МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК, ФІЗИКИ ТА МАТЕМАТИКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

РОЗРОБЛЕННЯ ЛОКАЛІЗАЦІЙНОГО ІНТЕРФЕЙСУ СИСТЕМИ KSU24 УКРАЇНСЬКОЇ, АНГЛІЙСЬКОЇ, НІМЕЦЬКОЇ ТА ІСПАНСЬКОЇ МОВИ

Кваліфікаційна робота
на здобуття другого (магістерського) рівня вищої освіти

Виконала: студентка 241 М групи
Спеціальності 121 Інженерія програмного забезпечення
Освітньо-професійної програми
Інженерія програмного забезпечення
Солонина Тетяна Олександрівна

Керівник доктор педагогічних наук,
кандидат фізико-математичних наук,
професор
Валько Наталія Валеріївна

Рецензент кандидат педагогічних наук,
доцент, докторант,
Мелітопольський державний педагогічний університет імені Богдана Хмельницького, Запоріжжя
Кушнір Наталія Олександрівна

Івано-Франківськ – 2024

ЗМІСТ

ВСТУП

Сучасні дослідження у галузі інженерії програмного забезпечення свідчать про те, що з часом до цих наук інтерес не зменшується, а навпаки тільки збільшується та набуває багато чого нового, тому що, використовує сучасні положення наукової парадигми.

Новими тенденціями розвитку інженерії програмного забезпечення, є вивчення локалізації інтерфейсу, про це свідчать роботи Мігель А. Хіменес-Креспо Локалізація у перекладі, Абдулфаттах О. Оцінка якості локалізації для більш надійних програм електронної комерції арабською мовою, Агуайо Аррабал, Н. та К. Рамірес Дельгадо Аналіз лінгвістичних помилок у веб-локалізації для експорту іграшок в Іспанії, Анастасіу Д. та Р. Шелер Переклад важливої інформації: локалізація, інтернаціоналізація та глобалізація.

Актуальність теми магістерського дослідження зумовлена тенденцією сучасних досліджень у розробці локалізації інтерфейсу програмного забезпечення, адаптації перекладу та системи під різні ринки, додаючи декілька мов до KSU24 система стає доступнішою для світового ринку та відповідно для більшої кількості користувачів.

Зв’язок роботи з науковими програмами, планами, темами. Магістерську роботу виконано в межах науково-дослідної теми кафедри комп’ютерних наук та програмної інженерії Херсонського державного університету «Розроблення системи управління якістю електронних освітніх ресурсів вищих навчальних закладів» (№ держреєстрації 0115U001 128).

Метою дослідження є розробка багатомовної локалізації інтерфейсу для системи KSU24, який забезпечить зручність використання та ефективну взаємодію користувачів, що говорять українською, англійською, німецькою та іспанською мовами.

Досягнення поставленої мети передбачає вирішення низки **завдань:**

- розглянути програмні бібліотеки та фреймворки для підтримки багатомовності;
- дослідити методологію локалізації програмних інтерфейсів;
- окреслити підходи до розроблення локалізаційної архітектури KSU24;
- проаналізувати існуючий інтерфейс KSU24;
- визначити текстові ресурси, що потребують перекладу;
- реалізацувати процес локалізації.

Об'єктом дослідження постає система KSU24 як програмне забезпечення для освітнього процесу, що потребує адаптації для багатомовного використання.

Предметом наукової розвідки є технологічні засоби та методи локалізації інтерфейсу системи KSU24 для підтримки української, англійської, німецької та іспанської мов, а також взаємодія користувача з системою на основі локалізованого інтерфейсу.

Матеріалом для дослідження слугувала система KSU24.

Методологічною основою дослідження є комплексний підхід до локалізації програмного забезпечення, що включає мовознавчі принципи перекладу та адаптації текстів, а також інформаційні технології, пов'язані з розробкою інтерфейсів користувача.

Методика дослідження обумовлена метою та завданнями роботи й ґрунтується на *аналізі* мовних і культурних особливостей української, англійської, німецької та іспанської мов. Метод *синтезу* використовується для поєднання результатів аналізу та інтеграції їх у розробку локалізованого інтерфейсу, що враховує відмінності між мовами. *Метод порівняльного аналізу* застосовується для порівняння мовних структур та культурних особливостей мов, щоб врахувати специфіку перекладу термінів, інтерфейсних елементів та текстів. *Метод*

системного підходу використовується для вивчення системи KSU24 як цілісного програмного продукту, що потребує інтернаціоналізації та локалізації. *Метод проєктування* використовується для створення концептуальної моделі локалізаційного інтерфейсу. Проєктування включає визначення ключових елементів інтерфейсу, які підлягають локалізації (кнопки, меню, повідомлення тощо), а також способи їх адаптації до кожної з мов. *Тестування* локалізованого інтерфейсу для виявлення помилок та проблем у взаємодії користувачів із системою. *Метод статистичного аналізу* застосований для аналізу даних, зібраних під час тестування та опитувань, з метою оцінки якості локалізації та ступеня задоволеності користувачів.

Наукова новизна одержаних результатів полягає у тому, що в дослідженні створено багатомовна локалізація інтерфейсу для освітньої системи KSU24, що поєднує мовні та культурні особливості чотирьох різних мов (української, англійської, німецької та іспанської). Особливо новаторським є застосування інтегрованих рішень для автоматизації локалізаційного процесу та покращення користувацького досвіду у багатомовному середовищі.

Теоретичне значення роботи визначається тим, що її основні положення та висновки слугують певним внеском у розвиток теорії локалізації програмного забезпечення та інтернаціоналізації інформаційних систем, особливо в освітній сфері. Та дослідженні і розробці принципів та методик, що дозволяють адаптувати освітні платформи до багатомовного середовища.

Практичне значення очікуваних наукових результатів полягає в можливості використання робочої багатомовної версії системи KSU24, що дозволить користуватися нею здобувачам та викладачами з різних мовних груп. Це сприятиме підвищенню доступності освітніх ресурсів, що особливо важливо в умовах глобалізації. Результати роботи можуть

бути застосовані для інших освітніх або інформаційних систем, що потребують багатомовної підтримки.

Апробація результатів роботи здійснювалась на засіданнях кафедри комп'ютерних наук та програмної інженерії Херсонського державного університету. Результати кваліфікаційної роботи відображено у науковій статті «Інструменти та етапи локалізації інтерфейсів для підвищення доступності та зручності програмного забезпечення» яку надруковано у збірнику наукових праць студентів ХДУ «Магістерські студії» Вип. XXIV. Івано-Франківськ. ХДУ, 2024. С. 766-769.

Структура та обсяг кваліфікаційної роботи. Робота складається зі вступу, двох розділів, загальних висновків, списку використаних джерел. Список використаних джерел складається з 26 позиції, з них 15 іноземною мовою.

Розділ 1

Теоретичні основи та методологія локалізації програмних інтерфейсів

1.1 Програмні бібліотеки та фреймворки для підтримки багатомовності.

Одним з важливих компонентів програмного забезпечення є використання у ньому багатомовності. Тому, слід визначити та обґрунтувати поняття інтернаціоналізації та локалізації – це процес, за допомогою якого продукт адаптується до певної культури, мови та вимог ринку. Щоб успішно реалізувати продукт у всіх важливих країнах, потрібно не тільки перекладати елементи інтерфейсу на мову покупця, але й звертати увагу на розміщення кнопок та зображення, які можуть зачепити чиюсь релігію, оскільки важливо, щоб значення слів, технологій, розташування кнопок, текстових полів та зображень не викликало негативних асоціацій, продукт не повинен викликати суперечливі емоції у кінцевого споживача. У цій сфері програмного забезпечення вимагаються наступні правила.

