УДК 004.04,004.05
DOI: 10.18523/2617-3808.2024.7.40-50

Верета В. В., Ткаченко В. О.

АВТОМАТИЗОВАНА ЛОКАЛІЗАЦІЯ ЗАСТОСУНКІВ У МІКРОСЕРВІСНІЙ АРХІТЕКТУРІ

У цій статті проведено аналіз різних інструментів і сервісів для здійснення локалізації (перекладу) вебзастосунків, методів і підходів до їх інтеграції та масштабування. Розглянуто наявні рішення для управління локалізаціями вебсервісів. Описано розроблену архітектуру, яка дає змогу швидко адаптувати сервіс під потреби різних користувачів і різних проєктів та ефективно інтегруватися з різними платформами з урахуванням простого масштабування сервісу.

На базі цієї архітектури реалізовано вебсервіс EchoLocal. Він покращує взаємодію учасників процесу локалізації вебзастосунків, а також дає змогу оптимально інтегрувати цей процес у розподілену мікросервісну архітектуру сучасних застосунків і просто розгортати у своїй закритій екосистемі. EchoLocal надає можливість своїм користувачам інтегрувати та управляти локалізаціями своїх продуктів в одному місці в реальному часі.

EchoLocal допоможе знизити витрати на локалізацію, спростити процес адаптації контенту під різні мовні ринки та підвищити ефективність комунікації між усіма учасниками процесу локалізації.

Ключові слова: локалізація, інтернаціоналізація, глобалізація, мікросервісна архітектура, Kubernetes, Google Cloud Platform, GCP, CI/CD.

Вступ

Локалізація, або переклад, вебсервісів стає вирішальною задачею у процесі глобалізації програмних продуктів, онлайн-рішень та автоматизації процесів оброблення інформації [1]. Нині дедалі більшого значення набувають своєчасність і чіткість адаптації контенту до потреб різномовних користувачів. Вебсервіси, які мають більше локалізацій, зазвичай здатні залучити ширшу аудиторію і забезпечити кращий, клієнтоорієнтований досвід. Однак управління багатьма мовами може стати складним, ресурсоємним завданням без належних інструментів та підходів. Водночас, потреба у своєчасній і точній локалізації лише зростає, а ефективність її реалізації може значно впливати на успіх продукту на міжнародному рівні.

Саме тому необхідно підвищувати обізнаність бізнесу та продуктових рішень щодо локалізації та забезпечити розуміння не меншої важливості цього, ніж конкретні особливості систем.

Нині існує декілька взаємопов'язаних понять, які безпосередньо стосуються приведення та перенесення онлайн-продукту на нові ринки, але не є чимось одним і тим самим. Є поняття локалізації (localisation, l10n), яке буде досліджене у цій роботі, і поняття інтернаціоналізації (internationalisation, i18n). Їх часто використовують як взаємозамінні, однак іноді це не так.

Цифровий продукт можна придбати онлайн, про нього можна дізнатись онлайн і він не вимагає якогось локального представництва. Тобто тепер, маючи ідею продукту для споживача, нам необов'язково створювати собі рамки і кордони. Ось тут нас і чекають ускладнення, адже що ми знаємо про весь світ? Ми знаємо і розуміємо мову нашого регіону, форматування дат, валют і чисел; однак знайомі нам образи і формати можуть бути незрозумілими нашим покупцям на іншому кінці світу. Отже, ми маємо подумати про інтернаціоналізацію, вбудувати її і додати локалізацію для підтримки інших регіонів.

Інтернаціоналізація — узагальнене поняття, що означає принципи проєктування та реалізацію програмного продукту або документації в такий спосіб, який максимально спростить подальшу локалізацію [11]. Можна сказати, що це технічне вдосконалення продукту для того, щоб забезпечити, по-перше, можливість його адаптації, а по-друге, легкість локалізації для різних ринків без необхідності виконання дорогих змін. Це фундамент, на якому будують подальші налаштування відкритості світу, іншим цінностям і форматам.

