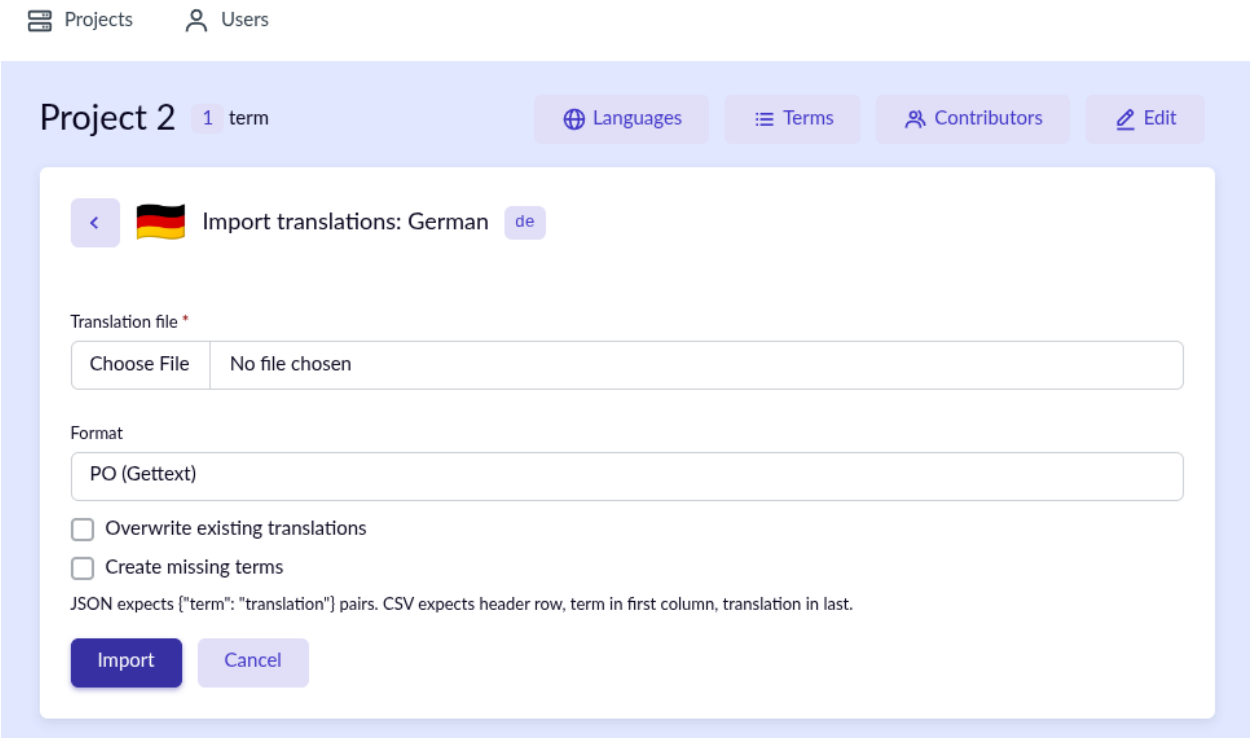


Рисунок В.6 – Знімок екрану сторінки імпорту ресурсного файлу

Для експорту перекладів натисніть кнопку «Export» поруч із потрібною мовою та оберіть формат вихідного файлу.

Керування учасниками проєкту

Адміністратор може додавати та видаляти учасників проєкту через сторінку «Contributors». Користувач, який не є учасником проєкту, не бачить цей проєкт у своєму списку.

REST API

Для машинної синхронізації перекладів використовуйте персональний API-ключ, який можна згенерувати на сторінці «API Key» (рисунок В.7), перейшовши до неї через меню користувача.