Таким чином, локалізація – це процес модифікації програмного продукту для врахування мови та культурних особливостей користувача [4]. За часту локалізацію позначають як l10n, де 10 вказує на кількість букв між буквами l та n.

Дане поняття може включати адаптацію слідуючих аспектів:

- Переклад користувацького інтерфейсу.
- Переклад документації.
- Цифрові формати а саме: формати дат та часові формати.
- Використання грошових одиниць.
- Використання клавіатури.
- Комплектування та сортування.

- Символи, знаки і кольори.
- Об'єкти, дії або ідеї, які в даному культурному середовищі можуть бути неправильно сприйнятими чи не зрозумілими.
- Різні правові вимоги.

Крім цього, у певних випадках підвищення уваги до дотримання вимог локалізації може впливати на безпосередньо проектування додатку [7].

Також, виокремлюють таке поняття, як локалізація контенту яка охоплює більше, ніж просто переклад тексту; вона передбачає адаптацію контенту відповідно до культурних і мовних уподобань цільової аудиторії. Також передбачає коригування контенту відповідно до культурних і мовних особливостей певного регіону чи цільової аудиторії. Основна мета локалізації контенту полягає в тому, щоб підвищити залученість користувачів, релевантність і зручність використання, що призведе до підвищення коефіцієнтів конверсії та задоволеності клієнтів. На відміну від прямого перекладу, локалізація контенту враховує кілька факторів, зокрема регіональні розмовні мови, культурні посилання та візуальні елементи [3].

На сьогоднішній день розробка програмного забезпечення для мови, відмінної від мови його авторів, зосереджена на адаптації інтерфейсу користувача. Цей метод називається інтернаціоналізацією програмного забезпечення. В основі інтернаціоналізації програмного забезпечення лежить припущення, що всі культурно та лінгвістично чутливі програмні компоненти можуть бути відокремлені від локально-незалежного ядра програми [19, с. 87].

Отдже, інтернаціоналізація – це проектування та розробка програмного продукту або документації з метою спрощення подальшої локалізації [4].

Зазвичай, інтернаціоналізацію зазначають як i18n, де 18 вказує на кількість букв між і та n в англійському варіанті слова.

Інтернаціоналізація зазвичай включає в себе: створення та розробку з урахуванням можливості легкої локалізації та глобального застосування. Це включає такі пункти: можливість застосування Unicode або використання правильного підходу до кодування символів (згідно потреб); забезпечення послідовностей рядків.

Також, потрібно забезпечити основу для елементів, які не можуть бути використані до процесу локалізації. Як приклад, в DTD можна додати розмітку для двостороннього тексту або визначення мови. Можна в CSS додати основу для вертикального тексту або нелатинських символів, скориставшись спеціальною властивістю.

Забезпечення підтримки кодування для мов, які є місцевими, регіональними, належать до конкретної культури або включаються у перелік бажаних. Зазвичай це стосується введення локалізованих даних або параметрів, що беруться з існуючих програм пам'яті або налаштувань користувача. До прикладів відносяться: формати дат і часу, місцевими календарями, числовими форматами і системами числення, сортуванням та представленням списків, поводження з особистими іменами і формами поводження.

Завантаження локалізованих елементів з початкового коду або контенту, щоб мати можливість використовувати їх пізніше або обирати на основі переваг користувача [7].

Також, слід розглянути самі поняття бібліотеки та фреймворки, та яка між ними різниця. У свою чергу поняття фреймворк у програмування розуміється як комплекс певних компонентів та бібліотек, інструментів, які пропонують структуру та готові рішення для роботи над завданнями. Іншими словами, фреймворки являють собою готові інструменти для розробки програмного забезпечення. Фреймворки розташовують у собі певний набір шаблонів та коду й інших елементів, що дає змогу зменшити час на написання коду програми та покращити його якість [11].

У свою чергу слід відмітити, що фреймворки бувають різні, в залежності від мови програмування, так, наприклад: для JavaScript це: React, Angular, Vue.js, Next.js, Ember.js. До мови програмування Python належать: Django, Flask, Pyramid, Tornado. Фреймворки, що використовуються для CSS є: Bootstrap, Foundation, Materialize, Bulma, Semantic UI.

Перейдемо до визначення терміну Бібліотека. Цей термін позначає набір функцій, які розв'язують конкретне завдання у програмі. Бібліотека може застосовувати шаблони повідомлень, раніше скомпільований код, класи або підпрограми також, вона може використовуватися багато разів [9].

У програмування є два типи бібліотек – це стандартні та сторонні. Під стандартними бібліотеками розуміють ті бібліотеки, які включають базовий набір функцій для розширення можливостей та вирішення різних завдань Ці бібліотеки надають функціональність для операцій з рядками, математичних операцій та роботи з файловою системою. Під сторонніми бібліотеками у свою чергу бібліотеки, які можуть включати складні алгоритми і додатковий функціонал для роботи з базами даних, мережами. Ці бібліотеки можуть створюватися окремими розробниками або командою розробників і є доступними для завантаження через мережу. Наприклад, серед таких бібліотек можна згадати NumPy, Pandas, TensorFlow і багато інших [1].

Однак із інтерналізацією додає у свою чергу роботи у процесі кодування (а саме: множина, форматування та ін.). Саме тому, низка програм і веб-сайтів покладаються на фреймворки, які мають усі складні функції, необхідні для їх інтернаціоналізації. Бібліотеки та фреймворки потрібно використовувати оскільки вони зазвичай полегшують розробникам інтеграцію певних функцій у своє програмне забезпечення, програми чи веб-сайти. Для інтерналізації життєво важливо шукати фреймворки, які підтримують такі функції:

- Множинність. Підтримка множини означає, що ви можете змінювати частини рядка відповідно до числа, переданого як параметр. Наприклад:

```
{
 "key_one": "{count} item", // `count` is the parameter.
 "key_other": "{count} items"
}
```

- Інтерполяція. Підтримка інтерполяції означає, що ви можете передавати динамічні значення в рядки. Наприклад, у наведеному вище коді *count* приймає динамічне значення, яким може бути будь-яке число.
- Контекст. Підтримка контексту дозволяє передавати додаткову інформацію для використання під час перекладу рядків. Наприклад:

```
{
      "dessert": "I like all desserts",
      "dessert_cake": "I like to eat cake",
      "dessert_muffin": "I like to eat muffin"
}
```

- Форматування. Форматування є важливим, оскільки воно забезпечує правильне та зрозуміле відображення даних відповідно до їх контексту.

Що стосується безпосередньо завантаження перекладів I18Next використовує ряд плагінів для завантаження перекладів. Вони дозволяють завантажувати переклади з:

- Сервери;
- файлові системи;
- об’єднання перекладів із Webpack.

Те, що відноситься Format.js і Polyglot, у такому випадку розробники повинні написати код, щоб завантажити або об’єднати переклади у свої веб-програми [9].

Отже, у програмному забезпечені KSU24 використовується бібліотека i18next, що має підтримку фреймворку React. У свою чергу бібліотека i18next – це не просто бібліотека, яка виконує переклади

тексту. i18next – це фреймворк, який дає змогу розгорнути повноцінне рішення для локалізації програмного забезпечення застосунку.

Оскільки на KSU24 використовується фреймворк React. Сама бібліотека i18next має гарну систему плагінів і конекторів для платформ. Враховуючи той факт, що програмне забезпечення будується у React, то у такому випадку потрібно працювати з двома основними бібліотеками:

- i18next – це платформа для локалізації;
- react-i18next – це набір біндингів (bindings) для React застосунку [6].

