Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування факультету)

Кафедра обчислювальної техніки
(повна назва кафедри)

Пояснювальна записка

до магістерської кваліфікаційної роботи

магістр
(освітньо-кваліфікаційний рівень)

на тему: «Програмний засіб для локалізації веб та мобільних додатків. Частина 1. «Серверна частина»

Виконав: студент 2 курсу, групи 1КІ-19м
напряму підготовки:
123 «Комп'ютерна інженерія»
(шифр і назва напряму підготовки)

Смольц Д. О.
(прізвище та ініціали)

Керівник: Черняк О. І.
(прізвище та ініціали)

м. Вінниця – 2020 року

АНОТАЦІЯ

У даній роботі розглянуто актуальність глобалізації та її частин: локалізації та інтернаціоналізації. Досліджено різноманітні джерела та статистичні дані світових науково-популярних джерел. Описано існуючі методи та варіанти реалізації глобалізації на бекенді. Обрано технологій та інструменти для розробки серверної частини додатку. Запропоновано універсальний веб програмний засіб для локалізації веб та мобільних додатків. Розроблено серверну частину застосунку за допомогою тришарової серверної архітектури. А також використано передові технології та сервіси, що дозволило зробити систему легкою підтримки та масштабування, кросплатформеною та надійною.

ANNOTATION

This paper considers the relevance of globalization and its parts: localization and internationalization. Examined various sources and statistics of world popular science sources. Described the existing methods and variants of globalization implementation on the backend. Selected technologies and tools for the development of the server part of the application. Offered the universal web software for the localization of web and mobile applications. Developed the server part of the application using a three-layer server architecture. And also used advanced technologies and services, which allowed making the system easy to support and scale, cross-platform and reliable.

ЗМІСТ

ВСТУП

На даний момент людство існує у глобальному світі, тому кожен, хто прагне знайти інформацію, не обмежений своїм місцезнаходженням. Відкритість мережі Інтернет дозволяє знайти майже будь-що у вільному доступі.

Щоправда, необхідна для нас інформація не завжди може бути актуальною і корисною. Інколи вона просто застаріла по часу її публікації, а часом бар'єром стає мова, якою вона подана. На жаль, лише одиниці людей можуть розуміти десятки мов, всім іншим доводиться покидати веб-сторінки одну за одною, поки не зустрінуть знайому для себе мову. Так відбується тому, що користувачі мережі Інтернет у всьому світі не хочуть витрачати зайвий час на розбір та переклад того, що їм пропонують, вони розраховують на доступ до різноманітних продуктів та інформації їх рідною мовою.

Деякі розробники розуміють важливість створення їхніх продуктів (веб-сайтів та додатків) доступними для всього світу і тому мають різні мовні версії на своїх сайтах. Проте, переклад — це лише частина та його зовсім недостатньо.

Просто перекладаючи текст веб-сайту на іншу мову і фактично не пристосовуючи його, можна створити повідомлення, що буде заплутувати або стане комічним, а це те, що ніколи не показує компанію в найкращому світлі. У деяких випадках речення, яке має гарний сенс в одній мові, насправді може бути образливим для іншої. Саме тому замість простого перекладу тексту важливо, щоб інформація була повністю локалізованою. Іншими словами, це означає, що кожна сторінка буде адаптована до мови та культури цільового ринку.

Коли розробники програмних засобів починають планувати локалізацію їхнього продукту, вони допускають одну типову помилку. Базовою мовою для локалізації зазвичай обирають англійську, адже вона вважається основною світовою мовою. Але приблизно для 70% відвідувачів веб-сайтів англійська мова не є рідною і вони віддають перевагу іншим сайтам [1].

Для того, щоб забезпечити локалізацію веб-додатку, потрібно визначити цільові ринки та знайти гарних перекладачів, які допомогли б як і з власне перекладом тексту на обрані мови, так і з його адаптацією. Одразу можна сказати,

що це дорогий та достатньо тривалий процес. Також можна скористатись платформами, де уже є зібрані спеціалісти по локалізації. Такий варіант достатньо спрощує цей процес.

Наразі, платформ, які надають послуги з локалізації, є лише невелика кількість. Зокрема, такими є сервіси, як Level Up Translation, Nitro, Rev, Lilt та інші. Але, на жаль, кожен з них має деякі великі мінуси, такі як: надмірна ціна за надання послуг, ізольованість замовника від роботи перекладачів над його проектом, недостача повного комплексу бажаних інструментів та інше.

Ряд недоліків, які наведені раніше, робить розробку насправді професійного додатку, який дозволить перекладачам поєднувати свою роботу з машинним перекладом, а також забезпечуватиме коректну, швидку та безперебійну роботу платформи, **актуальною задачею**.

Мета дослідження магістерської роботи це покращення процесу поширення доступності для власників веб та мобільних додатків, за рахунок надання практичної платформи для локалізації їхнього контенту та інтерфейсу, де спеціалісти-перекладачі зможуть пропонувати свої послуги. А також забезпечення доцільного розподілу трудових ресурсів, поєднання живого та машинного перекладу, надання можливості відслідковувати роботу перекладачів та інструментів для проведення аналізу їх прогресу і розробка гнучкого та швидкого веб-додатку з інтуїтивно зрозумілим інтерфейсом для спрощення роботи та комунікації перекладачів та замовників.

Задачі дослідження магістерської роботи:

— провести аналіз сучасних способів та методів втілення локалізації різних додатків на серверній частині;

— розробити тришарову архітектуру серверної частини системи;

— налаштувати взаємодію додатку з реляційною та нереляційною СУБД, а також файловим сховищем;

— забезпечити кросплатформеність додатку;

— налаштувати підтримку роботи з сервісами машинного перекладу;

— створити зручне місце для спілкування між користувачами (чат).

Об’єкт дослідження це явище локалізації веб-додатків та інших застосунків, яке забезпечує примітивний переклад необхідного тексту на цільову мову та подальшу його адаптацію згідно з обраною країною, регіоном чи мовною групою. А також варіанти реалізації даного явища із серверної частини додатку.

Предметом дослідження є процес розробки серверної частини веб-додатку, яка надасть можливість використовувати даний веб-додаток на всіх існуючих платформах і при цьому забезпечить високу та стабільну швидкість роботи, а також матиме способи та методи взаємодії з реляційними та нереляційними даними та сторонніми існуючими сервісами машинного перекладу.

Наукова новизна отриманих результатів полягає у поліпшені способу локалізації при розробці веб та мобільних додатків шляхом зміни форми даних, які представляються, організації фрагментації даних, які локалізуються, що надає можливість ефективнішого стиснення інформації та забезпечення безперебійної роботи системи про великих навантаженнях, а також шляхом створення зручної платформи для приватного спілкування чи спілкування у групах між різними типами користувачів. Використання програмного засобу, покращеного за даним методом, дозволяє спростити процес локалізації.

Практичне значення одержаних результатів магістерської роботи:

— покращено метод перекладу та локалізації додатків;

— спрощено форму представлення даних та організацію;

— створено можливість продуктивного стиснення інформації;

— надано можливість вільного спілкування між користувачами;

— реалізовано серверну частину платформи для локалізації веб та мобільних додатків.

Апробація результатів магістерської роботи полягає у тому, що було проведено ряд досліджень та написано статтю, яка описує проблему локалізації веб та мобільних додатків. Наведена стаття була розглянута на XLIX Науково-технічній конференції підрозділів Вінницького національного технічного університету (2020) [2].

1 АНАЛІЗ ІСНУЮЧИХ СПОСОБІВ ТА ЗАСОБІВ ЛОКАЛІЗАЦІЇ

1.1 Глобалізація та роль локалізації при глобалізації

1.1.1 Поняття глобалізації

Глобалізація — це процес планування та розробки такої системи, яка надаватиме культурно вірну інформацію, та, маючи лише один екземпляр програмного забезпечення, може приймати багатомовні дані для обробки. Щоб мати право називатись культурно вірним, даний застосунок повинен не лише містити інформацію на обраній мові, а також дотримуватись дрібних деталей різних країн та національностей. Наприклад, сортування списків має відповідати нормалізованому порядку для даної місцевості, форматування дат повинне бути відповідним до національних стандартів, а нумерація та числа — розділятися загальноприйнятими знаками.

Іншими словами, глобалізація є поєднанням технічної та перекладацької складової. Саме тому її поділяють на дві частини: локалізацію та інтернаціоналізацію (див. рис. 1.1).

Рисунок 1.1 — Процес глобалізації

1.1.2 Інтернаціоналізація

Інтернаціоналізацією називають процес проектування додатку, який абсолютно не залежить від особливостей мови, культури чи місцевості ринків, для яких він розроблений. Часто замість даного поняття вживається i18n, де 18 — це

кількість літер між «і» та «n» в англійському слові.

Інтернаціоналізація зазвичає складається з:

— проектування та імплементацію додатку таким, який міг би вільно використовуватись по всьому світу, а саме ізоляцію коду від елементів інтерфейсу та даних, що відображаються користувачу;

— підтримка різних систем писемності (кирилиця, латиниця, арабська писемність, китайська писемніть та інші);

— підтримка можливості керувати мовами сайту окремо від самого сайту.

1.1.3 Локалізація як фінальна частина глобалізації

Локалізація — це процес пристосування змісту, програми або веб-додатку до місцевих, мовних та інших вимог конкретного регіону. Поняття локалізації також має скорочений варіант для використання, а саме l10n, де 10 це кількість літер між буквами «l» та «n».

Помилково термін «локалізація» вживається у значенні перекладу, але його визначення є значно ширшим. Мається на увазі адаптація наступних аспектів:

— валюта;

— різні правові чи релігійні норми;

— використання розкладки клавіатури;

— символіка, знаки та кольори;

— зображення;

— формати чисел, дати;

— та інше.

Таким чином інтернаціоналізація підготовлює платформу, яка значно спрощує локалізацію. А локалізація, в свою чергу, є основним та завершальним етапом глобалізації. [3]

1.2 Важливість локалізації в сучасний час

На даний час можна виділити п’ять причин чому локалізація додатків є важливим процесом.

Перша причина – це охоплення ширшої аудиторії. Локалізація застосунків зробить їхню більш впізнаваними за межами країни, на яку спершу орієнтувався додаток, адже це полегшить вихід на нові ринки. Також локалізація створює необхідний рівень довіри і перетворює потенційних клієнтів на постійних.

Другою причиною є те, що користувачі бажають бачити індивідуальний контент. Такий вміст важливий для кожного користувача, і саме це робить локалізація – вона робить контент безпосередньо близьким до користувача. Тому використання локалізації призводить до швидкого зростання кількості аудиторії та клієнтів.

Третьою причиною є уникнення ризику. У певних місцевостях та культурах символіка, зображення тварин та рослин, а також кольори можуть нести різний сенс та асоціації, а це неодмінно вплине на сприйняття веб-сайту або програмного забезпечення. Вагому роль мають релігійні та політичні аспекти у текстах, символах або зображеннях, а це може стати підставою для заборони використання додатку на території певної країни. Забезпечення відсутності подібних елементів має стати для власника застосунку першим пріорітетом, а використання локалізація допоможе досягти уникнення подібного ризику.

Четверта причина — це можливість збільшити кількість відвідувачів шляхом удосконалення користувацького досвіду. Локалізуючи документацію, дописи в блозі, поширені запитання, електронні книги, путівники, звіти та підтримку клієнтів, додаток зможе гарантувати, що задовольнятиме конкретні потреби кожного користувача. А задоволені клієнти будуть гарно відгукуватись та поширюватимуть поради коистувати саме цей застосунок. Вони стануть рушійною силою та створять та будуть підтримувати певну спільноту, забезпечуватимуть розповсюдженість додатку, адже найуспішнійший маркетинг – це наявність реальних відвідувачів, які будуть ділитись гарними враженнями від користування.

І останньою причиною є те, що локалізація потрібна для географічного поширення розвитку. Компанії, які досягли успіху на міжнародному ринку, можуть підтвердити, що локалізація є основою для міжнародного визнання. Але варто ще раз нагадати, що локалізація не є безапеляційним варіантом для підкорення ширшої

аудиторії, адже потрібні також інші чинники, щоб налаштувати свою справу на глобальне поширення. Проте, якщо завчасно розпочати вірно планувати локалізацію, не забути про етап інтернаціоналізації при розробці додатку, то надалі процес локалізації буде значно легшим [4-5].

1.3 Типи локалізації

Зважаючи на те, що локалізація є складним і дорогим процесом, розробники певних продуктів вдаються до неї не повністю. Таким чином виділяють кілька типів локалізації за ступенем поглиблення: локалізація інтерфейсу, текстова, графічна або звукова локалізація. Але потрібно зауважити, що найпростіша локалізація майже завжди дорівнює порівняно найменшій розповсюдженості додатку.

Вдалим прикладом для пояснення даних типів локалізації можуть стати комп'ютерні відеоігри. Таким чином найбільш поверхневим є перший названий тип локалізації (локалізація інтерфейсу). Назва даного типу вказує на те, що лише інтерфейс з яким взаємодіє користувач, буде «перекладено» на його мову, а саме кнопки, поля для введення і подібне.

У більшості випадків у відеоіграх є сюжет, який часто подається у вигляді текстів. Якщо вони є локалізованими, то доречно стверджувати, що при глобалізації даної гри використовували текстову локалізацію (яка, звісно, включає в себе і локалізацію інтерфейсу). А якщо «перекладено» і текст на картинках, то графічну локалізацію, яка містить в собі два попередні типи.

Звукова локалізація є ізольованим видом від локалізації інтерфейсу, тесту та графічних елементів. Зазвичай відеоігри не можуть існувати без монологів, діалогів чи розповідей, тому розробники застосовують локалізацію звукового супроводу.

Найглибшим типом локалізації є культурна адаптація. Це адаптація усього існуючого матеріалу відповідно до моральних норм, традицій, засад та ментальності певного регіону, місцевості або країни, задля покращення його осягання кінцевими користувачами. Перекладач мусить досконало володіти обраною мовою, розумінням потреб користувачів, а також темою, конкретними термінами, тощо. Локалізація такого виду застосовується достатньо рідко, оскільки

переважно однакові сторінки веб-сайту, перекладені різними мовами, ґрунтовно відрізняються, а це викликає незручності. Але саме такі застосунки є найбільш популярними.