Локалізація — процес адаптації програмного продукту до мови та культури клієнта [11]. Тобто саме локалізація стосується процесу зміни вже інтернаціоналізованого (інколи глобалізованого) продукту для конкретного ринку. Це конкретні зміни виду, форматів та мови для споживачів конкретної країни або регіону.

Наведемо кілька прикладів, щоб уточнити поняття інтернаціоналізації:

– створення продукту з урахуванням можливості використання різних мов, які зберігаються в окремих джерелах і не вбудовані у саму систему;
– проєктування продукту з урахуванням нестандартного розміщення тексту справа наліво, зверху вниз; проєктування компонентів і використання вже перевірених компонентів, які вміють працювати з такими змінами;
– використання у продукті локалізованих знаків валют, дат, чисел тощо, що зазвичай досягається вбудованими утилітами мов програмування.

Тепер, розуміючи поняття інтернаціоналізації, визначимо наше розуміння локалізації [17]:

– процес адаптації продукту або контенту для конкретного ринку, регіону;
– передбачає переклад, адаптацію та зміну продукту, щоби врахувати конкретні лінгвістичні, культурні або регуляторні вимоги конкретної країни, регіону або ринку, наприклад використання локалізованого формату дат, валют та одиниць вимірювання;
– виконують за конкретним запитом, відповідає поточному етапу розширення та впливу.

Обидва процеси надзвичайно важливі для бізнесу або продукту, який хоче розширитися та бути досяжним глобально. Ці інтегровані процеси дають змогу компаніям надавати більш персоналізований і відповідний досвід для своїх користувачів.

Отже, локалізація — це процес адаптації продукту або контенту під конкретну культуру або мовну аудиторію. Локалізація забезпечує не тільки переклад тексту, а й адаптацію графічних елементів, валют, форматів дати та часу, використання правильних культурних референцій, а також технічних вимог, як-от кодування символів. Процес локалізації потрібен для того, щоб продукт легко сприймався користувачем, підвищуючи тим самим його зручність та ефективність використання в різних регіонах і ринках. Така адаптація дає змогу компаніям ефективніше взаємодіяти з міжнародною аудиторією та розширювати свій вплив.

Сучасні підходи до локалізації вебзастосунків

Вибір підходу до локалізації або перекладу вебзастосунків може бути непростою задачею. Звісно, кожен розробник, менеджер або власник проєкту обирає сам, як проходитиме процес локалізації, наскільки це критично і хто за це відповідатиме. Кількість інформації щодо цього, а також інструментів, теж немала, тож розглянемо загальні підходи і конкретно запропонуємо своє рішення на стику цих підходів.

Починаючи зі вбудованих, пов'язаних із кодом підходів до повноцінних сервісів, які повністю перепишуть ваш контент, немає єдиного правильного рішення, яке підходитиме в кожному випадку. Від загального підходу до бізнесу, налаштованих бізнес-процесів і навіть орієнтації самої компанії буде залежати, який підхід до локалізації служитиме краще. Наприклад, є повноцінно комп'ютерно налаштовані організації, де все йде від розробників, де всі говорять зрозумілою лише технічним спеціалістам мовою. Тут мабуть не буде абсолютною проблемою, якщо процес налаштування і зміни локалізованого контенту проходитиме через розробника із фактичними змінами у кодову базу. Але ж є і такі компанії, які все ще потребують послуг розробників, але не є тільки комп'ютерно налаштованими, де міркують більше самою ідеєю, заробітком та маркетингом. Напевно тут було б краще вибрати підхід, який потребує меншої участі розробника або ж взагалі все можна зробити без нього.

Перелічимо орієнтовні запитання, які варто поставити перед вибором підходу. Хто створює і коректує контент на вебсайті? Як часто планують зміни до вже готового тексту? Хто відповідає за переклад? Наскільки компанія орієнтована на ІТ і скільки розробників можуть бути задіяні? Чи важливою є продуктивність додатку? Чи є інші системи, з якими потрібно інтегруватися? Чи є вимоги до безпеки та системи авторизації?