Таким чином, слід визначити, що для створення локалізації додатку слід розрізняти самі поняття як локалізації, інтернаціоналізації та глобалізації, а також обрати необхідний фреймворк за допомогою якого буде створюватися додаток та бібліотеку, для спрощення процесу локалізації.

1.2 Методологія локалізації програмних інтерфейсів.

Локалізація програмного продукту означає адаптацію програми до вимог, стандартів і правил країни, у якій планується використовувати цей продукт. Це включає переклад і адаптацію елементів інтерфейсу, допоміжних файлів та документації.

Нині ринок вимагає використання продукту великою кількістю різних мов. Особливо це стосується ринку програмного забезпечення, де сам продукт складається майже виключно з інформації, яку можна локалізувати. Перекладом і адаптацією програмного забезпечення займаються різні фахівці: програмісти, перекладачі, інженери з локалізації, фахівці із забезпечення якості (QA) та менеджери проектів. Без спеціалізованих інструментів локалізація складного програмного забезпечення є дуже складним і трудомістким процесом, який часто зводиться до виконання одного й того ж завдання, яке безрезультатно

повторюється знову і знову [18, с. 175]. Інтернаціоналізація передує локалізації і має на меті одночасно зменшити зусиль і витрат на локалізацію та переклад, а також збільшити швидкість і точність, з якою можна виконати локалізацію [18, с. 176].

Тому інтернаціоналізація (i18n) у розробці програмного забезпечення спрямована на проектування та розробку програмного продукту, готового до глобального розширення з самого початку. Інтернаціоналізація програмного забезпечення у свою чергу полягає не в додаванні перемикача мов до інтерфейсу користувача а натомість він налаштовує продукт на прийом різноманітних форм даних і їх точну обробку відповідно до місцевих вимог.

Слід зазначити, що інтернаціоналізацію програмного забезпечення можна вважати технічним процесом який може передбачати підготовку:

По перше – це елементи інтерфейсу користувача, які можна перекладати: розробники або інженери з локалізації програмного забезпечення можуть перемістити весь вміст, який можна перекладати, в окремий файл ресурсів і залишити програмний код без змін. Це дає змогу перекладачам виконувати роботу з локалізації, не торкаючись вихідного коду.

По друге – це довжина тексту: під час написання програмного коду інтернаціоналізація передбачає, що розробники враховують різницю в довжині між різними мовами. Таким чином вони можуть гарантувати, що перекладений текст залишається видимим в програмному інтерфейсі, навіть якщо він довший за оригінал.

По третє – це числа та символи грошових одиниць: кожна мова має свій унікальний спосіб форматування чисел і валюти, і програмне забезпечення має бути готове до точної обробки всіх цих варіацій.

По четверте – це дата й час. Формати дати й часу відрізняються в різних країнах, тому для програмного продукту важливо відповідати цим різноманітним умовам.

По п’яте – це кодування символів: програмне забезпечення повинно мати можливість працювати з різними шрифтами, включно з нелатинськими.

По шосте – це напрямок мови: для мов, які читаються справа наліво, програмне забезпечення має бути розроблено для обробки тексту, орієнтованого в будь-якому напрямку [25].

З вищесказаного можна зробити висновок, що завдання локалізації не обмежується тільки перекладом, а й включає в себе:

- Використання національних символів валюти.
- Використання установлених форматів для відображення дати та часу.
- Використання методу алфавітного сортування текстових даних.
- Вирівнювання і розміщення елементів інтерфейсу.

Для програмного забезпечення з графічним користувацьким інтерфейсом локалізація також передбачає можливість появи субтитрів залежно від часу, їх форматування на екрані, відповідність відео стандартам та інше [8].

Загалом, щоб зробити програмне забезпечення глобальним (глобалізація), спочатку потрібно підготувати продукт до локалізації (інтернаціоналізації) та виконати наступні кроки:

- спочатку готується продукт – його вихідний код, користувальницький інтерфейс (UI) та інші компоненти – щоб згодом полегшити адаптацію до різних культур і мов;
- потім робиться перехід до фактичного процесу локалізації програмного забезпечення, який включає безпосередньо переклад;
- після того, як відбувся процес адаптації програмного забезпечення для певної мови (набір налаштувань інтерфейсу

користувача на основі мови чи країни), можна протестувати та розгорнути його на різних ринках.

У сучасному гнучкому середовищі локалізація виконується паралельно з циклами розробки програмного забезпечення. У свою чергу це означає, що замість того, щоб додавати повністю окремий робочий процес локалізації до існуючого процесу розробки програмного забезпечення, перекладачі залучаються до вмісту, щойно розробники роблять його доступним [25].

Для того, щоб зробити локалізацію, для початку потрібно обрати Unicode.

Unicode – це стандартний набір символів, який містить всю необхідну інформацію для написання великої кількості мов, що використовуються у комп'ютерах. Unicode являється розширенням всіх інших наборів символів, що вже були закодовані. Форми кодування які можуть застосовуватися з Unicode мають назву UTF-8, UTF-16, та UTF-32.

У наборі UTF-8, символи ASCII кодуються одним байтом, символи в деяких алфавітних блоках – двома байтами і решта BMP представляється трьома байтами. У використанні додаткових символів задіяно 4 байти. Для символів в BMP, UTF-16 резервує по два байти; для додаткових символів – чотири. Для всіх символів використовується 4 байти у UTF-32 [5].

Міжнародні стандарти ISO/IEC 10646 і Юнікод (Unicode) описують і визначають Універсальний набір символів (UCS), який є надмножиною всіх інших стандартів набору символів. Це гарантує сумісність з іншими наборами символів у обох напрямках. Це означає, що жодна інформація не втрачається при перетворенні будь-якого текстового рядка в UCS. Стандарт Юнікод версії 4.0 та ISO/IEC використовують ті самі таблиці наборів символів та методи кодування символів, але стандарт Юнікод додатково надає детальну інформацію про властивості символів,

алгоритми обробки та визначення, які є корисними для реалізаторів. Основним припущенням інтернаціоналізації програмного забезпечення є те, що «всі культурно та лінгвістично чутливі програмні компоненти повинні бути відокремлені від ядра програми». Інтернаціоналізація уможливлює побудову програмних архітектур, в яких усі локально-чутливі елементи відокремлені від локально-незалежного ядра [18, с. 177].

Подалі йде кодування, яке перетворює символи в байтові послідовності для зберігання або передачі даних. Для локалізації важливо підібрати правильне кодування, яке підтримує всі необхідні символи мови та забезпечує коректне відображення тексту.

Слід розглянути основні підходи до локалізації. Нами було виділено наступні: використання файлів ресурсів, вбудовані тексти, системи керування контентом (CMS).

До переваг файлів ресурсів можна віднести те, що у програмі використовуються текстові рядки, які знаходяться у файлах ресурсів. Це полегшує зміну мови інтерфейсу без необхідності вносити зміни у код. Наприклад: У Java використовуються файли з розширенням .properties, у C# — .resx. У JavaScript використовує JSON файли. Це відбувається за допомогою розподілу тексту та коду у різні файли. Перетворення файлів на потрібні мови. Залежно від налаштувань користувача завантаження потрібних файлів.

Перевагами вбудованих текстів є те, що вони для невеликих проектів є простшим у реалізації. У свою чергу до недоліків можна віднести, що зміна коду є необхідною для підтримки та масштабування, оскільки вона передбачає зміну тексту. Використання вбудованих текстів у локалізації відбувається шляхом: тексти вбудовуються прямо в код, і залежно від мови користувача показуються різні текстові дані.