Культурній адаптації включає незліченну кількість аспектів. Наприклад, у певних культурах тварини, рослини та кольори можуть нести різний сенс та викликати різні асоціації, що безпосередньо впливає на сприйняття веб-сайту або інтерфейсу програмного забезпечення. а це неодмінно вплине на сприйняття веб-сайту або програмного забезпечення. Вагому роль мають релігійні та політичні аспекти у текстах, символах або зображеннях, а це може стати підставою для заборони використання додатку на території певної країни. Але, найскладнішим пунктом у даному типі локалізації є будь-яке зображення людини, адже це викликає непорозуміння в інтерпретації жестів та етносу. Таким чином, розробники застосунків намагаються не використовувати такі зображення і єдиним задовольняючим варіантом є умовний силует [6].

Також при культурній адаптації відеоігор можуть виникнути підстави для повного перегляду персонажів та загалом сюжету, адже, наприклад, гра по грецькій міфології буде не зрозумілою для користувачів із Японії або Китаю. Але такі кардинальні зміни не завжди є обов’язково потрібними. Гра «Super Mario» поширена по всьому світу та має однакових героїв, проте, не можна стверджувати, що якби розробники використовували локалізованих персонажів (див. рис. 1.2) відеогра не набула б ще більшої популярності.

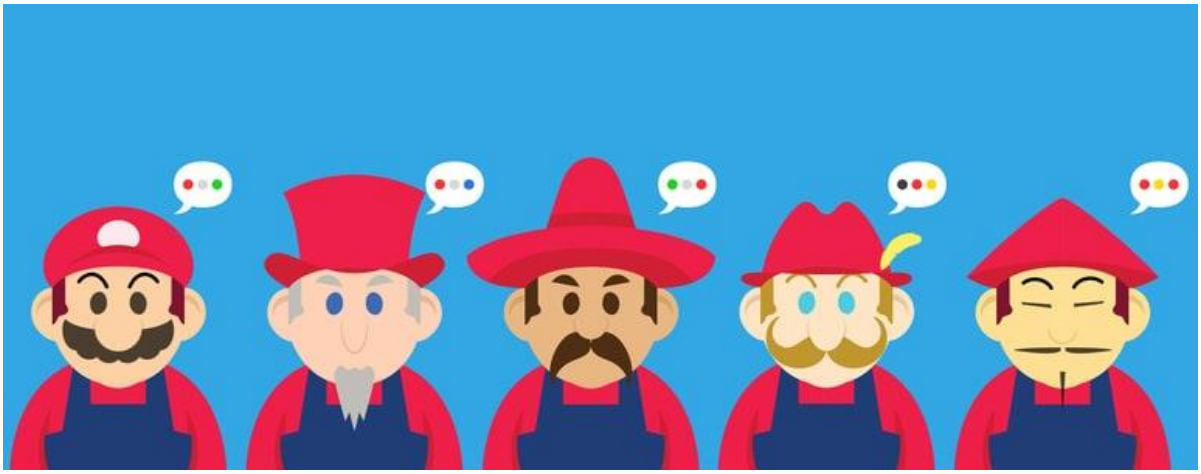

Рисунок 1.2 — Приклад локалізації персонажів відеогри

1.4 Принципи успішної локалізації

Для будь-якого бізнесу, який хоче розширитися по всьому світу, ключовим фактором є глобальна локалізація вмісту. Розглянувши статті із блогу перекладів та локалізації («Translation & Localization Blog») можна виділити шість порад для створення успішної локалізації.

Першою є заклик подумати про "різноманітність", адже кожна культура відрізняється. Плануючи свою стратегію для певної культури, потрібно переконатися, що враховуються відмінності, наприклад, як користувачі використовують і купують товари, а також те, як різні культури сприймають колір або цифри.

Наприклад, виробник м’ячів для гольфу вирішив упакувати їх в коробки по чотири в Японії. У японській культурі (як і в китайській) люди приділяють особливу увагу цифрам навколо себе. Деякі цифри вважаються дуже поганими, тому що їх вимова співзвучна з вимовою певних негативних слів, тому в побуті їх майже не зустрінеш. Однією із найбільш нещасливих цифр вважається «4», що робить чотири м’ячів для гольфу в одному наборі поганою ідеєю.

Друга порада – це постійна практика культурної чутливості.

Варто вести справи із тими людьми, хто має фундаментальні знання культури, цільової аудиторії. Культурні дрібниці створюють великий ефект на те, яким чином потрібно зацікавлювати певних користувачів. Необхідно переконатись, що не використовуються неприйнятні для людей зображення, словниковий запас чи барви. Наприклад, у Англії білий колір означає елегантність та чистоту, а країнах Азії білий — це нещастя.

Третя порада вказує, що глобальна локалізація вмісту означає розуміння переваг місцевих споживачів. Однією із достатньо популярних думок є те, що всі потреби споживачів однакові у всьому світі. Насправді все не так, наприклад, у Великобританії споживачі надають перевагу оплаті своїх товарів через Інтернет за допомогою дебетової або кредитної картки, а у Японії багато споживачів все ще вважають, що краще за все платити готівкою через міні-магазини.

Дуже важливо гарно дослідити якими є потреби та очікування потенійних

клієнтів, і не покладатись на стереотипні припущення. Наприклад, мережа універсальних магазинів Walmart зробила помилку припустивши, що найкращим місцем знаходження для їх точок продажу, будуть місця неподалік від індустріальних парків у Китаї, пропустивши те, що клієнти там надають перевагу робити покупки біля місця свого проживання, а не роботи.

Наступною, четвертою порадою є створення світових торгових марок. Вибір торгової марки, яку не потрібно перекладати, може допомогти вам створити репутацію світового бренду. Якщо все-таки було обрано ім'я, яке потрібно буде перекласти, це може зайняти більше часу і витрат, оскільки доведеться придумувати нові торгові марки під час розширення по всьому світу. Якщо справді подобається ім'я, яке потрібно буде перекласти, то слід зважити всі плюси і мінуси використання цього імені на інших ринку та його перекладу.

П'ята порада — це створення унікального та якісного контенту.

При написанні контенту, потрібно уникати використання великої кількості сленгу чи ідіом. Забезпечивши це, текст стане простішим для перекладу, і стане менше шансів мати проблеми у майбутньому. Розробіть деякі редакційні поради та запевніться, що автори тексту будуть постійно дотримуватись їх. Також слід інвестувати в гарних перекладачів, щоб бути переконаним, що кожному слову у вмісті контенту надається необхідна увага спеціаліста [7].

Останньою, але не найменш важливою порадою, є настанова не змушувати клієнта додатку задумуватись. Для кожного найкраще сприймається текст поданий його рідною мовою. Якщо відвідувач знаходить необхідну йому інформацію, але іншою мовою, скоріш за все, йому буде простіше витратити час та зусилля на пошуки, а не на переклад іноземного тексту тому, що це спонукає користувача задумуватись і втрачати на розбір інформації забагато часу. Отже, успішна локалізація це та, яка є найбільш приближеною для пересічного відвідувача і та, що звільнить його від подальших пошуків та витрат часу на переклад окремих слів, пошук визначень сталих виразів, які не можуть бути легко перекладеними на різні мови, власноруч, [8].

Глобальна локалізація контенту, яка є особливістю, є вагомою для будь-якої

справи, яка має на меті підкорення міжнародної аудиторії. Розробка та локалізація унікального та якісного контенту, який враховує всі нюанси окремих культур, поставить веб, мобільний чи будь-який інший застосунок на шлях до успішного підкорення міжнародного ринку.

1.5 Етапи реалізації локалізації

Можна виділити наступні п’ять етапів для успішної реалізації локалізації додатку, відеогри чи веб-сайту:

Першим етапом є вибір цільового ринку, тобто вибір на які країни чи регіони буде поширюватись додаток. Гарним способом для здійснення подібного вибору є дослідження аналітики веб-сайтів чи додатків, що є аналогами. Потрібно виділити кожну велику кількість відвідувачів або постійних користувачів веб-сайту, які походять з певного місця, та взяти цю місцевість до уваги та впроваджувати локалізацію для неї.

Далі потрібно розпочати дослідження даної аудиторії. Важливо пам’ятати, що те, що якщо щось працює на місцевому ринку, не обов’язково буде настільки ефективним на іншому. Тому гарною ідеєю є витрата певного часу на дослідження цільової аудиторії. Проте також потрібно зауважити, що додатку, можливо, доведеться адаптуватися для зв’язку з новою аудиторією.

Другим етапом є проведення технічної підготовки додатку. Потрібно дослідити всі технічні відмінності між оригінальною мовою додатку та цільовою аудиторією — варто звернути увагу на дату та час, правила плюралізації, валюту та цифри. Слід також врахувати особливості цифрових інструментів, таких як прийняті в тій чи іншій країні платіжні системи та інтеграції соціальних мереж.

Помилковою є думка про те, що усі сприймають символи або цифри однаково. Насправді світ є неймовірно різноманітним. Це є причиною, чому варто працювати із спеціалістами по перекладу та локалізації, які зможуть надавати підказки щодо будь-яких технічних особливостей, які необхідно врахувати. Цей етап варто об’єднати з інтернаціоналізацією застосунку.

Третій етап є найцікаішим — це переклад контенту на усі необхідні бізнесу

мови із мови оригіналу. На жаль даний етап не настільки примітивний, як може здатись із самого спочатку.

Наприклад, якщо цільовим ринком є Мексика, то можна подумати, що вся іспанська однакова, але це не так, адже це неймовірно різноманітна мова з багатьма регіональними адаптаціями та діалектами. Це не означає, що мексиканець не зможе зрозуміти іспанську мову з Іспанії – в більшості випадків це не завдасть складнощів. Але подібний контент стане викликом для користувача, адже йому потрібно буде самостійно адаптувати певні вирази чи словосполучення. Найкращим виходом з подібної ситуації є вибір перекладача-мексиканця, який би переклав вміст на місцеву іспанську мову.

Впровадження контенту для місцевої цільової аудиторії є четвертим етапом. Не зважаючи на те, що перекладач може бути родом із місця куди впроваджують додаток, навіть він може пропустити деякі нюанси, які інші користувачі можуть хибно зрозуміти. Наприклад, компанія Braniff Airlines потрапила в халепу, коли хотіла розповсюдити свої нові авіарейси на південь з тією ж самою кампанією, яку застосовували в інших англомовних країнах: «Літати в шкірі» («Fly in leather»). Загалом, переклад іспанською («Vuela en cuero») був вдалим у більшій частині Південної Америки, але він мав інше значення у Мексиці, де цей вислів перекладається як «літати голим». Подібне повідомлення стало зовсім далеким від того, яке планувала поширити авіакомпанія [9].

На сьогоднішній день існує можливість надіслати локалізовану версію додатку чи веб-сайту певній частині користувачів, потім отримати відгуки від них та аналізувати чи може такий варіант локалізації право на існування на даному ринку.

Завершальним етапом є налаштування служби підтримки користувачів. Невідворотним наслідком проведення процесу локалізації є отримання електронних запитів та листів від різномовних клієнтів. Таким чином, якщо додаток націлений на успіх в підтримці користувачів, потрібна розробити таку стратегію роботи, яка працюватиме для будь-яких клієнтів.

Вдалим варіантом є залучення клієнтів на даний ринок та передача їм всіх

звернень та запитань. Якщо це не є практичним або доступним, можна створити сторінку корисного вмісту мовою користувача, яким можна неодноразово користуватися, наприклад, сторінка поширених запитань та документів підтримки.

Також інколи потрібно переглядати статистику користувачів, для того, щоб знати до чого вони хочуть отримати доступ, які запитання зазвичай задають та досліджують. Потім необхідно надати вичерпні відповіді на всі поширені запити, та, обов'язково забезпечити їхню професійну локалізацію [10].

1.6 Мовні стандарти та коди

1.6.1 Призначення мовних кодів

Люди на нашій планеті в минулому та у сучасному світі використовували та використовують низку різних мов. Тому існує безліч причин чому потрібно ідентифікувати мову під час подання або запиту інформації [11].

Мовні налаштування користувача є обов'язковими для визначення для того, щоб у майбутньому застосувати належну обробку. До прикладу, мовні налаштування відвідувача у браузері, зазвичай, використовують для вибору відповідного веб-сайту. Також цю інформацію доцільно використовувати для застосування інструментів (наприклад, словників), що дають змогу обробляти або усвідомлювати контент різними мовами. Не менш важливим застосуванням знань про мову, яку використовує певний додаток, є можливість допомогти сервісам для різних типів обробки, а саме сервісам для перевірки орфографії, генерації транскрипції Брайля або синтезованого комп'ютером мовлення.

Одним із способів позначення мови, який існує наразі, є маркування інформаційного вмісту ідентифікатором («тегом»). Ці теги також можна використовувати для вказівки уподобань користувача при виборі інформаційного вмісту або для позначення додаткових атрибутів вмісту та пов'язаних з ним ресурсів. Наприклад, вказівка конкретної інформації про діалект, систему письма або орфографію, що використовуються в документі або ресурсі та може дозволити користувачеві отримати інформацію у формі, яку він може зрозуміти.

У підсумку, варто відзначити, що мета мовних тегів у тому, щоб допомогти

визначити письмові або розмовні мови. До них належать всі нині існуючі мови, враховуючи штучні, але виключаючи мови програмування, адже вони створені не для природнього спілкування людей.

1.6.2 Форматування мовних тегів та набори стандартів ISO

Тег мови складається з одного або послідовності декількох "підтегів", кожен з яких уточнює або звужує діапазон мови, визначений загальним тегом. Підтеги, у свою чергу, є послідовністю буквено-цифрових символів (літери та цифри), що відокремлюються між собою через дефіс.

Типи підтегів різняться за кількістю символів, розташуванням у тезі, а також вмістом. Вид будь-якого підтегу визначається лише за названими ознаками. Окрім цього, існує набір порад, пов’язаних із використанням великих букв у певних підтегах, але вони є не суттєвими.

Таким чином, мовний ідентифікатор «mn-Cyrl-MN» не відрізняється від «MN-cYRL-mn» або «mN-cYrL-Mn» (або будь-якої іншої комбінації), і кожна з цих варіацій має одне і те ж значення: монгольська написана кирилицею, що використовується в Монголії.