Розглянувши запитання, можна перейти до переліку підходів, яких насправді лише три: локалізація, вбудована у код (традиційна, ручна); локалізація, винесена в окрему CMS (напівавтоматична); проксі-локалізація (автоматична).

Залежно від технологічного стеку самого вебзастосунку можуть бути різні рішення, що підходять саме для нього. Можна вважати їх базовими, з яких усе починається і які підтримані розробниками. Підійдуть ці варіанти чи ні у кожному окремому випадку використання, потрібно дивитись на конкретних прикладах. Наприклад, природним вибором для React-коду є використання бібліотек i18n, таких як react-i18next, react-intl або lingui [16]. Існують також спеціалізовані бібліотеки для конкретних фреймворків, наприклад gatsby-plugin-i18n для Gatsby [5] та next-intl для Next.js [10]. На високому рівні архітектура більшості цих бібліотек схожа. Перевагами традиційної локалізації є:

1. Налаштування — розробники мають повний контроль і відповідають за реалізацію локалізації у повному обсязі. Окрім того, це роблять лише при налаштуванні самого проєкту.
2. Оптимізація — переклад отримується зі статичного файлу, який розміщений безпосередньо в проєкті або завантажується зі стороннього ресурсу. Це означає, що після того, як дані файлу доступні, переклад можливий миттєво.

Недоліками ж можна вважати такі пункти:

1. Підтримка розробників — потрібна постійна участь розробників у налаштуванні, перевірці та корекції контенту. Для будь-якої зміни потрібна взаємодія безпосередньо з кодом проєкту.
2. Необхідність тримати файли перекладу поряд — компроміс між розміром файлів коду та часом на завантаження певного мовного файлу.

Ми розглянули підхід і його імплементацію зі сторони розробників проєкту, як він влаштований і як його підтримують. Однак сама локалізація — переклад текстів — відбувається в іншому місці. Для невеликих проєктів із малою кількістю тексту це може виконуватись вручну перекладачами і потім знову збиратись у файли, які повернуться у код. Більші ж проєкти вже потребують TMS (Translation Management System) для управління локалізаціями. Це сервіси, які зазвичай відірвані від розробки і більше налаштовані на роботу і управління командою перекладачів, адже вони також мають свої процеси, які потрібно контролювати.

Однією з таких систем є Localazy [4]. Сервіс має багату функціональність і змістовну документацію, але тут я пропоную розглянути ту його частину, яка відповідає саме за закриття прогалини у локалізації, що у нас утворилась, — переклад контенту. Localazy спрощує та організовує процес перекладу, допомагає ефективно розподілити і проконтролювати роботу, щоб забезпечити якісний результат. Подібних сервісів багато, всі вони пропонують як додаткові просунуті функції на зразок рішень на основі штучного інтелекту та різного роду автоматизації.

Напівавтоматична локалізація, винесена в окрему CMS

Розглянемо підхід до локалізації, який вважають напівавтоматичним. У контексті цієї роботи він є таким, що відбувається за допомогою стороннього сервісу — CMS (Content Management System). Такий підхід підтримує написання контенту кількома мовами, які має підтримувати вебзастосунок, і спрощує редагування контенту для нетехнічних спеціалістів і перекладачів, які можуть керувати контентом через цей окремий сервіс.

CMS-системи мають бути глибоко інтегровані у кодову базу застосунку. Особливості першої конфігурації залежать від самого сервісу, але потім нетехнічні спеціалісти зможуть створювати новий контент або редагувати наявний безпосередньо через CMS. На платформі можна керувати локалізаціями або мовами, які доступні у вебзастосунку, можна додати новий переклад для наявної структури. Розробники отримують локалізований контент через вказаний ідентифікатор локалізації.