Системи керування контентом (CMS) включають такі переваги: забезпечують централізоване управління текстовим контентом включає

інструменти для локалізації, наприклад: WordPress, Drupal. Можна перекласти контент, який зберігається в базі даних, за допомогою інтерфейсу CMS.

Для автоматизації процесу локалізації можна використовувати наступні інструменти:

- debug: true за допомогою нього всі відомості про стан i18next будуть висвітлені в консоль браузера: до прикладу, коли програму було ініціалізовано, також, яка мова була установлена, і відсутність якогось перекладу.
- lng: 'en-US' у свою чергу застосовується як початкова мова.
- resources являє собою JavaScript-об'єкт із перекладом, який можна назвати ресурсом. Зазвичай є змога визначити їх всередині init-методу, або завантажити їх динамічно використову.чи метод i18n.addResourceBundle також, вони мають змогу завантажуватися асинхронно на основі обраної мови. Об'єкт має вміщати у собі таку структуру {localeName: {namespaceName: {key: 'value'}} Значенням для простору імен (namespaceName) за замовчуванням є translation.
- Namespace – ця функція дає змогу розміщувати переклади в окремих файлах. Що дає змогу не завантажувати весь файл .json, який блокуватиме відтворення контенту вашого веб-застосунку, натомість, можна завантажувати переклади по сторінках.
- react – це об'єкт, який описує конфігурацію для застосунку react-18next.
- I18nextProvider – це компонент, який наданий бібліотекою react-i18next. Він у свою чергу приймає один атрибут i18n із екземпляром i18n.
- @translate() вважається компонентом вищого порядку у React. Він може передавати два додаткові параметри компоненту, а саме: функцію t та екземпляр i18n.

- t – викликаючи функцію t('intro'), впершу чергу виконується перевірка наявності перекладу для ключа intro саме для обраної мови та простіру імен за замовчуванням (translation). Ця функція відобразить переклад, якщо ресурс був знайдений, або ключ.
- Save Missing — i18n може зберігати не наявні переклади на мові dev.
- Language (та Namespace) Fallback на той випадок, коли переклад не був знайдений у мовному файлі то пошук продовжиться в іншому файлі, а потім у dev.
- Gettext – використовується для меток у коді для позначення текстових рядків. Для генерації .pot файлів (шаблони перекладів). Перекладу шаблонів у .po файли для різних мов також, компіляція .po файлів у .mo файли для використання у програмі.
- ICU (International Components for Unicode) – це набір бібліотек для інтернаціоналізації, підтримує формати дати та часу, числові формати, сортування, текстовий аналіз та інші функції.
- eslint-plugin-i18next – плагін, який сповіщає про всі неперекладені рядки в застосунку.
- Декларації для TypeScript від i18next. Вони дають гарантію, що ключі перекладу насправді існують в словнику. Якби їх не було, то довелося б вручну шукати ключі з помилками, або ті, що забули додати [6].

Важливо розрізняти різницю між локалізацією програмного забезпечення та перекладом програмного забезпечення. Переклад перш за все лежить в основі процесу локалізації. Його проводять професійні перекладачі. Однак багато хто ототожнює це з локалізацією, хоча між перекладом і локалізацією є значна різниця. Локалізація ж у свою чергу виходить за рамки простого перекладу. У той час як переклад зосереджений на передачі слів між мовами, локалізація передбачає адаптацію зображень, кольорів, форматів дати й часу, елементів

інтерфейсу користувача тощо, щоб програмний продукт виглядав і відчувався якомога ріднішим для цільового ринку.

Звичайно, процеси локалізації програмного забезпечення відрізняються в різних компаніях, але все ж можна виділити декілька основних елементів:

Дизайн UI/UX: дизайнери змінюють інтерфейс програмного забезпечення відповідно до цільових ринків, вирішуючи будь-які проблеми дизайну (наприклад, розмір тексту перевищує розмір кнопки).

Переклад вмісту: одночасно відбувається переклад вмісту програмного забезпечення, а всі супровідні ресурси, такі як зображення чи символи, налаштовуються відповідно до вимог кожного ринку.

Тестування та забезпечення якості (QA): Тестування локалізації має важливе значення, щоб переконатися, що інтерфейс користувача, UX і функціональність програмного забезпечення є точними та ефективними для кожного регіону перед розгортанням програмного забезпечення у виробництві [25].

Отже, підсумовуючи вище сказане, локалізація програмних інтерфейсів потребує не лише ретельного планування та використання відповідних інструментів і методологій для забезпечення високої якості перекладу та зручності використання а й застосування файлів ресурсів, вбудованих текстів, систем керування контентом та автоматизаційних інструментів, що у свою чергу дозволяє ефективно адаптувати програмне забезпечення для користувачів з різних країн.

Розділ 2
Розроблення та впровадження локалізаційного інтерфейсу системи KSU24

2.1 Розроблення локалізаційної архітектури KSU24

Розроблення локалізаційної архітектури включає у себе низку процесів, саме тому, вважається складним процесом розробки, адже, локалізація це сукупність кроків і процедур, які описують, як виконується локалізація у компанії. Локалізація залучає значну кількість зацікавлених сторін і ролей - як у компанії, так і за її межами.

Слід звернути увагу на те, що у будь-якому процесі локалізації беруть участь наступні працівники:

- команда UX та дизайнерів, відповідальна за створення продукту або функції, яку ви локалізуєте;
- копірайтери та/або UX-розробники, які пишуть текст для локалізації;
- перекладачі, які локалізують текст різними мовами;
- продакт-менеджери, які відповідають за контроль потоку та синхронізацію роботи;
- розробники, які повинні будуть підштовхнути локалізацію до потрібних кроків і втягнути їх назад після її завершення;
- команда маркетологів, яка відповідає за запуск нового продукту чи функції на новому ринку [24].

Перш за все, слід зазначити, що електронний сервіс KSU24 – це інформаційний ресурс забезпечення якісного освітнього процесу Херсонського державного університету. Який дозволяє здобувачам з будь-якого місця та з різних пристроїв мати доступ до електронних

ресурсів, такі як академічні журнали, залікові книжки, інструкції, договори та ін.

Процес локалізації може складатиметься з таких етапів:

- визначення обсягу та цілей, оцінка вимог;
- створення команди для локалізації продукту;
- налагодження процесу локалізації на етапі проектування;
- впровадження інструменти локалізації;
- збір вихідних матеріалів;
- підготовка контенту до локалізації;
- локалізація контенту;
- здійснення ретельного контролю якості;
- налаштування циклу зворотного зв'язку [24].

Для планування архітектури KSU24, на першому етапі було здійснено оцінку вимог. Це стосувалося скільки мов буде підтримувати сайт, які саме елементи інтерфейсу потребують локалізації, також, виявлення чи є специфічні вимоги до правопису, дат та чисел.

Архітектура інтернаціоналізації програмного забезпечення, яка ґрунтується на редукціоністському погляді на культуру та інструментальній теорії технології, еквівалентна архітектурі локалізації користувацького інтерфейсу. Приклад локалізації інтерфейсу користувача наведено на схемі 2.1. Порівняльні дослідження допомагають визначити набір базових елементів, які є спільними для всіх культур, в яких розглядається локалізація. Ці базові елементи специфічні для даної культури (наприклад, символи), інтерпретації загальних елементів (наприклад, різні кольори та піктограми) та інші конструкції складають локалізацію цієї культури. Ці елементи використовуються для визначення спільного інструментарію графічного інтерфейсу користувача (GUI).