Проте, не зважаючи на те, що різниця між регістрами не є важливою в мовних ідентифікаторах, їх послідовне подання та визначене написання допомагає для кращого сприйняття. Тому Міжнародна організація зі стандартизації (англ. International Organization for Standardization, ISO), діяльність якої полягає у допомозі розвитку стандартизації та залежних міжнародних видів діяльності для гарантування міжнародного обміну послугами та розвиток співпраці в різних галузях, створила кілька переліків стандартів:

— ISO 639 — пов’язаний зі стандартизацією назв мов і мовних груп та складається з шести частин;

— ISO 15924 — стандарт для позначення назв письменностей. Визначає два набори кодів для ряду писемностей. Кожній писемності присвоюється два коди - числовий і літерний (містить чотири букви);

— ISO 3166-1 — частина стандарту ISO 3166, що містить коди назв країн і

підлеглих територій. Визначає три різних коди для кожної країни: двохлітерний, трьохлітерний та числовий. Для України це коди UA, UKR та 804 відповідно;

— та інші.

Дані стандарти включають багато різноманітних рекомендацій для написання мовних тегів або підтегів, наприклад:

— ISO639-1 рекомендує писати коди мов малими літерами («uk» — українська, «mn» — монгольська);

— ISO15924 рекомендує, щоб коди писемностей були написані малими літерами, що починаються з великої літери, тобто «Cyrl» — кирилиця, «Latn» — латиниця, «Phag» — монгольське квадратне письмо;

— ISO3166-1 рекомендує, щоб коди країн були написані великими літерами («UA» — Україна, «MN» — Монголія).

Таким чином можна сказати, що мовний ідентифікатор «mn-Cyrl-MN» є вірним за вказаними стандартами та сприйматиметься найкраще.

1.6.3 Дерево фолбеків

Часто мовні теги можна зустріти у URL адресах веб-сторінок, наприклад «uk.wikipedia.org», «www.microsoft.com/en-gb». Вони вказують на те, якою мовою відображається зміст веб-сайту. Таким чином, при пошуку певної інформації у пошуковій системі, можна одразу звертати увагу на мовні теги знайдених варіантів і, за їх допомогою, відбирати найзручніший для читання варіант.

На жаль, власники веб-додатків, не завжди мають змогу задовольнити усі потреби відвідувачів стосовно потрібної їм мови, тому пропонуватимуть варіанти своїх сторінок найбільш близькою до зручної мови. Щоб знайти подібні мови використовуються дерева фолбеків (див. рис. 1.3).

У таких деревах фолбеків мови розташовуються від найбільш загального варіанту використання мови, до найбільш вузьких її варіантів. Таким чином якщо власник веб-сторінки немає контенту, який написаний китайською мовою, спрощеною писемністю, що вживається на території Сінгапуру, то він запропонує користувачу лише спрощену писемність китайської мови без залежності від

території.

1.7 Машинний переклад та його недоліки

Перед тим як локалізувати додаток або веб-сайт, перед розробником постає вибір яким чином отримати локалізований контент: за допомогою людей, які надають послуги перекладачів, або за допомогою машинного перекладу.

Недоліком першого варіанту є його вартість. Адже один спеціаліст не зможе надати вдалий та коректний переклад на всі запрошені розробником мови. Таким чином доведеться користуватись послугами декількох або навіть і декількома десятками перекладачів, все залежить від того на яку кількість країн та мов має поширюватись додаток. Саме це робить даний варіант дороговартісним.

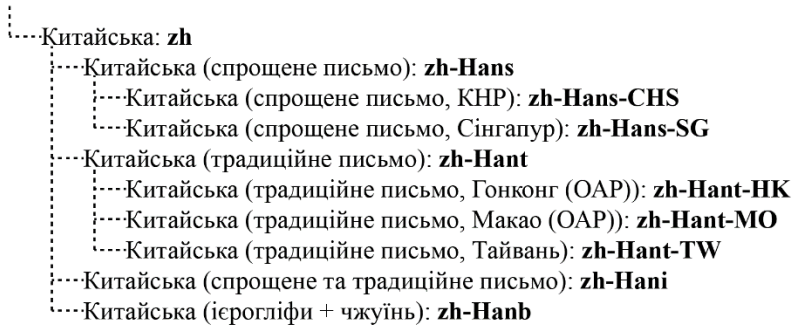

Рисунок 1.3 — Приклад дерева фолбеків

Другий варіант — машинний переклад (Machine translation, MT) — це процес перекладу текстів з однієї мови на іншу за допомогою спеціальної комп’ютерної програми. Зазвичай дані програми поділяють на:

— загальні (до них, зазвичай, відносяться такі платформи, як Google Translate, Bing та Naver, які надають MT для будь-яких можливих перекладів мільйонам людей);

— з можливістю налаштування;

— адаптивні, які поєднують в собі частини загальних платформ та здатність до навчання, як у програм з можливістю налаштування [12].

Програми з можливістю налаштування — це платформи, які мають базові набори словників та правил, і які можна навчити покращувати точність

термінології у вибраному сегменті (медична, юридична, технічна або власна вподобана термінологія). Наприклад, спеціалізований машинний переклад розроблений для Всесвітньої організації інтелектуальної власності (ВОІВ) перекладає патенти точніше, ніж загальні програми МТ, а рішення для eBay (американська інтернет-компанія) може зрозуміти та передати на інші мови сотні скорочень, що використовуються в електронній комерції;

Існує три основних підходи машинного перекладу:

— на основі правил;

— статистичний;

— нейронний.

Підхід на основі правил заснований на незліченних вбудованих лінгвістичних правилах та мільйонах двомовних словників для кожної мовної пари. генерується текст цільовою мовою. Для цього процесу потрібні великі словники з морфологічною, синтаксичною та семантичною інформацією та великі набори правил. Програмне забезпечення використовує ці складні правила, а потім трансформує граматичну структуру вихідної мови на цільову мову;

Статистичний використовує моделі статистичного перекладу, параметри яких випливають із аналізу одномовних та двомовних корпусів. Мовний корпус [13] — це великий, уніфікований, структурований, розмічений та філологічно компетентний масив мовних даних, який представлений в електронному вигляді.

Побудова статистичних моделей перекладу є швидким процесом, але технологія значною мірою спирається на існуючі багатомовні корпуси. Потрібно мінімум 2 мільйони слів для певного сегменту і навіть більше для загальної мови. Теоретично можна досягти певного варіанту гарної якості, але більшість компаній не має такої великої кількості існуючих багатомовних корпусів для побудови необхідних моделей перекладу. Крім того, статистичний машинний переклад є достатньо енергозатратним, тому що вимагає великої кількості процесорів і розширеної конфігурації обладнання для запуску моделей перекладу навіть для середнього рівня продуктивності.

А нейронні МТ (NMT) використовують технологію машинного навчання, щоб навчити програмне забезпечення досягти найкращого результату. Проте очевидним мінусом, як і у попередньому варіанті є енергозатратність, адже цей процес споживає велику обчислювальну потужність.

На сьогоднішній день найбільшою популярністю користуються технології машинного перекладу на основі правил та статистичні, кожна з яких має свої переваги та недоліки (див. табл. 1.1) [14].

Таблиця 1.1 — Порівняльна характеристика підходів машинного перекладу.

Критерії порівняння	МТ на основі правил	Статистична МТ
Якість перекладу неспеціалізованих текстів	Високоякісний	Низькоякісний
Якість перекладу текстів обраної сфери	Послідовна та передбачувана	Непередбачування
Продуктивність	Висока продуктивність та міцність	Високі вимоги до центрального процесора та дискового простору
Стабільність роботи	Відсутність стабільності	Хороша стабільність
Обробка граматичних правил	Присутня	Відсутня
Обробка винятків із правил	Погано справляється з обробкою винятків із правил	Добре відловлює винятки із правил

З огляду на загальні вимоги, які викладені в таблиці 1.1, можна підсумувати, що машинний переклад також не є оптимальним варіантом у виборі інструменту для отримання локалізованого контенту.

1.8 Дослідження існуючих способів реалізації серверної частини додатку

1.8.1 Визначення мови користувача

Початковою точкою глобалізації веб-сайту є процес визначення найбільш зручної для користувача мови. Це можливо зробити за допомогою класу CultureInfo, який надає інформацію про культуру користувача, таку як мова, підмова, країна чи регіон, календар та інше. Основною функцією даного класу є визначення мовних ідентифікаторів у вигляді комбінації підтегів, які були детально розглянути в підрозділі 1.6.2, а саме:

— двобуквеного підтегу стандарту ISO 639 (для відображення мови);

— двобуквеного підтегу стандарту ISO ISO3166, який пов’язаний з назвою країни чи регіону;

— підтегу, який визначає вид писемності (він є не постійним, так як не всі мови мають по кілька різних видів писемності).

.NET отримує інформацію про культуру з безлічі джерел, залежно від реалізації, платформи та версії, наприклад:

— у .NET Framework 3.5 та попередніх версіях вони надаються операційною системою Windows і .NET Framework;

— у .NET Framework 4 та у всіх пізніших версіях — операційною системою Windows;

— у всіх версіях .NET Core, що працюють на платформах Unix — бібліотекою International Components for Unicode (ICU).

ICU — це широко використовуваний набір бібліотек для C/C++ та Java, який забезпечує підтримку Unicode та глобалізації для програмних додатків.

Отримати поточні дані про культуру користувацького університету можна двома способами:

— отримавши значення властивості CultureInfo.CurrentCulture;

— отримавши значення властивості Thread.CurrentThread.CurrentCulture.

Приклад наведено нижче.

```
using System;
```

```
using System.Globalization;
using System.Threading;

public class Example
{
  public static void Main()
  {
    CultureInfo culture1 = CultureInfo.CurrentCulture;
    CultureInfo culture2 = Thread.CurrentThread.CurrentCulture;
    Console.WriteLine("The two CultureInfo objects are equal: {0}",
                      culture1 == culture2);
  }
}
```

Порівняння значень culture1 та culture2 показує те, що вони рівні, тому доцільно використовувати будь-який із можливих варіантів для визначення культурних особливостей користувача [15].

1.8.2 Огляд платформи .NET Core

.NET Core — це безкоштовна високопродуктивна платформа загального призначення з відкритим кодом, що підтримується корпорацією Майкрософт та націлена на розробників багатьох різних типів додатків [16]. Вона дозволяє:

— створювати веб-додатки та сервіси, програми IoT та мобільні серверні мережі;

— використовувати будь-які засоби для розробки на Windows, macOS та Linux;

— розгортати проект локально або віддалено.

Серед основних характеристик та переваг .NET Core виділяють наступне:

— відкритий код, що дозволяє переглядати, завантажувати або робити свої внески до вихідного коду;

— крос-платформеність (.NET Core працює на операційних системах

Windows, macOS та Linux, кожна з яких виконує код і генерує однакові результати);

— послідовність та стабільність (код виконується з однаковою поведінкою в різних архітектурах);

— широкий спектр програм, адже на платформі .NET Core можна розробляти та запускати різні типи програм, такі як мобільні, десктопні, веб, IoT, мікросервіси, ігри тощо;

— підтримка кількох мов, так як дана платформа дає можливість розробляти додатки мовами C#, F# та Visual Basic [17];

— підтримка глобалізації додатків.

Наведені переваги роблять платформу .NET Core вдалим вибором для розробки серверної частини додатку.

1.8.3 Глобалізація в платформі .NET Core

Так як багатомовність веб-додатків дозволяє охоплювати ширшу аудиторію, .NET Core надає сервіси та проміжне програмне забезпечення для їх глобалізації на різні мови та культури.

Як було сказано раніше, глобалізація передбачає інтернаціоналізацію та локалізацію. Для .NET Core перше — це процес розробки програми, яка підтримуватиме різні культури користувачів, а друге — це процес адаптації розробленого інтернаціоналізованого додатку до певної культури та мови і передбачає наступне:

— додавання до програми підтримку локалізації;

— надання локалізованих ресурсів для мов;

— впровадження стратегію для вибору мови.

Для того щоб успішно додати до програми підтримку локалізації були створені інтерфейси IStringLocalizer та IStringLocalizer <T>, що визначені у просторі імен Microsoft.Extensions.Localization.

IStringLocalizer використовує ResourceManager та ResourceReader для надання інформації пов’язаної з культурою. Інтерфейс має індексатор і IEnumerable для повернення локалізованих рядків і, за замовчуванням, вимагає зберігання

мовних рядків у файлах ресурсів.

Наступний приклад показує базовий варіант обгортання рядка «About Title» для локалізації.

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Localization;