Традиційний підхід і CMS мають багато спільного, зокрема обидва зберігають перекладений контент як структуровані дані, які застосунок може отримати за ідентифікатором. Однак вони відрізняються підходом до програмування та необхідною архітектурою кодової бази, адже мають різні API (Application programming interface) для взаємодії:

– для реалізації традиційного підходу потрібна реалізація і робота з контентом на рівні рядків;
– системи CMS дають змогу визначати типи самого контенту, групуючи текстові дані у так звані компоненти — публікація у блозі, картка товару тощо. Тобто коли застосунок отримує дані з CMS — це результат у вигляді об'єкта, що являє собою перекладений контент компоненту, який безпосередньо розміщений на сторінці. Такий підхід потребує завчасно продуманої архітектури застосунку, і його не можна просто замінити іншим підходом.

Перевагою напівавтоматичної локалізації на базі CMS є те, що розробники мають повний контроль над реалізацією перекладу, відображенням локалізованого контенту, і його легко редагувати.

Головний недолік — складність початкового налаштування та відповідний дизайн архітектури, який створений для роботи з конкретною CMS та запрограмованими компонентами. Типовим представником CMS є сервіс Contentful [2], що являє собою повноцінну CMS із безліччю функцій для управління даними, ролями, перекладами тощо.

Автоматична локалізація

Цей підхід є найцікавішим і майже нічого не потребує зі сторони розробників застосунку. Але водночас він більш ризиковий і не дає повної гарантії щодо повноцінної локалізації, адже для ідеального виконання цього підходу мають бути виконані певні умови. Для досягнення автоматичної локалізації багато сервісів зробили крок уперед і автоматизували збирання тексту з активного сайту, який доступний для користувачів, і навіть динамічну заміну тексту.

Ці автоматизовані сервіси зазвичай мають свої бібліотеки, які мають бути включені напряму в кодову базу застосунку. Під час завантаження ресурсу скрипт знайде увесь видимий текст і завантажить їх у панель керування. Сервіс покаже список і запропонує машинний переклад або дозволить додати його вручну. Тільки-но контент буде локалізовано, при завантаженні вебзастосунку вбудований скрипт знаходитиме відповідний до локалізації користувача переклад і замінюватиме DOM (Document object model) елементи, щоб відобразити текст із відповідним перекладом.

На перший погляд, цей підхід є такою собі срібною кулею, яка вирішує всі питання перекладу для вебсайтів, майже все автоматизовано і з мінімумом налаштувань. Потрібно тільки перекладати автоматично знайдений текст, але на практиці все може бути не так райдужно через обмежену кастомізацію, неможливість зберігати й повторно використовувати частки контенту, передати контекст тощо.

Отже, можна назвати такі переваги підходу: легке налаштування (підключити бібліотеку або скрипт напряму в застосунок, зазвичай на рівні базового HTML), легке використання (текст автоматично знайдений, вивантажений на платформу та показаний користувачу після перекладу; оскільки все робиться через окремий сервіс, то в основному застосунку жодні зміни не потрібні і можна дотримуватися свого поточного процесу без змін).

Недоліки також очевидні й випливають із переваг: оптимізація застосунку (оскільки текст замінюється напряму у DOM після завантаження перекладу, це відбувається з певною затримкою, і користувач це може помітити), обмежена кастомізація (оскільки сервіс автоматично зчитує написані текстові рядки у застосунку, він має чітко «розуміти» структуру кодової бази, щоб їх знайти; така автоматизація значно зменшує можливості індивідуальних проєктних налаштувань, адже має бути універсальною для всього застосунку).

Як ілюстрацію автоматичного варіанта розглянемо сервіс Gtranslate [3]. Цей сервіс застосовують у випадках, коли потрібна не обов'язково дуже точна і якісна локалізація, але її наявність може відігравати важливу роль. Для роботи із сервісом майже нічого не потрібно зі сторони розробника, зокрема жодних змін до кодової бази. Є певні налаштування DNS (Domain Name System), які потрібно зробити, щоб пропустити сайт через сервіс і виконати переклад. Основним є налаштування CNAME (Canonical Name), яке відповідатиме за перенаправлення вашого сайту на сервіс локалізації, а також перенаправлення локалізованих версій.