Абстрактна концепція інтерфейсу, позначена на схемі 2.1. як Текстове вікно «абстрактне», є узагальненням усіх можливих користувацьких інтерфейсів. Щоб уникнути великих досліджень різних культур, використовується редукціоністський підхід, згідно з яким текстове вікно розробляється у культурі А, абстрактне текстове вікно узагальнюється шляхом забезпечення підтримки локалі інших культур. Зокрема, текстове вікно культури Б отримується шляхом заміни компонентів локалі культури А на компоненти локалі культури Б. [19, с. 90].

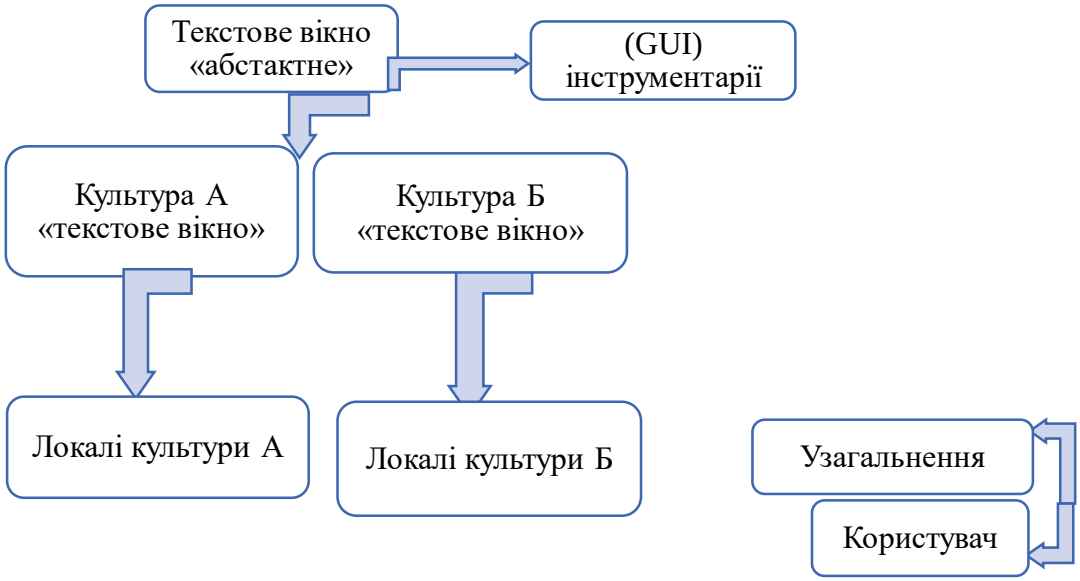

Схема 2.1 Абстрактна концепція інтерфейсу

Подалі, було розглянуто які існують методи локалізації інтерфейсу системи. Існують два основні підходи до локалізації, перший – клієнтська локалізація та другий – серверна локалізація.

Клієнтська локалізація у свою чергу виконується у браузері самого користувача за допомогою JavaScript. Можна вказати будь-які повідомлення для мови. Ресурси автоматично експортуються під час збереження метаданих і доступні для редагування іншими визначеними

мовами в редакторі ресурсів локалізації. У такій локалізації не обов’язково визначати значення повідомлень для всіх мов, які використовують JavaScript. Потрібно визначити хоча б одну мову за допомогою JavaScript. Під час виконання програма вибере переклади з бази даних із поверненням до явних значень, визначених у сценарії.

Серверна локалізація – це коли сам процес локалізації виконується на сервері і сайт генерується відповідно до мови вибраною користувачем. Підтримка такої локалізації JavaScript доступна в workflows і бібліотеках workflows [16].

Загалом, локалізація на стороні сервера призводить до меншого споживання пропускної здатності та трафіку, ніж надсилання метаданих локалізації клієнтам. Але сервер не може припустити, якою мовою хоче користуватися клієнт [26].

До плюсів серверної локалізації віднесено:

- Ваші повідомлення ніколи не покидають сервер і не потребують передачі на сторону клієнта.
- Код бібліотеки для локалізації не потрібно завантажувати на стороні клієнта.
- Не потрібно розділяти ваші повідомлення, наприклад, на основі маршрутів або компонентів.
- Відсутність витрат на виконання на стороні клієнта [20].

У цілому, вибір підходу до локалізації залежить від масштабу самого проєкту. Серверна локалізація підходить для великих проєктів, клієнтська у свою чергу для малих.

Власне кажучи, існує три методології циклу інтернаціоналізації та локалізації:

- локалізація під час компіляції;
- локалізація під час компонування;
- локалізація під час виконання.

Методологія часу компіляції вимагає зміни вихідного коду програми, щоб адаптувати його до місцевого ринку. Методологія часу компонування використовує одну версію вихідного коду, але пов'язує різні бібліотеки для різних локалізацій. Методологія часу виконання використовує єдиний інтернаціоналізований продукт, який може використовувати різні динамічні бібліотеки або визначені користувачем локальні файли ресурсів [18, с. 176].

Можна виділити ще два методи локалізації програмного продукту. Перший, це локалізація на етапі проектування. Практика проектування продукту для міжнародного випуску. Локалізуючи паралельно з розробкою продукту, команди можуть заощадити час і витрати, працюючи над одночасним глобальним запуском.

Другий, традиційний метод водоспадної локалізації полягає в тому, що вона відбувається наприкінці створення продукту. Часто після того, як продукт уже вийшов на свій початковий ринок. Таким чином, виходячи з цих двох методів, можна сказати, що було застосовано традиційний метод локалізації [17].

Наступний етап, у розробці архітектури локалізаційного інтерфейсу – це вибір формату локалізаційних ресурсів. Для того щоб зберігати переклади у проєкті було розглянуто наступні можливі формати ресурсів. JSON, який являє собою найпопулярніший формат для додатків Front-end у таких фреймворках та бібліотеках як до прикладу React. PO-файли, так файли є більш влучними саме у PHP та Python проєктах, вони володіють чіткою структурою, і що не маловажливо, є зручними саме для перекладачів. XLIFF, загалом, цей стандартний формат для локалізації використовують саме для інтеграції перекладацьких серверів. Отже, існують різні формати локалізаційних ресурсів, але, у нашому випадку було обрано перший, а саме JSON, який відповідає усім вимогам проєкту.

Відповідно, після вибору формату локалізаційних ресурсів було обрано структуру каталогів для перекладу. Кожна мова має власні файли для адаптації перекладів, наприклад, рис. 2.1.