namespace Localization.Controllers
{
    [Route("api/[controller]")]
    public class AboutController : Controller
    {
        private readonly IStringLocalizer<AboutController> _localizer;
        public AboutController(IStringLocalizer<AboutController> localizer)
        {
            _localizer = localizer;
        }
        [HttpGet]
        public string Get()
        {
            return _localizer["About Title"];
        }
    }
}
```

Якщо локалізоване значення не знайдено, тоді повертається ключ індексатора, тобто рядок «About Title».

1.8.4 Файли ресурсів

Використання файлів ресурсів є механізмом для надання локалізованих ресурсів та відокремлення рядків, що піддаються локалізації, від власне коду. Це

файли, які містять в собі таблиці із значенням за замовчуванням та його локалізованим варіантом (див. рис. 1.4).

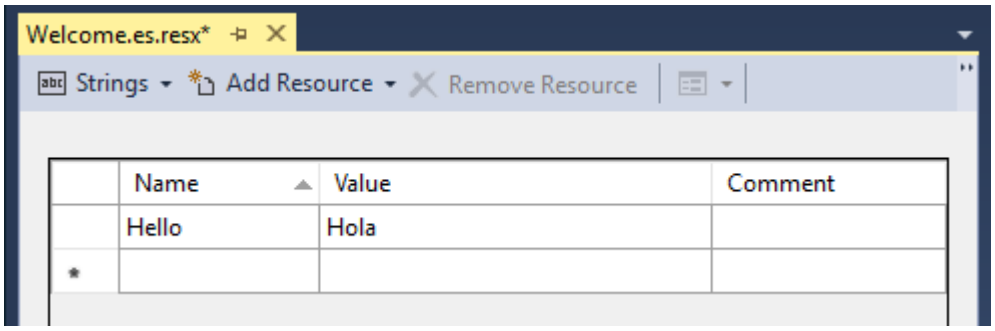

Рисунок 1.4 — Приклад файлу ресурсів

Назви даних файлів мають бути визначеними за певними правилами, адже вони повинні вказувати на мовний ідентифікатор та клас програми, до якого належать. Таким чином, файли ресурсів прийнято іменувати повною назвою класу, від якого відіймається ім’я збірки, а потім додається тег або підтег мови. Наприклад, файл ресурсів для французької мови у проекті, основною збіркою якого є LocalizationWebsite.Web.dll та клас LocalizationWebsite.Web.Startup, буде називатися Startup.fr.resx. [18]

2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Проектування архітектури додатку

Архітектура додатку описує шаблони та методи, які використовуються для розробки та побудови програми. Вона дає карту того, як окремі частини додатку збираються разом та взаємодіють між собою, а також найкращі практики, яких слід дотримуватися під час розробки, щоб забезпечити масштабованість, структурованість та надійність розробки і можливість легко виявляти прогалини у функціональності.

Існує багато різних типів архітектур додатків, але найбільш відомими на сьогодні є: монолітна, мікросервісна та клієнт-серверна. Перша це традиційна модель проектування програмного забезпечення, з назви якої можна зрозуміти, що вона сказається з однієї великої частини. Монолітне програмне забезпечення розроблено для автономності, усі компоненти програми взаємопов’язані та взаємозалежні. Але якщо будь-який компонент програми потрібно оновити, скоріше за все доведеться переписувати більшу частину коду, що робить дану архітектуру не найкращим варіантом при виборі.

Архітектура мікросервісів — це архітектура, при якій частини додатку розбиваються на найдрібніші компоненти, які не залежать один від одного. Це дозволяє уникнути переписування всього коду, як при монолітній архітектурі, але створює додаткову складність при впровадженні механізмів для комунікації між даними сервісами, при розгортанні та управлінні системою загалом, а також такий підхід збільшує споживання пам’яті.

Клієнт-серверна архітектура — це така архітектура, при якій сервер розміщує, постачає та управляє більшістю ресурсів та послуг, що споживаються клієнтом. Вона підтримує один або кілька клієнтських комп’ютерів, підключених до центрального сервера через мережу (див. рис. 2.1) і працює за принципом розділення завдань між провайдерами послуги, які називаються серверами, і запитувачами послуг, що називаються клієнтами. Таким чином всі розрахунки виконуються на сервері, що збільшує продуктивність в десятки і сотні разів та є головним фактором доцільності розробки систем за допомогою даної архітектури.

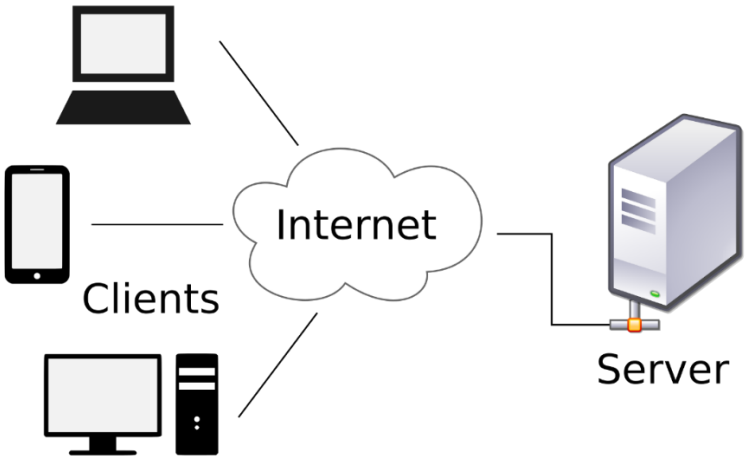

Рисунок 2.1 — Клієнт-серверна архітектура

2.2 Вибір інструментів для розробки

Щоб реалізувати веб-сервер для клієнт-серверної архітектури, можна скористатись різними серверними веб-фреймворками. Вони спрощуються розробку, подальшу підтримку і масштабування веб-додатків, а також надають інструменти для полегшення роботи з маршрутизацією, взаємодією з базою даних, підтримки авторизації користувачів та покращення захисту від веб-атак.

Серед найбільш популярних фреймворків наразі є:

— Django (Python);

— Express (Node.js/JavaScript);

— ASP.NET Core

Django є повноцінною системою, що має в комплекті багато речей, які можна використовувати чи не використовувати залежно від потреб програми. Таким чином замість того, щоб писати власний код для аутентифікації користувачів, управління сесіями та сповіщеннями і подібне, потрібно лише імпортувати уже створені для цього пакети. Проте серед вагомих мінусів даного фреймворку є монолітна архітектура його внутрішніх модулів та надмірність для невеликих проектів [19].

Express — це швидкий, гнучкий та мінімалістичний веб-інтерфейс, який забезпечує надійний набір функцій для веб і мобільних додатків та надає корисні HTTP-утиліти і middleware. Серед його плюсів є можливість легко почати роботу з

фреймворком (тому що JavaScript є простою для вивчення мовою) та швидко масштабувати додаток [20]. Проте використання Express не забезпечує безпеку додатку і надає не завжди релевантні повідомлення про помилки.

Однією з найкращих речей ASP.NET Core є те, що він базується на об'єктно-орієнтованому програмуванні, що дозволяє ділити проект на мілкі фрагменти, які у подальшому можна поєднувати та ефективніше управляти ними. Ще одним плюсом є система кешування, яка робить надійним та простим тимчасове зберігання даних. Також серед переваг є можливість легко розгортати та обслуговувати додаток та розробляти його кросплатформенним.

Зважаючи на переваги фреймворку ASP.NET Core, що були згадані у списку вище та у розділі 1.8.2, він є найкращим варіантом для реалізації веб-сервера.

2.3 Проектування архітектури серверу

Для забезпечення нормальної розробки серверу варто обрати архітектуру та потім слідувати їй. Найвдалішим варіантом для даного додатку є тришарова архітектура (див. рис. 2.2).

Вона не ділить серверну частина додатку на незалежні по реалізації частини, а лише абстрактно відокремлює три логічні шари. Кожен з них має власне призначення, яке існує лише в межах цього шару і є повністю ізольованим від інших.

І найпоширенішому варіанті реалізації такої архітектури, шари мають наступні назви: проміжний (Intermediate Layer), шар логіки (Logic Layer) та шар доступу до даних (Data Access Layer). Останній містить в собі усі необхідні інструменти для роботи з різноманітними базами даних (наприклад, як MongoDB, MySQL та Microsoft SQL Server), а саме моделі БД, репозиторії реляційних та нереляційних баз даних, контексти даних даних EF Core та інше.

Для обробки всієї інформації та виконання обчислень існує шар логіки. Цей рівень також називають середнім, логічним або бізнес-логічним. У загальному він керує основною функціональністю програми.

Intermediate Layer слугує так званою проміжною ланком між клієнтською

частиною додатку та серверною. Саме цей шар отримує дані з клієнта, аналізує їх та передає на подальшу обробку до шару логіки, а також після виконання всіх необхідних функцій, повертає уже оброблені дані назад на клієнт.

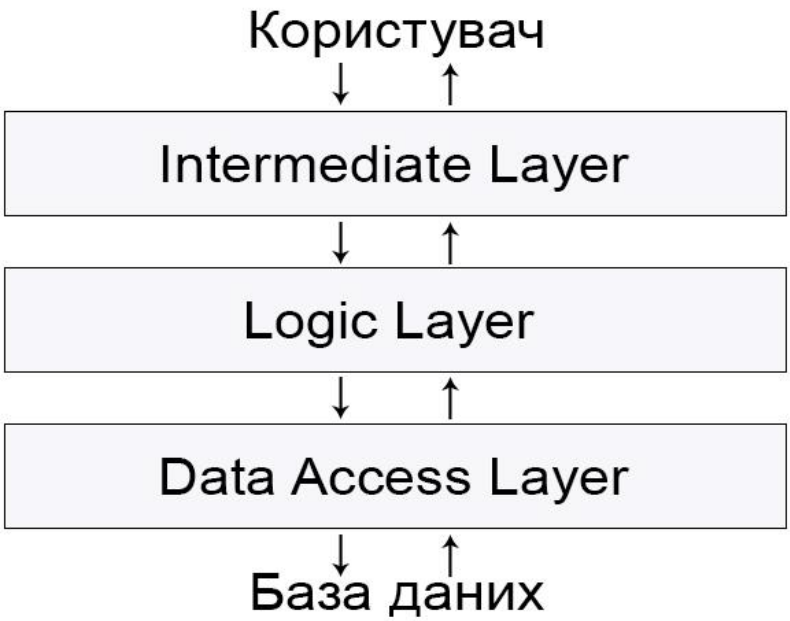

Рисунок 2.2 — Тришарова архітектура серверу

2.4 Налаштування додатку

При створенні нового веб-додатку ASP.NET Core у середовищі Visual Studio, остання автоматично генерує файл appsettings.json (див. рис. 2.3).

Рисунок 2.3 — Автоматично згенеровані файли при створенні нового веб-додатку

Він є файлом конфігурації програми, який використовується для зберігання параметрів конфігурації, таких як рядки підключень до бази даних, ключі для авторизації у інших незалежних зовнішніх сервісах, будь-які глобальні змінні області застосування, тощо [21].

Для даного проекту файл appsettings.json має наступний вигляд:

```
{
 "Logging": {
  "LogLevel": {
   Default": "Warning"
  }
 },
 "MongoConnection": {
  "MongoConnectionString": "mongodb://localhost:27017",
  "Database": "PolyglotDB"
 },
 "AllowedHosts": "*",
 "ConnectionStrings": {
  "PolyglotDatabase": "Data Source=.\\SQLEXPRESS;Initial Catalog=Polyglot; IntegratedSecurity=True; ConnectTimeout=30; Encrypt=False; TrustServerCertificate =True; ApplicationIntent=ReadWrite;MultiSubnetFailover=False",
  "PolyglotStorage": "UseDevelopmentStorage=true"
 },
 "Firebase": {
  "ProjectId": "polyglot-dbc9a"
 },
 "Azure": {
  "SignalR": {
   "ConnectionString": ""
  }
 },
```

```
 "elasticsearch": {
  "url": "http://localhost:9200/",
  "index": "polyglot-test",
  "updateIndex": true
 },
 "UseLocalSignalR": false
}
```

Для того, щоб мати можливість застосовувати особливі налаштування при різних умовах ASP.NET Core підтримує створення кількох файлів конфігурацій. Назва для таких файлів генерується таким чином: appsettings.<середовище застосування>.json, наприклад appsettings.Development.json для середовища розробки. Зміст файлу appsettings.Development.json:

```
{
 "Logging": {
  "LogLevel": {
   "Default": "Debug",
   "System": "Information",
   "Microsoft": "Information"
  }
 },
 "ConnectionStrings": {
  "PolyglotStorage": "UseDevelopmentStorage=true",
  "PolyglotDatabase": "Data Source=DESKTOP-HM5HN5D; Initial Catalog=Polyglot; Integrated Security=True; Connect Timeout=30; Encrypt=False; TrustServerCertificate=True;ApplicationIntent=ReadWrite;MultiSubnetFailover=False"
 },
 "elasticsearch": {
  "url":
"https://elastic:p56iamsUdFsNDn4JMooOlH94@802bb6d77d8244c482aa8ee1f1f4d321
```

```
.eu-central-1.aws.cloud.es.io:9243/",
    "index": "polyglot-test",
    "updateIndex": false
  },
  "UseLocalSignalR": true
}
```

Щоб отримати доступ до інформації про конфігурацію всередині додатку, потрібно скористатися інтерфейсом IConfiguration, яку надає ASP.NET Core Framework. Наприклад:

```
public IConfiguration Configuration { get; }
public void ConfigureServices(IServiceCollection services)
{
  // ...
  services.AddFirebaseAuthentication
  (Configuration.GetValue<string>("Firebase:ProjectId"));
  // ...
}
```

2.5 Розробка серверної частини додатку

2.5.1 Проміжний шар

Як було сказано раніше, даний шар є своєрідною проміжною ланкою між клієнтською частиною додатку та серверною. Його основна ціль це прийом різноманітних запитів від клієнта, тому він складається з контролерів які слугують для авторизації клієнта та передачі даних запитів далі до шару логіки.

У фреймворку ASP.NET Core усі необхідні контролери повинні наслідуватись від ControllerBase.

```
[Produces("application/json")]
[Authorize]
[Route("[controller]")]
```

```
[ApiController]
public class UserProfilesController : ControllerBase
```

Цей код показує, що контролер повертає дані у форматі JSON, вимагає обов’язкової авторизації та є доступним за адресою, яка одночасно є назвою контролера. Повну файл реалізації контролера наведено у додатку Б.

Додаючи до контролера HTTP методи можна перевизначити його налаштування в залежності від власних потреб, наприклад:

```
// GET: UserProfiles
    [HttpGet("user")]
    public async Task<IActionResult> GetUserByUid()
    {
       var user = await service.GetByUidAsync();
       return user == null ? NotFound($"User not found!") as IActionResult
          : Ok(user);
    }

    // GET: UserProfiles/5
    [HttpGet("{id}", Name = "Get")]
    public async Task<IActionResult> Get(int id)
    {
       var entity = await service.GetOneAsync(id);
       return entity == null ? NotFound($"Translator with id = {id} not found!")
as IActionResult
            : Ok(entity);
    }
```

2.5.2 Шар логіки додатку

Уся логіка веб-додатку зосереджена у даному шарі.

У випадку коли веб-сайту необхідне зчитування даних, шар логіки отримує дані з шару доступу та передає їх до проміжного, а у випадку, які потрібно внести

певно зміни, бере дані з проміжного шару, за потреби обробляє їх та модифікує і передає до шару роботи з даними.

Найчастіші методи, які будуть використовуватись усіма сутностями, доцільно декларує в окремому інтерфейс:

```
public interface ICRUDService <TEntity, TEntityDTO>
  where TEntity : Entity, new()
  where TEntityDTO : class, new()
  {
    Task<IEnumerable<TEntityDTO>> GetListAsync();
    Task<TEntityDTO> GetOneAsync(int identifier);
    Task<TEntityDTO> PutAsync(TEntityDTO entity);
    Task<bool> TryDeleteAsync(int identifier);
    Task<bool> TryAddAsync(int identifier);
    Task<TEntityDTO> PostAsync(TEntityDTO entity);
  }
```

Повну реалізацію даного інтерфейсу наведено у додатку В.

Так як основною задачею додатку є локалізація та переклад, то важливою функцією є робота з машинним перекладом. На сьогодні найпопулярнішим, найшвидшим та найбільш надійним є Google Translate API.

Для роботи з ним створено модель, яка визначає дві мови, ту з якої буде здійснено переклад, та ту, на яку він відбувається, а також власне текст, який потрібно перекласти:

```
public class TextForTranslation
{
  [Required]
  [JsonProperty(PropertyName = "q")]
  public string Text { get; set; }
  [Required]
  [JsonProperty(PropertyName = "target")]
```

```
  public string TargetLanguageCode { get; set; }
  [JsonProperty(PropertyName = "source")]
  public string SourceLanguageCode { get; set; }
 }
```

Завдяки можливості Google Translate автоматично визначати мову, з якої здійснюється переклад, це поле не є обов’язковим.

Для того щоб формувати запити до даного API, було розроблено простий HTTP клієнт у вигляді сервісу з методом Translate:

```
 public async Task<string> Translate(TextForTranslation item)
 {
  var path = _url + "?key=" + _key;
  HttpWebRequest httpRequest = WebRequest.CreateHttp(path);
  httpRequest.Method = "POST";
  httpRequest.ContentType = "application/json";
  byte[] byteArray = Encoding.UTF8.GetBytes(JsonConvert
.SerializeObject(item));
  httpRequest.ContentLength = byteArray.Length;
  using (var streamWriter = httpRequest.GetRequestStream())
  {
   await streamWriter.WriteAsync(byteArray, 0, byteArray.Length);
  }
  using (var response = (HttpWebResponse) await
httpRequest.GetResponseAsync())
  using (var stream = response.GetResponseStream())
  using (var reader = new StreamReader(stream))
  {
   string responseFromServer = await reader.ReadToEndAsync();
   if (response.StatusCode != HttpStatusCode.OK)
   return string.Empty;
```

```
    return JObject.Parse(responseFromServer) ["data"]["translations"].ToString();
  }
}
```

Для забезпечення можливості роботи з веб-додатком у режимі реального часу використовується бібліотека SignalR. Вона обробляє всі підключення клієнтів і може автоматично розсилати сповіщення для всіх доступних наразі клієнтів, які підписані на дану подію (див. рис. 2.4).

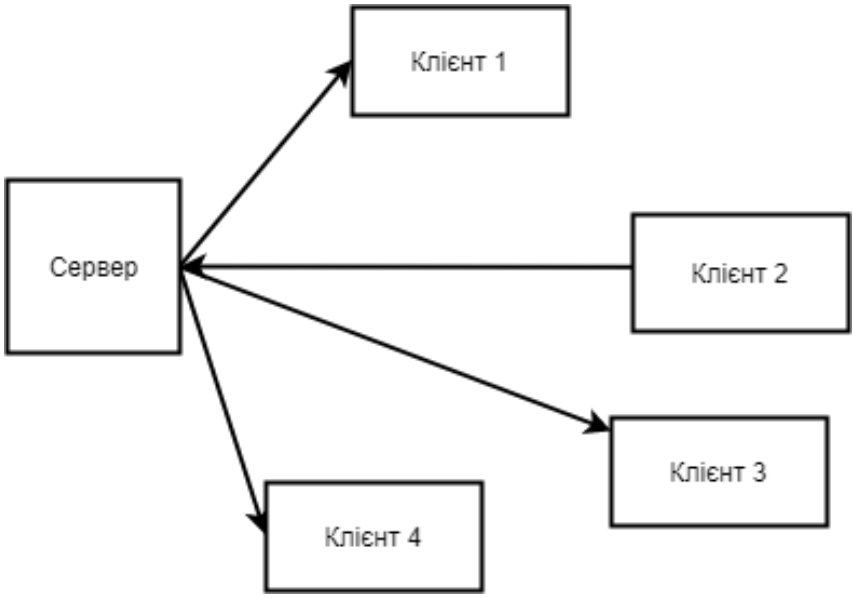

Рисунок 2.4 — Розсилання сповіщень усім підписаним на подію клієнтам

Такий підхід схожий на патерн «Observer», у якому також відбуваються підписки на певні події, але на іншому рівні абстракції. Підключення потребує лише унікального ідентифікатора клієнта та має наступний вигляд:

```
public virtual async Task JoinGroup(string groupName)
{
  await Groups.AddToGroupAsync(Context.ConnectionId, groupName);
}
```

2.5.3 Шар роботи з даними

Для того, щоб зчитувати, оновлювати, видаляти та створювати нові дані за допомогою Entity Framework Core необхідно створити моделі (C# класи), що

співставлятимуть сутності та зв’язки між ними, наприклад:

```
public class UserProfile : Entity
{
  public string FullName { get; set; }
  public string Uid { get; set; }
  public DateTime? BirthDate { get; set; }
  public DateTime RegistrationDate { get; set; }
  public string Country { get; set; }
  public string City { get; set; }
  public string Region { get; set; }
  public string PostalCode { get; set; }
  public string Address { get; set; }
  public string Phone { get; set; }
  public string AvatarUrl { get; set; }
  public Role UserRole { get; set; }
  public virtual ICollection<Rating> Ratings { get; set; }
  public virtual ICollection<TeamTranslator> TeamTranslators { get; set; }
  public virtual List<Project> Projects { get; set; }
}
```

Це модель користувача, у наборі полів якого зберігаються дані про нього та зв’язки. Наприклад:

— ICollection<Rating> Rating вказує на зв’язок один до багатьох із моделлю рейтингу;

— ICollection<TeamTranslator> TeamTranslators — зв’язок багато до багатьох з моделлю TeamTranslators;

— List<Project> Projects — один до багатьох з моделюю проектів.

Після створення усіх моделей, доцільно створити клас контексту даних, який повинен наслідуватись від System.Data.Entity.DbContext. За його допомоги можна відслідковувати зміни, які додали до об’єкту та зв’язувати об’єкти в пам’яті до

елементів користувацького інтерфейсу:

```
public class DataContext : DbContext
{
//…
  public DbSet<Project> Projects { get; set; }
  public DbSet<ProjectHistory> ProjectHistories { get; set; }
  public DbSet<Rating> Ratings { get; set; }
  public DbSet<Tag> Tags { get; set; }
  public DbSet<Team> Teams { get; set; }
//…
}
```

Щоб інкапсулювати логіку роботи із даними можна використовувати патерн «Репозиторій» в поєднання з патерном «Робоча Одиниця». Цей підхід гарантує використання усіма репозиторіями одного і того ж контексту даних, а також спрощує роботу з великою кількістю репозиторіїв (див. рис. 2.5).

Аби мати можливість впоратись з великим навантаженням на БД, її роботу всередині репозиторіїв варто реалізовувати асинхронно. Це також дозволить уникнути «простоювання» під час виконання масивного запиту.

```
public class Repository<TEntity> : IRepository<TEntity> where TEntity : Entity, new()
{
   protected DbContext context;
   protected DbSet<TEntity> DbSet;

   private List<Expression<Func<TEntity, object>>> includeExpressions;

   public Repository(DbContext c)
   {
     this.context = c;
```

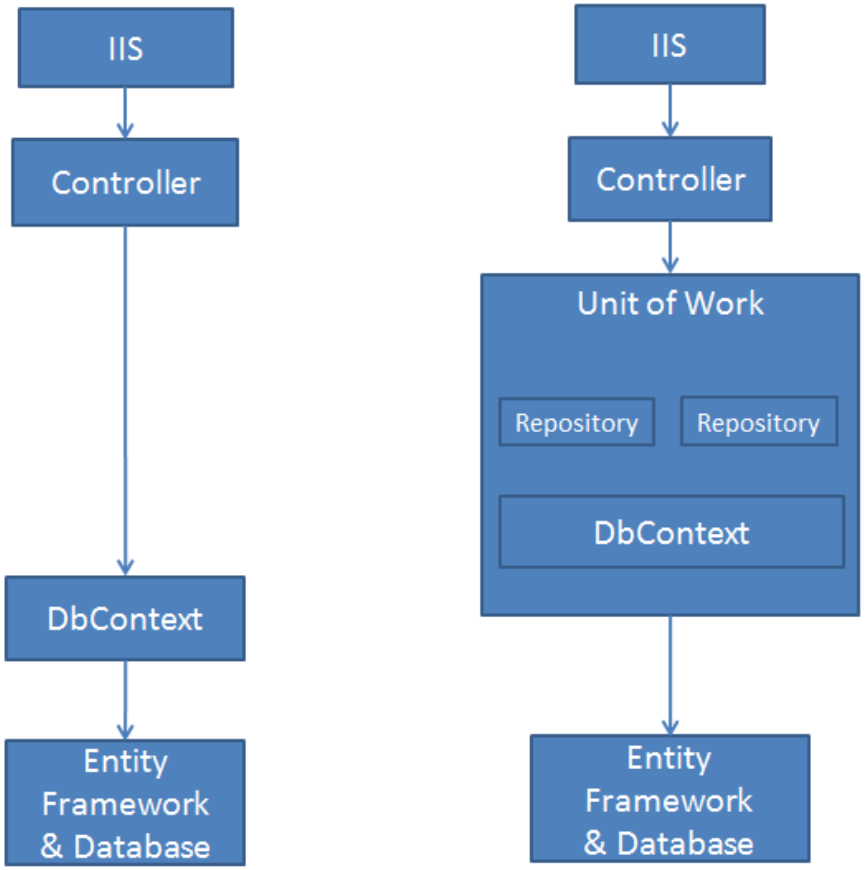

Рисунок 2.5 — Архітектура патернів «Репозиторій» та «Робоча Одиниця»

```
    DbSet = context.Set<TEntity>();
    includeExpressions = new List<Expression<Func<TEntity, object>>>();
  }
  public async Task<TEntity> PutAsync(int id)
  {
    TEntity temp = await DbSet.FindAsync(id);
    if (temp != null)
    {
      var result = DbSet.Update(temp).Entity;
      await Elasticsearch.ElasticRepository.UpdateSearchIndex(temp,
CrudAction.Update);
      return result;
    }
  return null;
```

```
    }

    public async Task<TEntity> GetAsync(int id)
    {
      return await DbSet.FindAsync(id);
    }
  // …
  }
```

Повний лістинг даного класу наведено у додатку Г.

2.6 Розробка чату

Для того, щоб надати можливість перекладачам та менеджерам вільно спілкуватись між собою, було розроблено чат.

Через велику кількість функціональності, яка потрібно при використанні чату, таку як створення діалогу з іншим користувачем, початок переписки, відправлення, отримання, видалення повідомлення та інше, варто створити окремий інтерфейс, який декларує набір даних методів:

```
public interface IChatService
   {
      Task<IEnumerable<ChatDialogDTO>> GetDialogsAsync(ChatGroup targetGroup);
      Task<IEnumerable<ChatMessageDTO>> GetDialogMessagesAsync(ChatGroup targetGroup, int targetGroupDialogId);
      Task<ChatUserStateDTO> GetUserStateAsync(int id);
      Task<ChatDialogDTO> CreateDialog(ChatDialogDTO dialog);
      Task<bool> DeleteDialog(int id);
      Task<ChatMessageDTO> SendMessage(ChatMessageDTO message);
      Task<ChatMessageDTO> GetMessageAsync(int messageId);
      Task<ChatDialogDTO> StartChatWithUser(UserProfileDTO user);
```

```
        Task ReadMessages(int dialogId, string whoUid);
        Task<int> ChangeUserStatus(string targetUserUid, bool isOnline);
        Task<int> GetNumberOfUnreadMessages(int userId);
    }
```

Також було розроблено сервіс, що пронаслідуваний від даної реалізації та описує роботу кожного з методів для чату, наприклад методи для початку чату з іншим користувачем та для видалення чату:

```
    public async Task<ChatDialogDTO> StartChatWithUser(UserProfileDTO user)
        {
            ChatUserDTO chatUser = mapper.Map<ChatUserDTO>(user);
            ChatDialogDTO dialog = new ChatDialogDTO()
            {
                DialogType = ChatGroup.dialog
            };
            List<ChatUserDTO> dp = new List<ChatUserDTO>();
            dp.Add(mapper.Map<ChatUserDTO>(user));
            dialog.Participants = dp;
            return await CreateDialog(dialog);
        }

        public async Task<bool> DeleteDialog(int id)
        {
            var dialogParticipantIds = (await uow.GetRepository<ChatDialog>().GetAsync(id)).DialogParticipants.Select(p => p.ParticipantId).ToList();
            await uow.GetRepository<ChatDialog>().DeleteAsync(id);
            var res = await uow.SaveAsync() > 0;
            foreach (var participantId in dialogParticipantIds)
            {
```