Сервіс пропонує безкоштовну версію із машинним перекладом, а також декілька варіантів із розміщенням локалізованих версій сайту, із використанням субдоменів або субдиректорій. Переклад, який було створено за допомогою параметра "language_edit=1", доступний до редагування прямо на сторінці. Також сервіс надає мінімальні можливості для встановлення селектора мов і має плагіни для налаштувань для певних популярних систем, наприклад Wordpress.

Підсумовуючи інформацію щодо різних підходів із локалізації вебзастосунків, можна сказати, що немає єдиного рішення для всіх випадків. Часто рішення залежить від того, наскільки тісно з цим працювати може сам розробник, чи можлива його підтримка. Автоматизована локалізація може бути гарним варіантом, коли потрібна найменша підтримка зі сторони розробників. Зазначимо, що перехід між будь-якими підходами теж не є легким завданням. Для кожного з них кодова база має мати певну структуру та по-різному викликати локалізовані елементи, а для автоматизованої не всі технології підійдуть.

Важливо виділити окремо ситуацію, коли можлива інтеграція у вже наявний проєкт без локалізації взагалі. В такому разі важливо зорієнтуватись, якого типу застосунок, його архітектура та яка кількість тексту для локалізації.

Вебсервіс EchoLocal для оптимізації локалізації

Отже, врахувавши згадані в попередньому розділі підходи до локалізації та особистий досвід роботи, у нас виникла ідея розробити застосунок, який покращив би, оптимізував цей процес загалом.

У чому головна складність перекладу? На нашу думку, у передаванні інформації у ланцюжку замовник — дизайнер — розробник — перекладач. Те, що здається на перший погляд логічним і очевидним, може значно змінити своє значення у процесі, якщо немає спілкування між учасниками. Також складності додає те, що цей процес асинхронний і потребує певного часу для виконання роботи кожним учасником. Потрібна певна база знань, де буде збережено контекст, щоб максимально зручно й ефективно передати значення кожного тексту перекладачу.

На ринку існують, з одного боку, інтегровані рішення для технологічного аспекту, бібліотеки, які реалізують сам функціонал перекладу у вебзастосунку, а з іншого — гарні вебрішення для перекладачів, які спрощують роботу саме для них. Наша ж ідея полягала в тому, щоб почати реалізовувати щось середнє між цим, даючи можливість ефективно співпрацювати розробникам і людям, які відповідають за контент, без додаткових, непотрібних дій.

Один із підходів до перекладу контенту — групування ключів та їхніх значень у файл. Цей файл лежить на сервері разом із проєктом і завантажується разом із ним у браузер користувача. Це найшвидший і найлегший спосіб для проєкту отримати нові значення, але процес перекладу ключів, знання контексту цих ключів залишається абсолютно незручним і неоптимізованим. Наша ідея проєкту вирішує цю проблему. Тепер цей файл ключів має зберігатись не у проєкті всередині, а на окремому ресурсі, який бере на себе всю роботу з організації перекладу та наповнення вебсторінки контентом. Самому проєкту залишається лише підвантажити перелік необхідних ключів із перекладами, це можуть бути всі ключі або лише частина. Також за такого підходу може бути доступна певна метаінформація про переклад, як-от дата створення, дата останнього оновлення тощо.

При цьому ми не зменшуємо зручність для самих перекладачів, адже саме для них на ринку дуже багато платформ, так званих менеджерів перекладів. Нова платформа для менеджменту перекладів, до якої допускають самих розробників проєктів, які потребують перекладу, надає певні переваги та поліпшення. Наприклад, тепер ключі можуть створювати люди, які фактично їх додають у проєкт, є можливість завантажити картинку для контексту та іншу додаткову інформацію, яка допоможе перекладачеві краще зорієнтуватись.