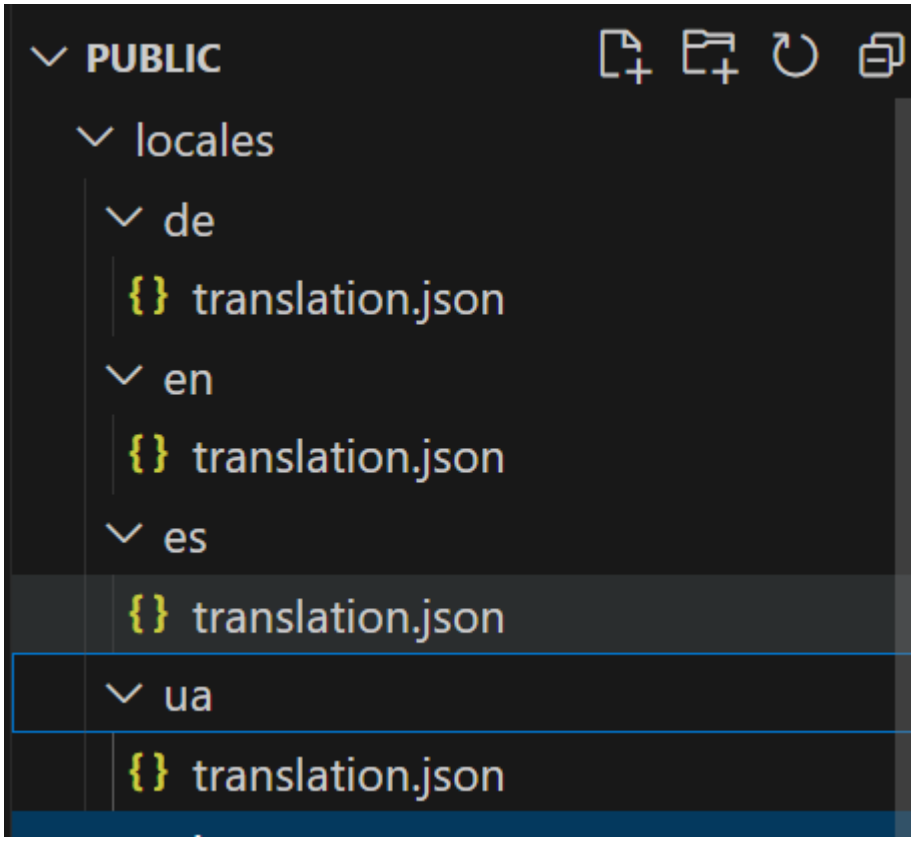

Рисунок 2.1 Структура каталогів для перекладу.

Подалі, було підібрано ключі для текстів, цей етап вважається важливим, оскільки слід уникати прямого використання текстів у коді, саме такий метод спрощує підтримку локалізації інтерфейсу, та з легкістю дозволяє додавати нові мови та тексти рис. 2.2.

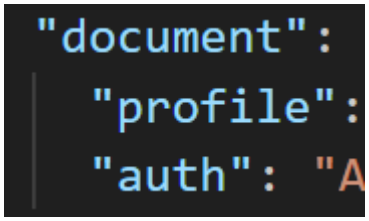

Рисунок 2.2 Ключі до текстів

Що стосується бібліотек для локалізації інтерфейсу, то було обрано i18next, для React.

Також, слід звернути увагу та локалізацію текстів та їх довжину, оскільки, до прикладу німецькі слова можуть бути довшими ніж

англійські чи українські рис. 2.3, саме цьому слід застосовувати гнучкі стилі в CSS для уникнення порушення верстки.

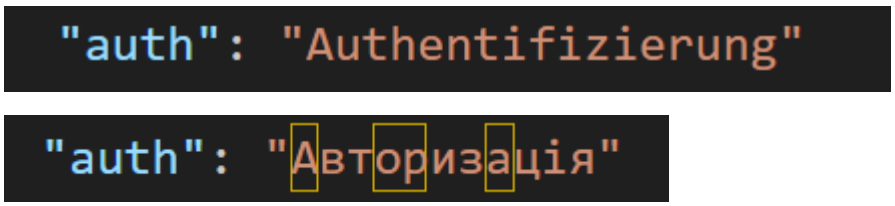

Рисунок 2.3 Локалізація текстів

Передостаннім етапом у розробці локалізаційної архітектури KSU24 звичайно є тестування. Взагалі, є два види тестування – це автоматичне та ручне тестування. Різниця у цих тестуваннях полягає у тому, що ручне тестування передбачає тестування людьми та взаємодію з програмним додатком або продуктом для виявлення проблем, тоді як автоматизоване тестування використовує комп'ютерні програми, програми або сценарії для написання попередньо визначених тестів і запуску їх програмним шляхом [10].

Отже, під час розробки архітектури локалізації було вирішено з яких етапів буде складатися архітектура, та хто буде залучений до процесу, було обрано мови для локалізації, зроблено словники та структуру каталогів, також на передостанньому етапі було здійснено тестування локалізації.

2.2 Аналіз існуючого інтерфейсу KSU24. Визначення текстових ресурсів, що потребують перекладу. Реалізація локалізації.

Перед початком створення локалізації на іноземні мови, для початку слід зробити аналіз вже існуючого інтерфейсу, а саме таких ключових елементів як навігаційне меню, панель інструментів, форми введення даних та діалогові вікна для взаємодії з користувачем. Слід зазначити, що основними функціональними компонентами є такі

елементи, як кнопки, поля вводу, випадаючі списки та підказки, які надають користувачам зрозумілу та інтуїтивну взаємодію.

Загалом, перед аналізом інтерфейсу, ми виокремили наступні характеристиками, якими повинні володіти інтерфейс:

- Адаптивність: означає здатність інтерфейсу налаштовуватися та задовольняти різні потреби та вподобання.
- Достатність: Інтерфейс повинен бути зрозумілим і чітким для користувачів всіх рівнів, незалежно від їхніх знань і задач.
- Дружність: Він повинен бути максимально простим у використанні та задовольняти потреби користувача при вирішенні поставлених задач.
- Гнучкість: Інтерфейс повинен адаптуватися до конкретних задач, полегшуючи формулювання запитів і надаючи результати у зрозумілій формі [2, с. 90].

Безпосередньо, для здійснення самої оцінки зручності інтерфейсу було проведено тестування, яке показало, що інтерфейс досить зручний, але все-таки було виявлено кілька недоліків, які можуть впливати на ефективність його використання, особливо під час адаптації для інших мов. Наприклад, деякі текстові елементи в інтерфейсі мають фіксовану ширину (наприклад у журналах В та П рис. 2.4, або місця, де застосовані скорочення та абревіатури), що створює обмеження під час локалізації для мов з довшими словами.

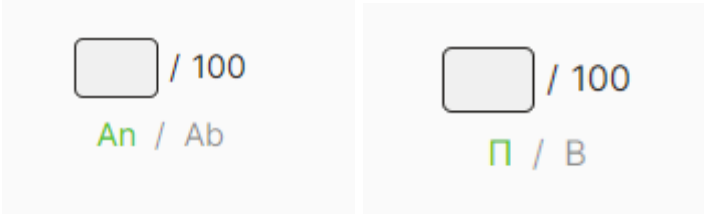

Рисунок 2.4 Академічний журнал: відсутність та присутність здобувачів

У своїй більшості, інтерфейс має статичні текстові елементи, які закодовані безпосередньо у вихідному коді. Саме цьому це може

створювати потенційні проблеми для локалізації, оскільки кожен елемент потребує ручного вилучення та інтеграції в окремі файли мовних ресурсів. Слід відмітити той факт, що в інтерфейсі безпосередньо є динамічні елементи, такі як сповіщення та підказки, які генеруються під час роботи системи. Зрозуміло, що такі елементи також потребують адаптації для багатомовної підтримки.

Подальшим кроком було визначення елементів для перекладу та створення безпосередньо словника елементів, таких як заголовки, кнопки, меню, підказки, повідомлення про помилки, сповіщення та інформаційні блоки. Всі ці текстові ресурси було зібрано, для подальшого перекладу та перенесені в окремі мовні файли рис. 2.5.

home	головна
schedule	розклад занять
diary	звіт про проходження практики
chat	чат
gradebook	академічний журнал
survey	опитування
recordbook	залікова книжка
documents	документи
directory	довідник
useful_links	корисні ресурси
qr	qr код
admin_panel	панель адміністрування
settings	налаштування
developers	розробники
support	допомога
educational	навчальний модуль
admins	адміністрування
finances	фінансовий модуль
legal	юридичний модуль
plans	індивідуальні плани
debts	заборгованості по оплаті за навчання
claims	претензії

Рисунок 2.5 Словник для перекладу

Відповідно, було й проаналізовано динамічні тексти, які генеруються системою під час роботи. Це можуть бути сповіщення про зміни в системі, інформаційні повідомлення або контекстно-залежні підказки рис 2.6.

```
"max_file_size": "Великий файл або недозволений тип файлу. Дозволено до <strong>100МБ</strong>",
```

Рисунок 2.6 Динамічні тексти

Наступним кроком була систематизація текстових елементів, та створення ІД до цих текстових елементів, які слід перекласти. Відповідно до цього була створена таблиця з ІД, текстовим елементом українською мовою та полем для перекладу рис 2.7, що дозволило легко керувати текстовими ресурсами під час перекладу та підтримки мовних версій.