```
                await this.signalRChatService.DialogsChanges($"{ChatGroup.direct
.ToString()}{participantId}", (int)participantId);
                await signalRNavigationService.NumberOfMessagesChanges(
                    $"{Group.notification.ToString()}{participantId}",
                    await this.GetNumberOfUnreadMessages((int)participantId)
                );
            }
            return res;
        }
```

Повний лістинг реалізації чату наведено у додатку Д.

3 РОБОТА ПРОГРАМИ

Перше, що відображається при відкритті веб-сайт Polyglot — це веб-сторінка, яка пропонує відвідувачу зареєструватись в системі, а користувачу, який уже має обліковий запис, увійти до неї. (див. рис. 3.1).

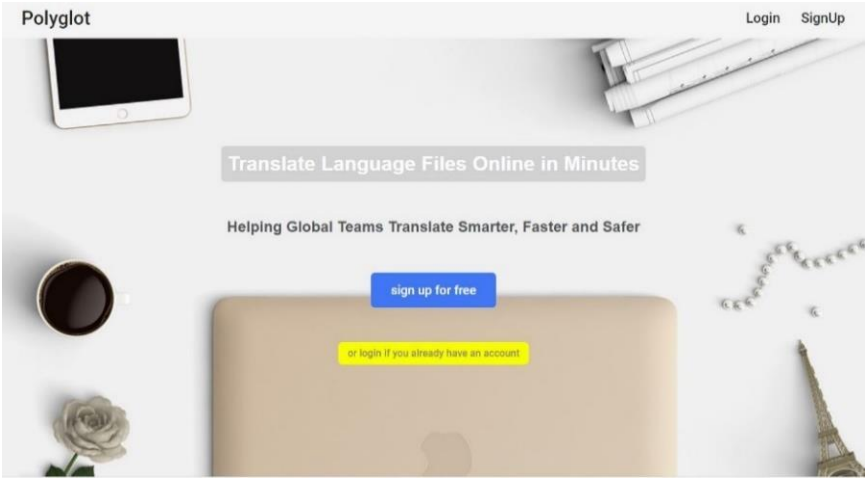

Рисунок 3.1 — Вигляд першої сторінки веб-додатку

Прості та інтуїтивні форми для входу та реєстрації до веб-сайту (див. рис. 3.2) надають можливість власноруч ввести персональні дані або скористатись авторизаціїєю через акаунти Google чи Facebook.

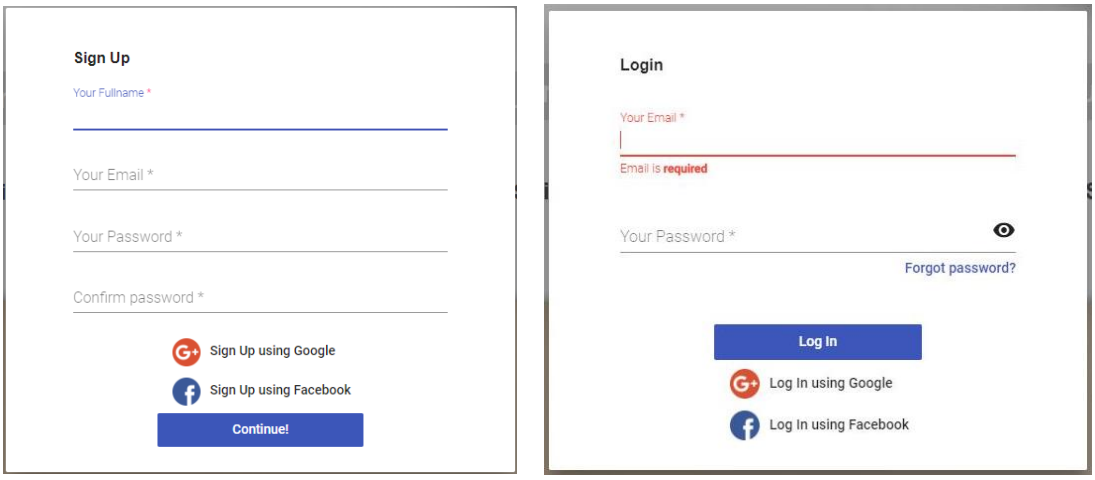

Рисунок 3.2 — Форми для реєстрації та входу до платформи Polyglot

Залежно від ситуації, клік по кнопці «Log In» або «Continue!», пересилає дані, які були введені користувачем, до одного із шарів клієнтської частини додатку, а саме до шару сервісів.

Якщо користувач вперше реєструється на даному веб-сайті, то йому буде запропоновано обрати свою роль: перекладач або менеджер. Завдання першого у локалізації наданого йому контенту на мову, з якою він працює.

У свою чергу менеджер — це або достатньо обізнана людина, яка представлятиме на даній платформі замовника, або власне замовник. Він може створити новий проект, підібрати найкращих та найбільш доцільних для даного проекту спеціалістів. Для полегшення пошуку на особистій сторінці кожного перекладача є своєрідне резюме — коротка інформація про мови, з якими він працює, рівень володіння ними та можливе перебування у складі уже сформованої команди. З іншого боку, для того, щоб перекладачу знати чи погоджуватись на пропозицію менеджера, можна перевірити особисту сторінку останнього, де знаходиться інформація про раніше створені ним проекти, місце роботи та інше.

Повний перелік можливостей перекладача та менеджера наведено у use-case діаграмі на рисунку 3.3.

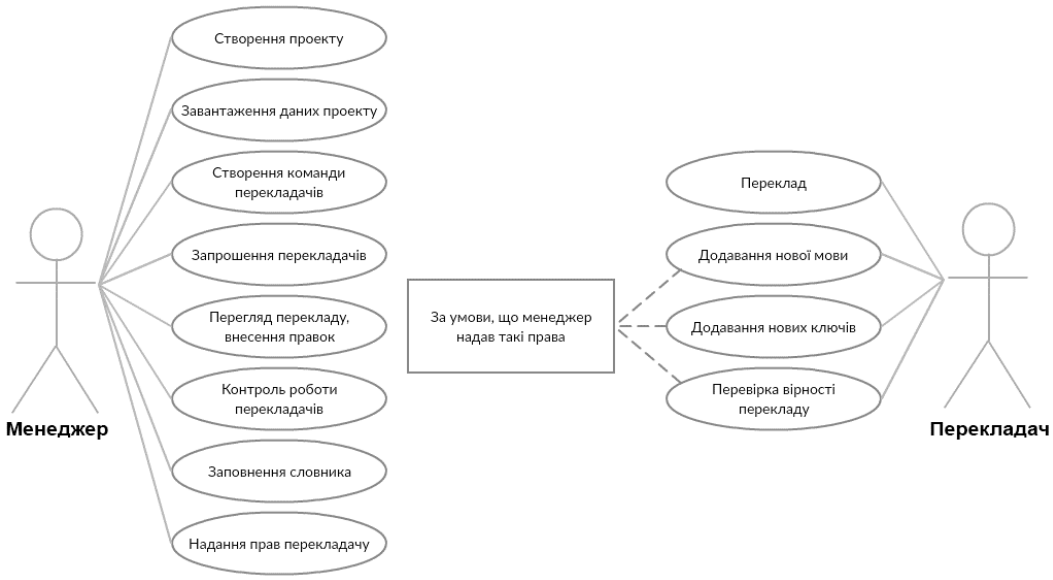

Рисунок 3.3 — Use case діаграма взаємодії користувача з веб-сайтом відповідно до обраної ролі

3.1 Керування для менеджера

Як було сказано раніше, перше, що може зробити менеджер, це створити новий проект. Для цього потрібно натиснути на знак «+» у нижньому правому куті

на вкладці «Projects». У вікні, що з’явилось (див. рис. 3.4), потрібно ввести назву, короткий опис, які передає всю необхідну інформацію, обрати основну мову для даного проекту та технологію, а також є можливість додавання картинки.

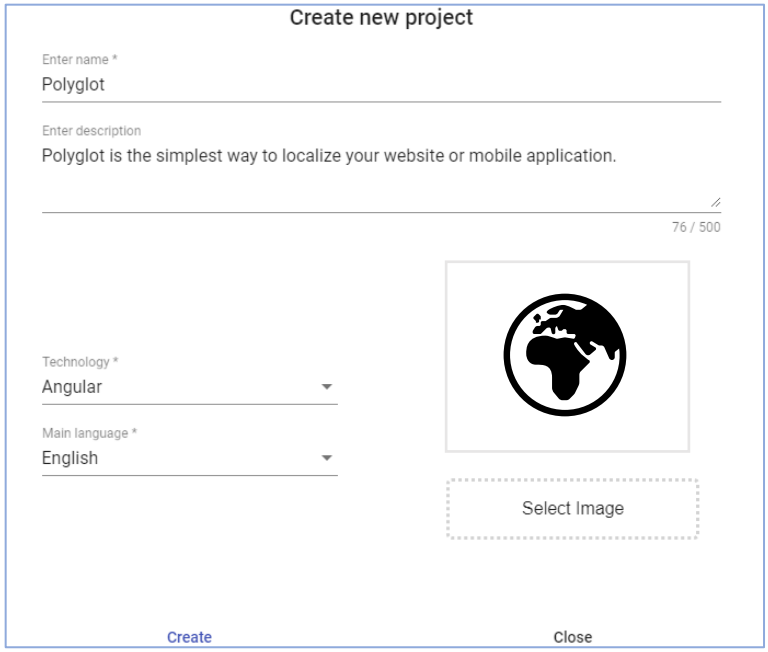

Рисунок 3.4 — Форма для створення проекту

Після заповнення даних, потрібно натиснути кнопку «Create». Тепер і надалі даний проект буде відображатись на вкладці «Projects» (див. рис. 3.5).

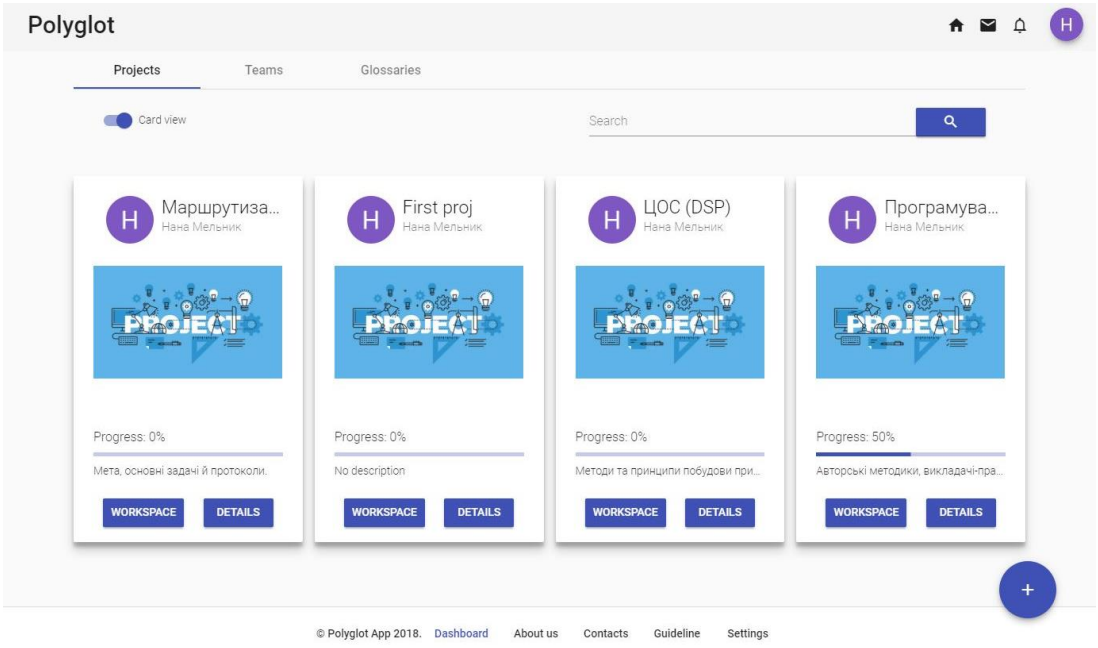

Рисунок 3.5 — Вигляд вкладки «Projects» та створених проектів на ній

Після створення проекту, необхідно почати його заповнення, перейшовши до сторінки «Workspace», натиснувши однойменну кнопку (див. рис. 3.5). Тут менеджер має можливість додавати рядки, які в подальшому будуть перекладатись. Для цього потрібно натиснути кнопку «Add new string» та ввести ключ, рядок оригінальною мовою.

Також додатково можна ввести короткий опис (примітки) до даного рядку та додати тег або картинку (див. рис. 3.6).

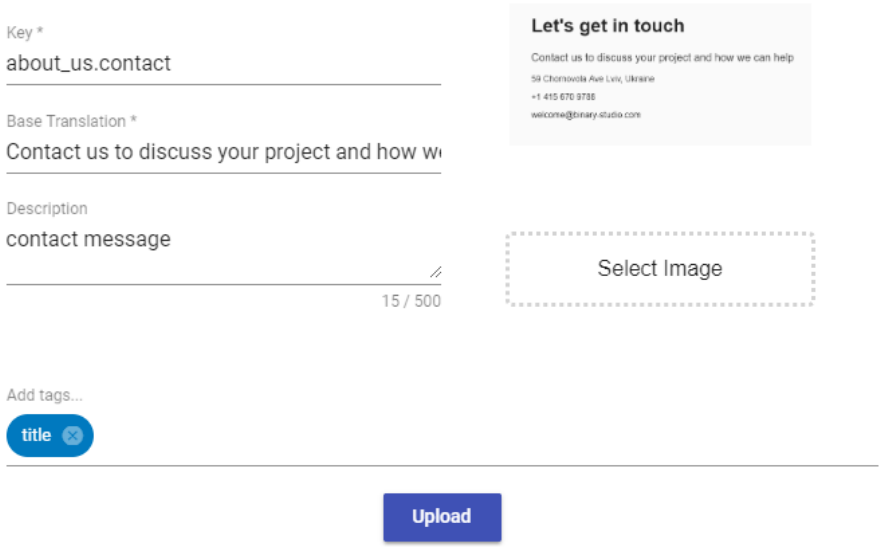

Рисунок 3.6 — Форма для додавання нового рядка

Якщо менеджер має заготований для локалізації текст у файлах з розширенням .resx чи .json, то можна додати його до проекту (даний веб-сайт автоматично розіб'є його на рядки) (додаток Е). Це можна зробити натиснувши кнопку «Details» (див. рис. 3.5) та перейшовши до вкладки «Files» (див. рис. 3.7).

На цій же сторінці в кінці можна буде експортувати локалізований контент у вигляді файлів з розширенням .resx чи .json.

Щоб почати працю над проектом, менеджеру потрібно відібрати перекладачів та зібрати з них колектив. Процес створення команди схожий до процесу створення проекту, лише відбувається на вкладці «Teams». Під час натиснення на кнопку «+» у нижньому правому куті, з'явиться модальне вікно (див. рис. 3.8), де необхідно ввести назву колективу та додати спеціалістів. У додатку Ж приведено алгоритм процесу додавання перекладача до команди.

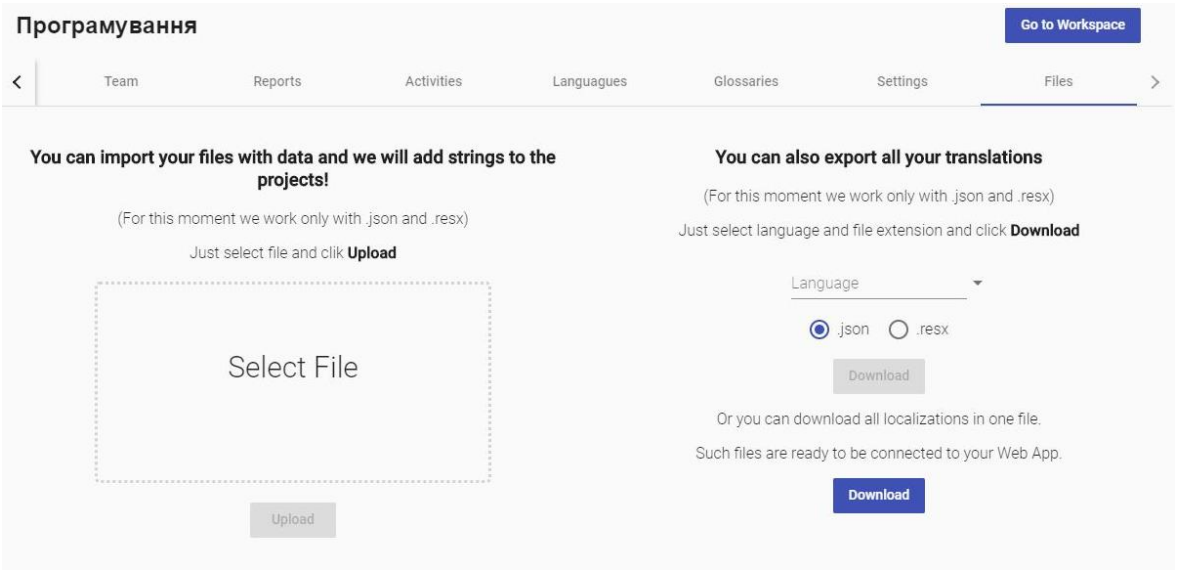

Рисунок 3.7 — Вкладка для імпорту та експорту

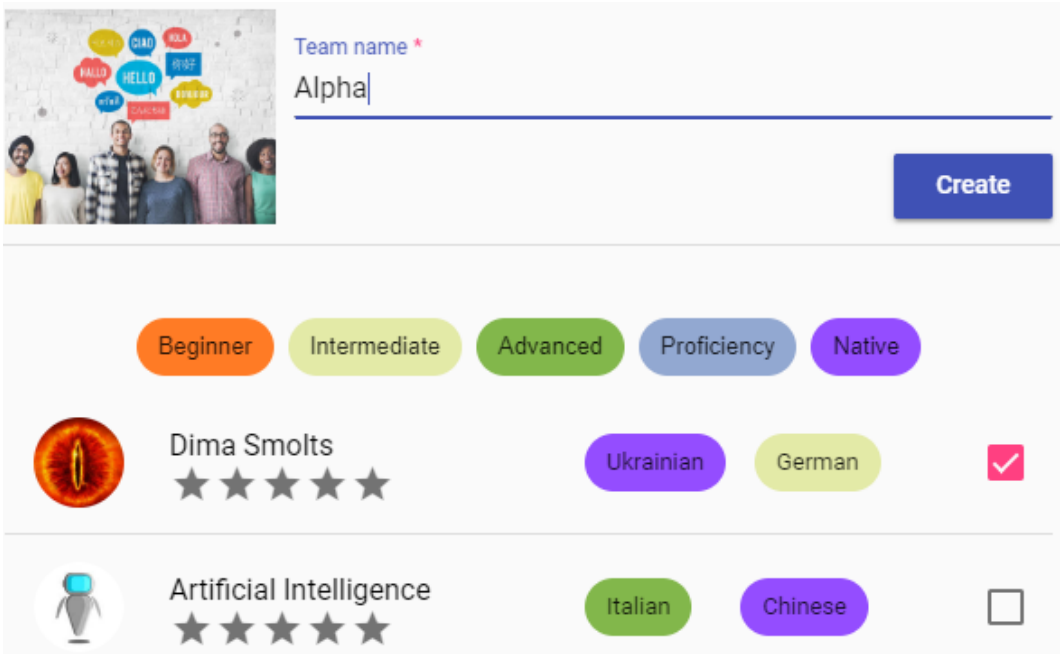

Рисунок 3.8 — Форма створення нової команди

Оскільки перекладачі можуть бути мало обізнаними у спеціалізованому для конкретного проекту контенті, менеджер може полегшити їм працю створивши відповідний словник термінології та абревіатур. Для цього необхідно перейти до вкладки «Glossaries», додати новий словник, клікнувши на кнопку «+», та заповнити його визначеннями (див. рис. 3.9).

Для того, щоб призначити створену команду працювати на проекті, необхідно перейти до вкладки «Team» у вищезгаданій вкладці деталей. А для того,

щоб додати словник потрібно скористатись вкладкою «Glossaries».

Відтепер менеджер має змогу спостерігати за роботою спеціалістів та їх прогресом за допомогою вкладки «Report».

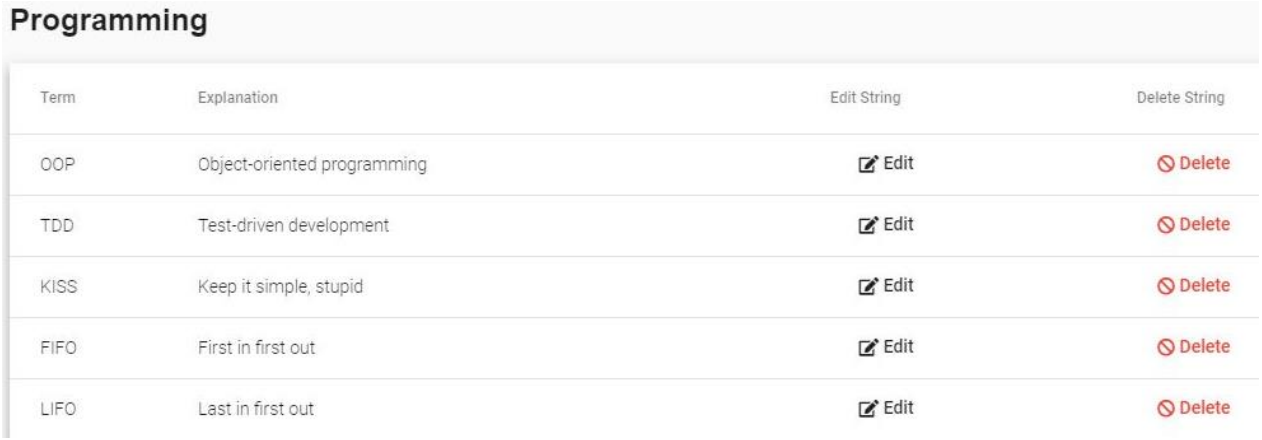

Рисунок 3.9 — Приклад заповненого словника

3.2 Керування для перекладача

Користувач, котрий обрав роль перекладача, має або самостійно знайти проект, куди якого хоче долучитись, або чекати допоки менеджер не вибере його як одного із спеціалістів на свій проект. Останній наведений випадок надає можливість прийняти запрошення до команди або ні.

Головне місце праці для перекладача — це «Workspace», який поділений на три частини (див. рис. 3.10).

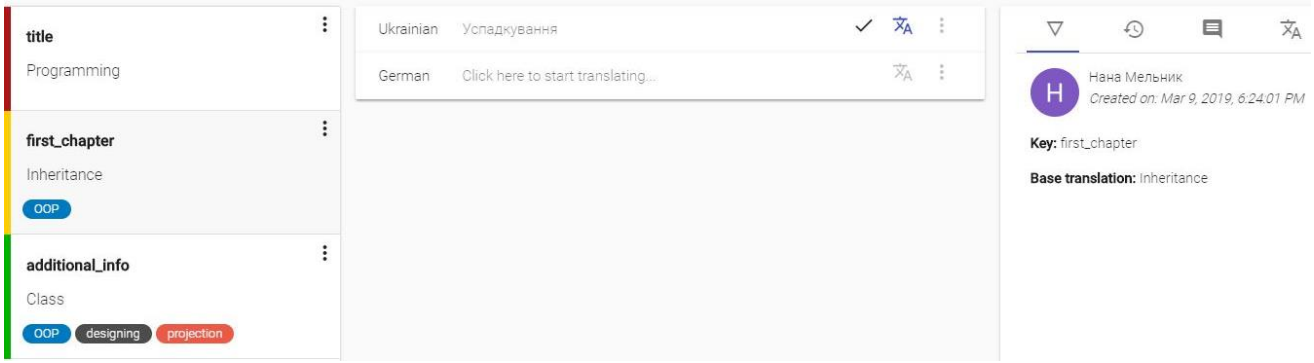

Рисунок 3.10 — Вигляд Workspace

Кожна із них має власну мету, до прикладу:

— ліва, яка містить у собі список всіх рядків;

— середня, котра є місцем для реалізації головної задачі — перекладу обраних рядків та створення їх локалізованих версій;

— права, яка скаладється із декількох вкладок із додатковою інформацією (див. рис. 3.11).

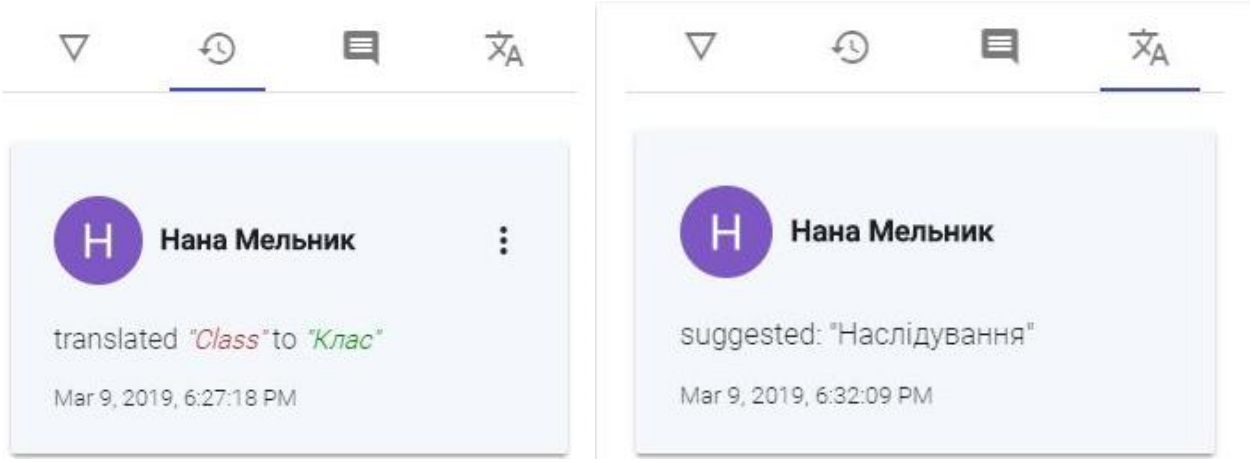

Рисунок 3.11 — Приклад вкладок із додатковою інформація про рядок

У випадку, якщо перекладач натрапить на невідому абревіатуру чи символ, він може використати словник, що завчасно був доданий менеджером. Щоб дізнатись визначення, необхідно почати введення даної абревіатуру до середньої частини Workspace і з’явиться коротка підказка (див. рис. 3.12).

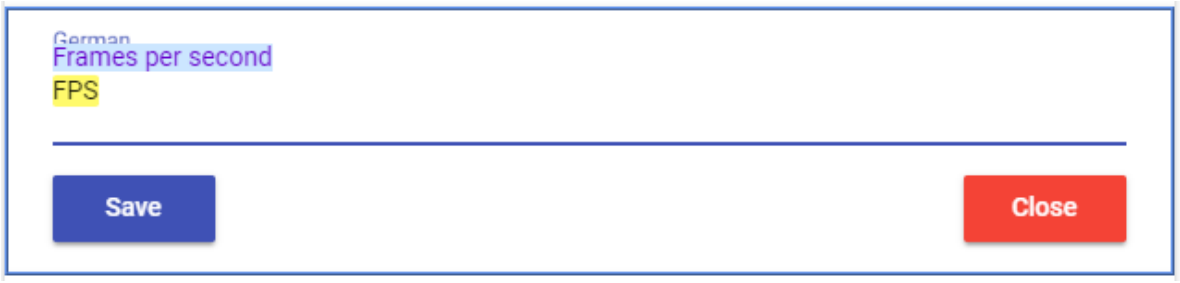

Рисунок 3.12 — Приклад застосування словника

На рисунку 3.13 відображено алгоритм життєвого циклу рядка під час процесу його локалізації.

У додатку И можна розглянути повну схема бази даних, а у додатку К дослідити схему роботи системи.

Рисунок 3.13 — Алгоритм життєвого циклу рядка під час його локалізації

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 3-х незалежних експертів. Такими експертами будуть Черняк О. І., Кадук О. В. та Захарченко С. М.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма рекомендованими критеріями (див. табл. 4.1) за 5-ти бальною шкалою.

Таблиця 4.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-те-Рій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Кінець таблиці 4.1

Бали (за 5-ти бальною шкалою)					
Крите.	0	1	2	3	4
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамі-Кою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної,так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної,так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси.Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали,що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10років	Термін реалізації ідеї більший за 5років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х Років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту,що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації про- дукту вимагає незначних коштів та часу	Необхідно тільки повідомлення від- повідним орга- нам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.2.