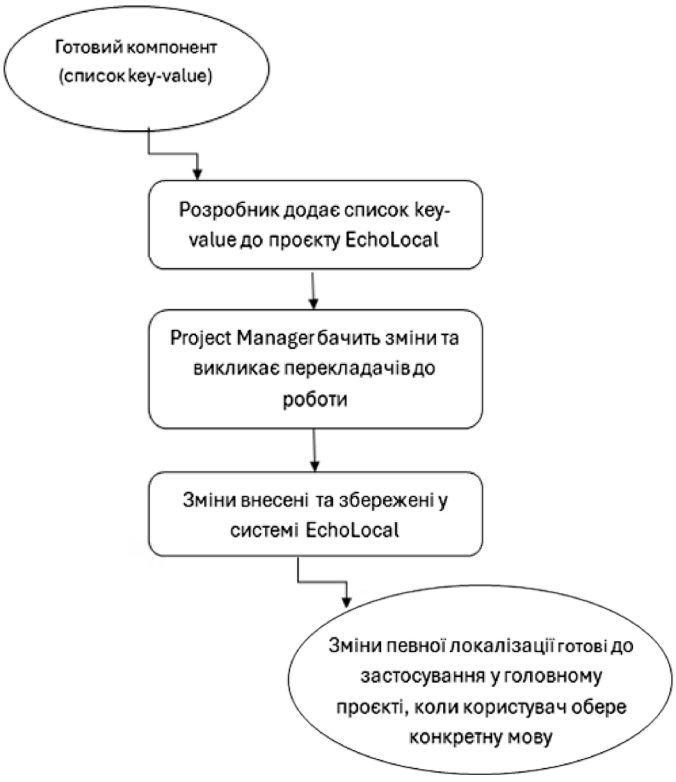

Рис. 1. Сценарій роботи сервісу

Сценарій роботи сервісу такий (рис. 1). Уявімо, що є дизайн і розробник уже спроєктував і вніс необхідні зміни до коду. Маємо робочий функціонал на вебсторінці з текстом англійською мовою. Текст не є вбудованим, а підв'язаний до ключів, які своєю чергою перетворюються на реальний текст необхідною мовою. Саме ці ключі і текст необхідною мовою будуть завантажуватись у проєкт. У самому проєкті є налаштований функціонал для роботи із сервісом перекладів, де отримують ключі, а також один файл перекладів основною мовою (так званий fallback, переклади англійською мовою згідно з цим прикладом). Розробник самостійно вносить цей ключ і переклад у запрограмовану систему EchoLocal і прикріплює необхідні допоміжні елементи, щоб передати контекст. На цьому робота замовника закінчується. Далі штат перекладачів вносить зміни, додаючи переклади на інші мови, а застосунок уже сам підтягне ці зміни залежно від вибраної користувачем мови.

У чому полягає основний виграш від використання платформи EchoLocal? На нашу думку, у вирішенні проблеми ефективності шляхом зняття зайвої, непрофільної роботи з розробника. Тепер кожен займається своєю роботою і не залежить від часу й роботи іншого. Спочатку зробив свою роботу (запрограмував), підготував роботу для іншого (завантажив ключі) і пішов, далі все автоматично запрацює, коли буде готово (після оновлення контенту перекладачами). Також є неочевидна перевага у контролі, адже тепер не потрібно шукати текст на вебсайті або, перевіряючи за якимось критерієм, мати доступ до всіх частин сайту. Можна просто переглянути список усіх ключів і їх перекладів на зручному ресурсі та зробити необхідні зміни без будь-якої зовнішньої допомоги.

Основними можливостями сервісу є: зберігання даних проєкту, для якого роблять переклад, і списку усіх доступних перекладів, зберігання листів перекладів на різні мови для зручної роботи перекладачів і менеджерів проєкту, можливість оновлення контенту за межами сайту і автоматичного оновлення на сайті, незалежність всіх учасників процесу у часі, ефективність роботи всіх учасників, кожен з яких займається своєю справою.

Далі пропонуємо розглянути загальну архітектуру проєкту, схему бази даних і підхід до розгортання проєкту.