"loader"	"Завантаження",	Herunterladen
"cancel"	"Скасувати",	Abbrechen
"submit"	"Надіслати",	Senden
"add-file"	"Додати файл",	Datei hinzufügen
"add"	"Додати",	Hinzufügen
"required"	"Обов'язкове поле",	Erforderliches Feld
"404"	"Сторінка відсутня",	Seite fehlt
"yes"	"Так",	Ja.
"no"	"Ні",	Nein.
"update"	"Оновити",	Aktualisieren
"create"	"Створити",	Erstellen
"total"	"Всього	Gesamt
"empty_data"	"Дані відсутні",	Daten fehlen
"not_mentioned"	"Не вказано",	Nicht angegeben
"clear_form"	"Очистити форму",	Formular löschen

Рисунок 2.7 Таблиця для перекладу

Подалі, було вилучено всіх текстові ресурсів із вихідного коду та перенесено їх в окремі файли мовних ресурсів. У нашому випадку, це файли у форматі JSON Наприклад: рис 2.8.

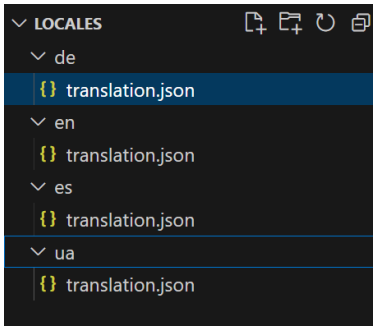

Рисунок 2.8 Файли мовних ресурсів

Також, слід звернути увагу на те, що є три основні типи локалізації програм:

- Мінімальна життєздатна локалізація (MVL)
- Повна локалізація програми
- Гібридна локалізація

Мінімальна життєздатна локалізація (MVL) — під нею розуміється така локалізація, коли перекладається (локалізуються) лише основні функції чи вміст програми.

Повна локалізація програми, саме таку локалізацію було використано на KSU24, є найповнішою формою локалізації, де локалізується все наповнення програми, включаючи текст, зображення, дати, час і одиниці вимірювання.

Крім того, ще є гібридна локалізація, яка поєднує у собі MVL і повну локалізацію програми. У гібридній стратегії локалізації певні ключові елементи програми локалізовано всіма мовами, тоді як інші елементи локалізовані лише на найважливіших ринках [22].

Подалі, було визначено можливі етапи до реалізації локалізації інтерфейсу, вони включали у себе:

- Використання бібліотек для Frontend, було обрано i18next для React.
- Локалізація на сервері.
- Формат файлів для локалізації, було обрано JSON.

Після підготовчих етапів до локалізації інтерфейсу програми, було здійснено реалізацію локалізації інтерфейсу.

Було створено файли для кожної мови з відповідними перекладами, кожен з цих файлів містить ключі для текстових ресурсів, подалі йде процес інтеграції цих файлів у код. Оскільки, на платформі KSU24

використовується бібліотека i18next для React, то за допомогою неї було виведено переклад у компоненті, наприклад:

```
// i18n.js
import i18n from 'i18next';
import { initReactI18next } from 'react-i18next';

// Мовні ресурси
const resources = {
  en: {
    translation: {
       "loader": "Loading",
        "cancel": "Cancel"
    }
  },
  uk: {
    translation: {
      "loader": "Завантаження",
    "cancel": "Скасувати"
    }
  },
  de: {
    translation: {
      "loader": "Herunterladen",
      "cancel": "Abbrechen"
    }
  },
  es: {
    translation: {
      "loader": "Carga",
    "cancel": "Cancelar"
    }
  }
};

i18n
  .use(initReactI18next) // підключення React до i18next
  .init({
    resources,
    lng: ' uk ', // мова за замовчуванням
    fallbackLng: 'en', // резервна мова
    interpolation: {
      escapeValue: false
    }
  });

export default i18n;
```

Подалі, після налаштувань елементів конфігурації, для реалізації локалізації потрібно вивести перекладені тексти наступним чином, наприклад:

```
// MyComponent.js
import React from 'react';
import { useTranslation } from 'react-i18next';

function MyComponent() {
  const { t, i18n } = useTranslation();

  // Функція для перемикання мови
  const changeLanguage = (lang) => {
    i18n.changeLanguage(lang);
  };

  return (
    <div>
      <h1>{t('Головна')}</h1>
      <p>{t('Розклад')}</p>

      {/* Кнопки для зміни мови */}
      <button onClick={() => changeLanguage('en')}>English</button>
      <button onClick={() => changeLanguage('uk')}>Українська</button>
      <button onClick={() => changeLanguage('de')}>Deutsch</button>
      <button onClick={() => changeLanguage('es')}>Español</button>

    </div>
  );
}

export default MyComponent;
```

Звичайно, після підключення бібліотеки до додатку, та виконання всіх необхідних налаштувань, здійснюється етап тестування. Потрібно переконатися, що усі тексти на усіх мовах відображаються коректно, що було перекладено всі елементи, також, що інтерфейс користувача відповідає особливостям кожної мови. Зрозуміло, що це не останній етап у створені локалізації до інтерфейсу, завершальним етапом є регулярні оновлення та підтримка локалізації.

Таким чином, ми дійшли наступних висновків, локалізація – це не лише переклад контенту але й адаптація продукту до регіону, де буде

застосовано локалізацію. Важливим чинником до підготовки локалізації інтерфейсу є безпосередньо аналіз його. Зробивши аналіз інтерфейсу, ми дійшли висновків, що він являється доволі зручним для кінцевого користувача. Що стосується текстових ресурсів, то було виконано аналіз, який дозволив зібрати всі необхідні терміни у словник та потім зробити їх систематизацію та присвоїти всім термінам певний ІД. На етапі реалізації було підготовлено конфігурацію, налаштовано кнопки для перекладу та перенесено адаптований текст до файлів мовних ресурсів. На передостанньому етапі було виконано тестування локалізації інтерфейсу. Отже, локалізація інтерфейсу являє собою складний процес, який включає велику кількість етапів, для створення якісної локалізації інтерфейсу.

ВИСНОВКИ

У ході проведеного дослідження ми дійшли наступних висновків.

У досліджені були розглянуті теоретичні та практичні аспекти локалізації програмного забезпечення, які дозволили сформувати цілісне уявлення про важливість застосування іноземних мов не лише у програмних додатках а й у KSU24.

Було визначено, що використання спеціалізованих бібліотек і фреймворків значно полегшує процес локалізації, забезпечуючи гнучкість і масштабованість інтерфейсів. У програмному забезпечені KSU24 використовується бібліотека i18next, що має підтримку фреймворку React. У свою чергу бібліотека i18next – це не просто бібліотека, яка виконує переклади тексту. i18next – це фреймворк, який дає змогу розгорнути повноцінне рішення для локалізації програмного забезпечення застосунку.

У процесі дослідження, було розмежовано поняття між локалізацією та інтернаціоналізацією, таким чином, локалізація – це процес модифікації програмного продукту для врахування мови та культурних особливостей користувача, а інтернаціоналізація – це проектування та розробка програмного продукту або документації з метою спрощення подальшої локалізації.