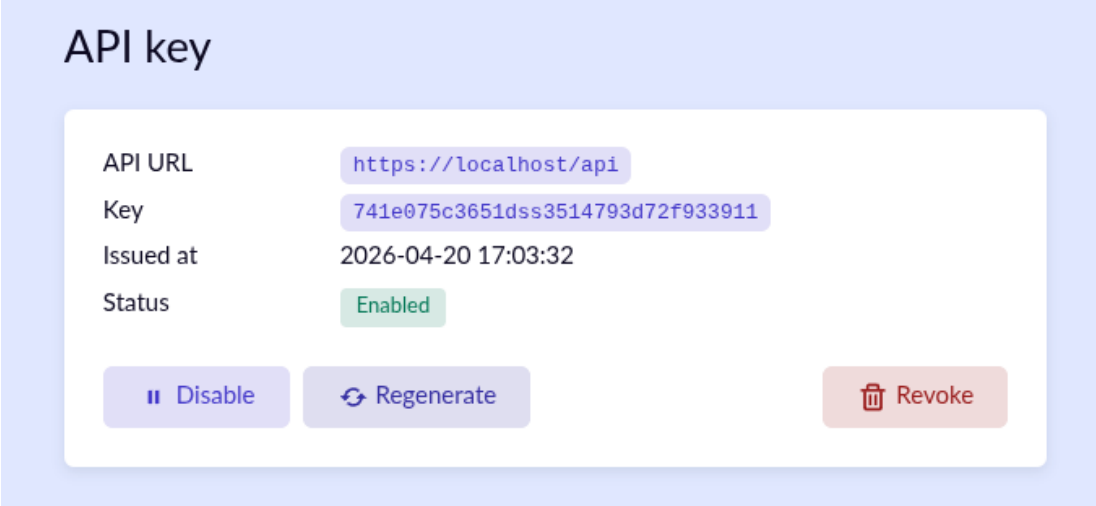


Рисунок В.7 – Знімок екрану сторінки управління персональним API-ключем

Кнопка «Regenerate» на цій сторінці дозволяє згенерувати новий API-ключ, старий при цьому буде видалено. Кнопка «Disable» дозволяє тимчасово відключити ключ. Кнопка «Revoke» видаляє збережений ключ.

Ключ передається в HTTP-заголовку *Authorization* запиту до точки доступу /api/projects/export. Відповідь повертається у форматі JSON.

Адміністрування користувачів

Сторінка «Admin → Users» доступна користувачам із роллю «адміністратор» і дозволяє переглядати, додавати, редагувати та видаляти облікові записи інших користувачів. Кожному користувачеві призначається роль «адміністратор» або «користувач», що визначає набір доступних дій.

ДОДАТОК Г – ОПИС ВЕБЗАСТОСУНКУ УПРАВЛІННЯ ПЕРЕКЛАДАМИ ДЛЯ БЕЗПЕРЕБІЙНОЇ ЛОКАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вступ

Найменування програми: вебзастосунок управління перекладами для безперебійної локалізації програмного забезпечення (Translation Management Web Application).

Позначення програми: TMWA.

1 Функціональне призначення

1.1 Класи розв’язуваних завдань та призначення програми

Призначенням розроблюваного програмного забезпечення є автоматизація процесу локалізації програмного забезпечення шляхом створення централізованого середовища для управління перекладами інтерфейсів програм. Вебзастосунок забезпечує імпорт ресурсних файлів у форматах PO, JSON, CSV; інтерактивний інтерфейс для перекладу фраз із підтримкою контексту та коментарів; підтримку множинних мов у рамках одного проєкту; експорт готових локалізацій у потрібних форматах та REST API для автоматичної синхронізації перекладів із кодовою базою.

1.2 Функціональні обмеження на застосування

Кількість одночасних користувачів обмежується технічними характеристиками сервера. Розмір ресурсного файлу, що імпортується, обмежено налаштуваннями PHP-інтерпретатора (за замовчуванням 8 МБ).

2 Опис логічної структури

2.1 Опис структури

Програмне забезпечення реалізовано із застосуванням принципів об’єктно-орієнтованого програмування (ООП). Сутності предметної області

(User, Project, Locale, Term, Translation) представлено окремими класами Entity, а доступ до даних інкапсульовано у класах Table.

Архітектура відповідає шаблону MVC (Model-View-Controller): Model відповідає за роботу з базою даних, View – за подання даних користувачеві у вигляді HTML-сторінок, Controller – за обробку HTTP-запитів та координацію роботи Model і View.

2.2 Мови програмування

Серверну частину програмного забезпечення реалізовано на мові PHP (версії 8.1 і вище) із використанням фреймворку CakePHP 5. Клієнтську частину реалізовано на мовах HTML5, CSS3 та JavaScript із використанням бібліотеки jQuery та шаблону Sneat (Bootstrap 5). Структуру бази даних описано на мові SQL (діалект MySQL).

2.3 Вхідні та вихідні дані

Вхідними даними є: дані автентифікації (електронна пошта та пароль або API-ключ); дані форм створення та редагування проєктів, фраз і перекладів; ресурсні файли локалізації у форматах PO, JSON, CSV.

Вихідними даними є: HTML-сторінки інтерфейсу користувача; JSON-відповіді REST API; ресурсні файли локалізації, що завантажуються для зовнішнього використання.

3 Склад і функції

3.1 Склад програмного забезпечення

Програмне забезпечення складається із серверної частини (PHP-код проєкту, шаблони вигляду, конфігураційні файли), фреймворку CakePHP 5, бази даних MySQL зі схемою із семи таблиць та клієнтських ресурсів (CSS, JavaScript, шрифти і зображення шаблону Sneat).

3.2 Функції програмного забезпечення

Програмне забезпечення реалізує наступні основні функції: автентифікація користувачів та керування персональними даними; створення, редагування та видалення проєктів локалізації; додавання, редагування та видалення мов проєкту; додавання, редагування та видалення фраз проєкту; інтерактивний переклад фраз із контекстом; імпорт-експорт ресурсних файлів у форматах PO, JSON, CSV; адміністрування користувачів та ролей; надання REST API для зовнішніх систем.

4 Умови застосування

4.1 Вимоги до складу та параметрів технічних засобів

Сервер: процесор з тактовою частотою не менше 4 ГГц; оперативна пам’ять не менше 4 ГБ; вільне місце на жорсткому диску не менше 1 ГБ; стабільне підключення до мережі Інтернет.