Таблиця 4.2 — Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1. Черняк О. І.	2. Кадук О. В.	3. Захарченко С. М.
	Бали, виставлені експертами:		
1	4	4	4
2	3	3	3
3	4	4	3
4	4	4	4
5	4	4	4
6	3	3	4
7	3	3	3
8	4	4	4
9	3	4	4
10	4	4	4
11	4	3	3
12	4	4	4
Сума балів	$СБ_1 = 44$	$СБ_2 = 44$	$СБ_3 = 44$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = 44$		

Отже, з отриманих даних таблиці 4.1 видно, що нова розробка має високий рівень комерційного потенціалу.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторськ-технологічної роботи

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (4.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р = 22$ дні;

t - число днів роботи розробника, t = 30 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 4.3.

Таблиця 4.3 — Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	8 500.00	404.76	6	2 428.57
Інженер-програміст 1	9 000.00	428.57	21	9 000.00
Інженер-програміст 2	9 000.00	428.57	21	9 000.00
Всього:				20 428.57

Додаткова заробітна плата $З_д$ всіх розробників розраховується як 10-12% від суми основної заробітної плати всіх розробників. Розрахуємо додаткову заробітну плату, взявши середнє відсоткове значення:

$$З_{дод} = 0.11 \cdot 20\,428.57 = 2\,247.14 \text{ (грн.)}$$

Нарахування на заробітну плату $Н_{зп}$ для працівників бюджетної сфери становить 22% від суми основної та додаткової заробітної плати:

$$Н_{зп} = (З_о + З_д) \cdot 22\% = (20\,428.57 + 2\,247.14) \cdot 0{,}22 =$$

$$= 4\,988.66 \text{ грн.}$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12},$$

де Ц — балансова вартість обладнання, грн;

$Н_а$ — річна норма амортизаційних відрахувань % (для програмного

забезпечення 25%);

Т — Термін використання (у даному випадку 1 місяць).

Таблиця 4.4 — Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн
Персональний комп’ютер	12 000	25	1	249.00
Ноутбук	9600	25	1	199.20
Всього:				448.20

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_{1}^{n} \text{Н}_\text{і} \cdot \text{Ц}_\text{і} \cdot \text{К}_\text{і},$$

де n — кількість комплектуючих;

$\text{Н}_\text{і}$ — кількість комплектуючих і-го виду;

Ці — покупна ціна комплектуючих і-го виду, грн;

Кі — коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 4.5 — Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флеш-пам’ять	шт.	150	1	150
Блокнот	шт.	45	2	90
Ручка	шт.	15	2	30
Всього з урахуванням транспортних витрат:				297

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В·П·Ф·К_п,$$

де В — вартість 1кВт-години електроенергії (В = 3 грн/кВт);

П — установлена потужність комп'ютера та ноутбука (П = 0,6 кВт);

Ф — фактична кількість годин роботи комп'ютера та ноутбука (Ф = 336 год.);

$К_п$ — коефіцієнт використання потужності ($К_п$< 1, $К_п = 0{,}65$).

$$В_е= 3 \cdot 0.6 \cdot 336 \cdot 0.65 = 393.12 \text{ (грн.)}$$

Інші витрати $В_{ін}$ охоплюють: витрати на управління організацією, оплату службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Їх можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р).$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 \cdot (20\,428.57 + 2\,247.14) = 22\,675.71 \text{ грн.}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + В_{ін}$$

$$В = 20428.57 + 2247.14 + 4988.66 + 448.20 + 297 + 393.12 + 22675.71 =$$

$$= 51\,478.40 \text{ грн.}$$

Розрахуємо загальну вартість наукової роботи $В_{заг}$ формулою:

$$В_{заг} = \frac{В_{ін}}{\boldsymbol{\alpha}}$$

де α — частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{51\ 478.40}{1} = 51\ 478.40$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}$$

де β — коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{51\ 478.40}{0.9} = 57\ 198.22 \text{ (грн)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зрозуміло, що всі зроблені розрахунки будуть приблизними і не передбачають деталізації.

Виконання даної наукової роботи та впровадження її результатів займе один рік. Основні позитивні результати після впровадження результатів виконаної наукової розробки очікуються протягом 3-х років. А одним із основних позитивних результатів впровадження є зростання прибутку.

При впровадження результатів виконаної наукової розробки покращується якість програмного продукту, що дозволяє підвищити ціну реалізації на 1500 грн. Кількість одиниць реалізації програмного засобу також збільшиться: протягом першого року — на 40 шт., протягом другого року — ще на 30 шт., протягом третього року — ще на 20 шт.

Реалізація продукції до впровадження результатів наукової розробки орієнтовно складала 15 шт., а ціна — 4000 грн.

Не можливо прямо оцінити зростання чистого прибутку підприємства від провадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується

отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta Ц_я \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \quad (4.2)$$

де $\Delta Ц_я$ — покращення основного якісного показника від впровадження результатів розробки у даному році;

N — основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN — покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$Ц_o$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n — кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки;

λ — коефіцієнт, який враховує сплату податку на додану вартість (ставка 20%, коефіцієнт λ =0,8333);

ρ — коефіцієнт, який враховує рентабельність продукту;

ϑ — ставка податку на прибуток (18%).

Таким чином, збільшення чистого продукту ΔП1 протягом першого року складатиме:

$$\Delta П_1 = [4000 \cdot 15 + (4000 + 1500) \cdot 40] \cdot 0{,}8333 \cdot 0{,}27 \cdot \left(1 - \frac{18}{100}\right) = 51\,520 \text{ грн}$$

Протягом другого року:

$$\Delta П_2 = [4000 \cdot 15 + (4000 + 1500) \cdot (40 + 30)] \cdot 0{,}8333 \cdot 0{,}27 \cdot \left(1 - \frac{18}{100}\right) = 81\,880 \text{ грн}$$

Протягом третього року:

$$\Delta П_3 = [4000 \cdot 15 + (4000 + 1500) \cdot (40 + 30 + 20)] \cdot 0{,}8333 \cdot 0{,}27 \cdot \cdot \left(1 - \frac{18}{100}\right) = 102\,120 \text{ грн}$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Розрахований у підрозділі 4.3 комерційний ефект від можливого впровадження розробок не означає, що розробка буде реально впровадження. Основними показниками, які визначають доцільність фінансування наукової розробки інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності. Розрахунок ефективності вкладених інвестицій передбачає проведення таких робіт:

На першому кроці розраховується теперішня вартість інвестицій PV, що вкладаються в наукову розробку. Такою вартістю можемо вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР, розраховану за формулою, тобто будемо вважати, що ЗВ = PV = 57 198.22 грн.

На другому кроці розраховується очікуване збільшення прибутку ΔПі , що його отримає підприємство (організація) від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження. Таке збільшення прибутку також було розраховане нами, за допомогою формули 4.2.

На третьому кроці будуємо вісь часу, на яку наносимо всі платежі (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів.

Загальні витрати ЗВ на виконання та впровадження результатів наукової роботи (або теперішня вартість інвестицій PV) дорівнює 57 198.22 грн. Результати вкладених у наукову розробку інвестицій почнуть виявлятися протягом трьох років. Вони виявляться у тому, що у першому році підприємство отримає збільшення чистого прибутку на 51 520 грн відносно базового року, у другому році — збільшення чистого прибутку на 81 880 грн (відносно базового року), у третьому році — збільшення чистого прибутку на 102 120 грн (відносно базового року).

Таким чином, рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, наведений на рисунку 4.1.

На четвертому кроці розраховується абсолютна ефективність вкладених інвестицій $E_{абс}$.

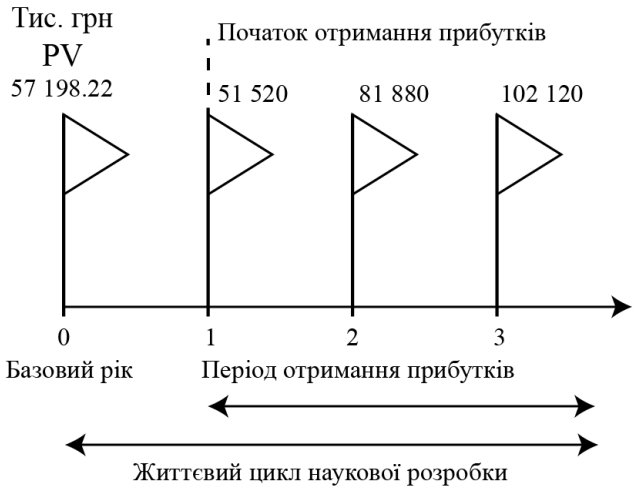

Рисунок 4.1 — Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів наукової роботи

Абсолютна ефективність $E_{абс}$ вкладених інвестицій розраховується за формулою:

$$E_{абс} = (ПП - PV),$$

де ПП — приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.;

PV — теперішня вартість інвестицій PV=ЗВ, грн.