Таким чином, проведене дослідження дозволило систематизувати знання про процеси локалізації програмних інтерфейсів. Було виявлено, що ефективна локалізація вимагає не лише перекладу текстових рядків, але й глибокого розуміння культурних особливостей цільової аудиторії, а також урахування технічних аспектів розробки програмного забезпечення.

На основі проведеного аналізу системи KSU24 було розроблено локалізаційну архітектуру, досліджено, хто бере участь у процесах локалізації, визначено етапи локалізації, а саме:

- визначення обсягу та цілей, оцінка вимог;
- створення команди для локалізації продукту;
- налагодження процесу локалізації на етапі проектування;
- впровадження інструменти локалізації;
- збір вихідних матеріалів;
- підготовка контенту до локалізації;
- локалізація контенту;
- здійснення ретельного контролю якості;
- налаштування циклу зворотного зв'язку.

Було проаналізовано існуючий інтерфейс системи KSU24 і визначено основні текстові ресурси, що потребували перекладу, після вибору формату локалізаційних ресурсів було обрано структуру каталогів для перекладу. Кожна мова має власні файли для адаптації перекладів. Було підібрано ключі для текстів, цей етап вважається важливим, оскільки слід уникати прямого використання текстів у коді, саме такий метод спрощує підтримку локалізації інтерфейсу, та з легкістю дозволяє додавати нові мови та тексти. Реалізація локалізації була проведена з урахуванням кроскультурних відмінностей, що сприяло підвищенню зручності використання системи різними мовними групами.

Таким чином, проведене дослідження підтвердило важливість і актуальність локалізації для забезпечення зручності та доступності програмного забезпечення на міжнародному рівні. Розроблені рекомендації можуть бути використані для подальшого вдосконалення системи та її інтеграції з новими мовними модулями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бібліотеки мови програмування. URL: https://ua5.org/osnprog/1660-biblioteky-movy-programuvannya.html#:~:text=%D0%86%D1%81%D0%BD%D1%83%D1%94%20%D0%B4%D0%B2%D0%B0%20%D1%82%D0%B8%D0%BF%D0%B8%20%D0%B1%D1%96%D0%B1%D0%BB%D1%96%D0%BE%D1%82%D0%B5%D0%BA%20%D0%BC%D0%BE%D0%B2%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F%3A%20%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D1%96%20%D1%82%D0%B0%20%D1%81%D1%82%D0%BE%D1%80%D0%BE%D0%BD%D0%BD%D1%96. (дата звернення: 20.08.2024).

2. Бідюк П.І., Коршевнюк Л.О. Проектування комп'ютерних інформаційних систем підтримки прийняття рішень: Навчальний посібник. Київ: ННК „ІПСА" НТУУ „КПІ", 2010, 340 с.

3. Дивиєш Батасана. Повний посібник з локалізації контенту: Вихід на світову аудиторію, 10.24.2023 URL:https://www.ranktracker.com/uk/blog/complete-guide-to-content-localization-reaching-global-audiences/(дата звернення: 15.08.2024).

4. Інтернаціоналізація та локалізація. URL:https://qalight.ua/baza-znaniy/internatsionalizatsiya-ta-lokalizatsiya/(дата звернення: 15.08.2024).

5. Кодування символів: Основні поняття. URL:https://www.w3.org/International/articles/definitions-characters/index.uk(дата звернення: 30.08.2024).

6. Локалізація React (Mobx) застосунку за допомогою i18next. URL: https://devzone.org.ua/post/lokalizatsiia-react-mobx-zastosunku-za-dopomohoiu-i18next (дата звернення: 22.08.2024).

7. Локалізація в порівнянні з Інтернаціоналізацією. URL:https://www.w3.org/International/questions/qa-i18n.uk.html (дата звернення: 15.08.2024).

8. Локалізація програмного продукту. URL: https://uk.wikipedia.org/wiki/%D0%9B%D0%BE%D0%BA%D0%B0%D0%BB%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82%D1%83 (дата звернення: 29.08.2024).

9. Навіщо потрібні фреймворки та бібліотеки. URL: https://robotdreams.cc/uk/blog/251-zachem-nuzhny-freymvorki-i-biblioteki (дата звернення: 20.08.2024).

10. Ручне та автоматизоване тестування. URL: https://www.testrail.com/blog/manual-vs-automated-testing/#:~:text=Manual%20testing%20involves%20humans%20testing,tests%20and%20run%20them%20programmatically(дата звернення: 09.09.2024).

11. Фреймворки у мовах програмування: навіщо служать та як їх вибрати. URL: https://cloud.itstep.org/blog/frameworks-in-programming-languages-what-are-they-for-and-how-to-choose-them (дата звернення: 20.08.2024).

12. Abdulfattah O., et al. Localization Quality Assessment for More Reliable E-Commerce Applications in Arabic, Education Research International. https://doi.org/10.1155/2022/6942735.

13. Aguayo Arrabal N., and C. Ramírez Delgado Análisis de errores lingüísticos en localización web para la exportación del juguete en España, Quaderns de Filologia: Estudis Lingüístics, 2022, 27, S. 113–152. https://doi.org/10.7203/QF.27.24670.

14. Anastasiou, D., and R. Schäler Translating Vital Information: Localisation, Internationalisation, and Globalisation, Synthèses, 2009, 3, S. 11–25.

15. Essential JavaScript i18n Libraries You Want to Use in 2023. URL: https://www.transphere.com/javascript-i18n-libraries/ (дата звернення: 22.08.2024).

16. Client-side and Server-Side Localization. URL: https://docs.fintechos.com/Studio/22.1/UserGuide/Content/DigitalJourneys/Localization/ClientServerSideLocalization.htm(дата звернення: 05.09.2024).

17. Design-stage localization: How to integrate design into the localization process. URL: https://www.gridly.com/blog/design-stage-localization-how-to-integrate-design-into-the-localization-process/ (дата звернення: 09.09.2024).

18. Elvis Hau, Manuela Aparício . Software Internationalization and Localization in Web Based ERP Conference Paper · September 2008 C. 175 - 180
DOI: 10.1145/1456536.1456570

19. Gregory Kersten, Mik Kersten, Wojciech M. Rakowski,. Software and Culture: Beyond the Internationalization of the Interface. *Journal of Global Information Management*, 2002, 10(4), C. 80-100 URL:https://www.researchgate.net/figure/User-interface-localization_fig1_220500443 (дата звернення: 15.08.2024).

20. Internationalization of Server & Client Components. URL: https://next-intl-docs.vercel.app/docs/environments/server-client-components(дата звернення: 05.09.2024).

21. OpenAI, ChatGPT URL: https://chatgpt.com/(дата звернення: 20.09.2024).

22. Mobile app localization: Importance and steps. URL: https://poeditor.com/blog/mobile-app-localization/#App-elements-you-need-to-localize(дата звернення: 09.09.2024).

23. Miguel A. Jiménez-Crespo Localization in Translation DOI: 10.4324/9781003340904

24. Ross Weldon. 9 steps for building a bulletproof localization process. URL: https://lokalise.com/blog/localization-process/(дата звернення: 05.09.2024).

25. Stephan Schoening The Complete Guide to Software Localization. URL: https://phrase.com/blog/posts/software-localization/(дата звернення: 29.08.2024).

26. Where should I do localization (server-side or client-side)? URL: https://softwareengineering.stackexchange.com/questions/313726/where-should-i-do-localization-server-side-or-client-side(дата звернення: 05.09.2024).