Клієнт: процесор з тактовою частотою не менше 1 ГГц; оперативна пам’ять не менше 2 ГБ; підключення до мережі Інтернет.

4.2 Вимоги до програмної сумісності

Сервер: операційна система Linux (Ubuntu 20.04+ або аналог); PHP 8.1+; СКБД MySQL 8.0+; вебсервер Apache 2.4+ із модулем mod_rewrite. Клієнт: сучасний вебоглядач (Google Chrome 90+, Mozilla Firefox 90+, Safari 14+, Microsoft Edge 90+).

4.3 Спосіб виклику програми

Виклик програмного забезпечення відбувається шляхом відкриття у вебоглядачі URL-адреси сервера, на якому розгорнуто вебзастосунок.

Вхідною точкою у програмне забезпечення є файл index.php, що ініціалізує екземпляр класу App\Application та передає управління фреймворку CakePHP. Фреймворк на основі URL-адреси визначає контролер і метод (action), що оброблятимуть запит.

ДОДАТОК Д – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ ВЕБЗАСТОСУНКУ УПРАВЛІННЯ ПЕРЕКЛАДАМИ ДЛЯ БЕЗПЕРЕБІЙНОЇ ЛОКАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1 Об’єкт випробувань

Об’єктом випробувань є вебзастосунок управління перекладами для безперебійної локалізації програмного забезпечення.

2 Мета випробувань

Мета випробувань – перевірити можливості програмного забезпечення щодо відповідності вимогам технічного завдання.

3 Вимоги до програми

При проведенні випробувань необхідно перевірити, що розроблене програмне забезпечення реалізує наступні функції:

- автентифікація користувача за електронною поштою і паролем;
- створення, редагування та видалення проєкту локалізації;
- додавання та видалення мов проєкту;
- додавання, редагування та видалення фраз;
- збереження перекладу фрази;
- імпорт ресурсних файлів у форматах PO, JSON, CSV;
- експорт перекладів у форматах PO, JSON, XLIFF, CSV;
- надання переліку перекладів через REST API за персональним API-ключем.

4 Вимоги до програмної документації

До складу програмної документації, що надається на випробування входить

- технічне завдання;

- текст програми;
- інструкція користувача;
- опис програми;
- програма і методика випробувань.

5 Засоби та порядок випробувань

5.1 Засоби випробувань

Склад технічних засобів, які мають використовуватися при випробуваннях: комп’ютер з процесором Intel Core i3 або еквівалентом, оперативною пам’яттю 4 ГБ та підключенням до мережі Інтернет.

До складу програмних засобів, які мають використовуватися при випробуваннях, належать операційна система Ubuntu 22.04 LTS з пакетами PHP 8.1, MySQL 8.0, Apache 2.4; вебоглядач Google Chrome версії 120 або вище.

5.2 Порядок проведення випробувань

1) Перевірка комплектності програмної документації;

2) Перевірка автентифікації користувача;

3) Перевірка функцій управління проєктами, фразами та перекладами; перевірка функцій імпорту-експорту ресурсних файлів; перевірка REST API.

6 Методи випробувань

1) Випробування функції автентифікації користувача

Запустити вебоглядач та перейти за адресою вебзастосунку. Ввести коректні електронну пошту та пароль адміністратора. Натиснути кнопку «Login».

Очікуваний результат: користувач успішно входить у систему та перенаправляється на сторінку списку проєктів.

2) Випробування функції створення проєкту локалізації

На сторінці списку проєктів натиснути кнопку «Add». Ввести назву та опис проєкту. Натиснути кнопку «Save».

Очікуваний результат: проєкт створюється, після чого виконується перенаправлення на сторінку списку проєктів, у якому міститься новий проєкт, а користувачу відображається повідомлення про те що успіх.

3) Випробування функції імпорту ресурсного файлу

На сторінці проєкту обрати мову та натиснути кнопку «Import». Обрати ресурсний файл у форматі PO, JSON або CSV. Натиснути кнопку «Import».

Очікуваний результат: фрази та переклади з ресурсного файлу імпортуються до проєкту, користувачу відображається повідомлення про кількість успішно імпортованих фраз та перекладів.

4) Випробування функції перекладу фрази

Перейти до списку фраз проєкту. Обрати фразу. Ввести текст перекладу для потрібної мови. Натиснути «Save».

Очікуваний результат: переклад зберігається у базі даних, у списку фраз відображається оновлений прогрес перекладу.

5) Випробування функції експорту локалізації

На сторінці проєкту обрати мову та натиснути кнопку «Export». Обрати формат експорту (PO, JSON, XLIFF або CSV).

Очікуваний результат: у вебоглядачі починається завантаження файлу локалізації обраного формату, що містить усі переклади проєкту для обраної мови.

6) Випробування REST API

Виконати GET-запит до точки доступу /api/projects/export із HTTP заголовком Authorization, що містить персональний API-ключ користувача.

Очікуваний результат: сервер повертає HTTP-код 200 та JSON-документ із переліком проєктів, мов та перекладів, доступних користувачеві. Якщо API-ключ невалідний, сервер повертає HTTP-код 401.