У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{m} \frac{\Delta П_i}{(1 + \tau)^t}$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т — період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t — період часу (в роках) від моменту отримання чистого прибутку до

точки «0».

Розрахуємо вартість чистого прибутку:

$$ПП = \frac{51\,520}{(1+0{,}1)^1} + \frac{81\,880}{(1+0{,}1)^2} + \frac{102\,120}{(1+0{,}1)^3} = 191\,232.05 \text{ грн}$$

Тоді $Е_{абс}$:

$$Е_{абс} = 191\,232.05 - 57\,198.22 = 134\,033.83 \text{ грн.}$$

Оскільки $Е_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР є доцільним.

На п’ятому кроці розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій Ев за формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1$$

де $Е_{абс}$ — абсолютна ефективність вкладених інвестицій, грн;

PV — теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ — життєвий цикл наукової розробки, роки.

Далі, розраховану величина $Е_в$порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар’єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2020 році в Україні d = 0,2;

f — показник, що характеризує ризикованість вкладень, середня величина f = 0,08.

$$\tau = 0{,}2 + 0{,}08 = 0{,}28$$

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{134\,033.83}{57\,198.22}} - 1 = 0.50 \text{ або } 50\%$$

Оскільки $E_в = 50\% > \tau_{мін} = 0{,}28 = 28\%$, то інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

На шостому кроці розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}$$

Якщо $T_{ок}$ буде менше 5 років, то фінансування даної наукової розробки є доцільним.

Для даної розробки термін окупності вкладених у реалізацію проекту інвестицій $T_{ок}$ складе:

$$T_{ок} = \frac{1}{0.50} = 2 \text{ (роки)},$$

Так як $T_{ок}$ < 3…5-ти років, то фінансування даної наукової розробки є доцільним.

Отже, в даному розділі було здійснено оцінювання комерційного потенціалу даної наукової роботи.

Проведено технологічний аудит з залученням трьох незалежних експертів. Аналіз експертних даних показав, що наукова робота має високий рівень комерційного потенціалу.

Згідно із розрахунками всіх статей витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи загальна вартість витрат на розробку і впровадження складає 57 198.22 грн.

Абсолютна ефективність вкладених інвестицій, яка дорівнює 134 033.83, є більшою 0 і показує, що інвестори будуть зацікавлені у нашій розробці.

Відносна ефективність розробки становить 50%, що є досить гарним

показником та вищим за мінімальну ставку дисконтування, тому вкласти гроші у нашу наукову розробку вигідніше, ніж, наприклад, покласти на депозит у банку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій складе 2 роки, що є менше 5-ти і вказує швидку окупність вкладених інвестицій.

Такі показники дозволяють зробити висновок про доцільність розробки та впровадження, а також актуальність фінансування даної наукової розробки.

ВИСНОВКИ

У магістерській дипломній роботі було розроблено серверну частину програмного засобу для локалізації веб та мобільних додатків з метою полегшення процесу розширення цільової аудиторії для власників даних додатків, за рахунок надання зручної місця для локалізації їх інтерфейсу.

Було вирішено задачу створення тришарової архітектури серверної частини додатку, кросплатформеного програмного, а також забезпечено асинхронну систему, яка підготована до великих навантажень та має можливість роботи та перекладу в режимі реального часу.

Реалізовано можливість працювати з реляційною та нереляційною базами даних, забезпечено обробку даних, що надходять, додано можливість надсилання запитів до сторонніх сервісів, зокрема до Google Translate API для підтримки можливості машинного перекладу, а також реалізовано можливість вільного спілкування між різними типами користувачів шляхом розробки чату.

Таким чином поставлена мета була досягнута.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Speak to Global Customers in Their Own Language [електронний ресурс] // Режим доступу: https://hbr.org/2012/08/speak-to-global-customers-in-t — Назва з екрану.

2. Азаров О. Д., Черняк О. І., Смольц Д. О., Мельник Ж. А. «Проблема локалізації веб та мобільних додатків» в Матеріали конференції «XLIX Науково-технічна конференція підрозділів Вінницького національного технічного університету (2020)», Вінниця, 2020. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/allvntu/index/pages/view/zbirn2020.

3. Локалізація в порівнянні з Інтернаціоналізацією [електронний ресурс] // Режим доступу: https://www.w3.org/International/questions/qa-i18n.uk — Назва з екрану.

4. What is localization and why is it so important in 2020? [електронний ресурс] // Режим доступу: https://www.smartcat.com/blog/what-is-localization/ — Назва з екрану.

5. Черняк О. І., Залізецький В.В. «Вирішення проблем локалізації в мультимодульній структурі програмного забезпечення» у X Міжнародній конференції «Молоді вчені 2019–від теорії до практики», Дніпро, 07 березня 2019 р.: 179-181.

6. Top 10 cultural adaptation tips from technology giants [електронний ресурс] // Режим доступу: https://www.deseretnews.com/top/1554/0/Top-10-cultural-adaptation-tips-from-technology-giants.html — Назва з екрану.

7. 7 principles that make your global content localisation stand out [електронний ресурс] // Режим доступу: https://www.brightlines.co.uk/translationservices/localisation/7-principles-that-make-your-global-content-localisation-standout/ — Назва з екрану.

8. Website Localization: 3 Basic Principles [електронний ресурс] // Режим доступу: https://uxpamagazine.org/website-localization/ — Назва з екрану.

9. Lost in Translation: 10 International Marketing Fails [електронний ресурс] // Режим доступу: https://www.businessnewsdaily.com/5241-international-marketing-

fails.html — Назва з екрану.

10. 6 essential steps to planning a successful localization strategy [електронний ресурс] // Режим доступу: https://blog.applingua.com/blog/localization-strategy — Назва з екрану.

11. 5. Черняк О. І., Залізецький В.В. «Вирішення проблем локалізації в мультимодульній структурі програмного забезпечення» у X Міжнародній конференція «Молоді вчені 2019–від теорії до практики», Дніпро, 07 березня 2019 р.: 179-181.

12. Language Technology [електронний ресурс] // Режим доступу: https://www.gala-global.org/knowledge-center/about-the-industry/language-technology — Назва з екрану.

13. Дем'янчук Ю. І. «Різновиди корпусу текстів у процесі перекладу документів офіційно ділового стилю» [Електронний ресурс]. Режим доступу: http://ddpu-filolvisnyk.com.ua/uploads/arkhiv-nomerov/2016/NV_2016_5-1/27.pdf.

14. What is Machine Translation? Rule Based Machine Translation vs. Statistical Machine Translation [електронний ресурс] // Режим доступу: https://www.systransoft.com/systran/translation-technology/what-is-machine-translation/ — Назва з екрану.

15. CultureInfo Class [електронний ресурс] // Режим доступу: https://docs.microsoft.com/en-us/dotnet/api/system.globalization.cultureinfo?view=net-5.0 — Назва з екрану.

16. ASP.NET documentation [електронний ресурс] // Режим доступу: https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-5.0 — Назва з екрану.

17. .NET Core Overview [електронний ресурс] // Режим доступу: https://www.tutorialsteacher.com/core/dotnet-core — Назва з екрану.

18. Globalization and localization in ASP.NET Core [електронний ресурс] // Режим доступу: https://docs.microsoft.com/uk-ua/aspnet/core/fundamentals/localization?view=aspnetcore-5.0#implement-a-strategy-to-select-the-languageculture-for-each-request-2 — Назва з екрану.

19. "Why Django is the Best Web Framework for Your Project" [електронний ресурс] // Режим доступу: https://steelkiwi.com/blog/why-django-best-web-

framework-your-project/ — Назва з екрану.

20. What Are The Benefits Of Using Express.Js For Backend Development [електронний ресурс] // Режим доступу: https://www.techomoro.com/what-are-the-benefits-of-using-express-js-for-backend-development/ — Назва з екрану.

21. ASP.NET Core appsettings.json file [електронний ресурс] // Режим доступу: https://dotnettutorials.net/lesson/asp-net-core-appsettings-json-file/ — Назва з екрану.