Загальна архітектура EchoLocal складається із окремих сервісів для клієнта та сервера, які співпрацюють один з одним за допомогою обміну HTTP запитами (рис. 2). Клієнт і сервер являють собою два окремі застосунки, розміщені в одному репозиторії (монорепозиторії), незалежні частини, які співпрацюють. Оскільки однією із цілей застосування є підтримування роботи з мікросервісами, то такий розподіл є необхідним, адже API має бути доступним для зовнішніх джерел: запити від зовнішніх проєктів мають йти напряму до серверної частини застосунку і завантажувати переклад або його частину. Тож користувачі не прив'язані до фронтенд-частини і мають можливість користуватись API напряму.

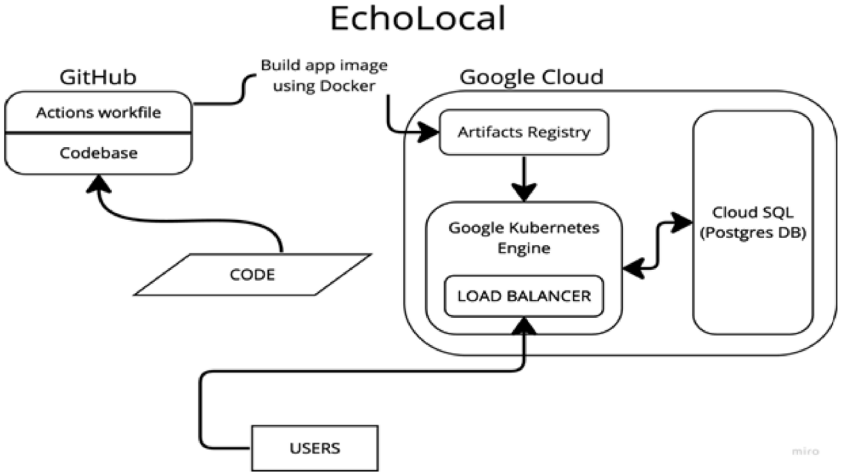

Рис. 2. Загальна архітектура проєкту EchoLocal

Проєкт EchoLocal запущений у хмарі на потужностях Google Cloud. Складовими частинами проєкту є: Google Kubernetes Engine [8], Artifacts Registry [7], Cloud SQL [9], Docker [12], GitHub [13], GitHub Actions [6].

Частина хмарного рішення Google Kubernetes Engine є сервісом для роботи із Kubernetes [15] від Google Cloud [14]. Головний тримач застосунку — автоматизований Kubernetes кластер, який має

декілька реплікацій і доступ ззовні, може гнучко налаштовуватись під потреби застосунку залежно від навантаження. Сервіс Artifacts Registry використовують для зберігання та управління образами застосунків для подальшого використання у Kubernetes. Cloud SQL є частиною хмарного рішення для баз даних від Google Cloud. Для нашого застосунку ми вибрали базу даних Postgres (детальну схему буде показано далі). Рішення GitHub використано для збереження кодової бази та подальшого розгортання проєкту. Репозиторій містить весь код проєкту і дає змогу зручно його переглядати та оновлювати. Для проєктів із відкритим кодом він дає можливість переглядати код іншим спеціалістам і пропонувати зміни та покращення.

Допоміжні інструменти Docker, GitHub Actions використано для зручнішого розгортання застосунку на платформі Google Cloud. Docker — інструмент контейнеризації, який дає змогу «упакувати» застосунок у контейнер із заданими налаштуваннями оточення так, що будь-який інший розробник зможе відтворити це оточення, якщо у нього встановлений Docker. Також було налаштовано CI/CD процес, який автоматично оновлює застосунок до останньої версії у Google Cloud за допомогою GitHub Actions. Загалом, рішення розмістити проєкт у хмарі було пов'язане з гнучкістю і легкістю налаштування необхідної потужності. Порівняно просто можна налаштувати оновлення застосунку за допомогою процесів Continuous Integration і Continuous Delivery. Також потрібно згадати про те, що застосунок є доступним, зручно показувати демо-версію та налаштовувати необхідні доступи. Google надає певний кредит для навчальних цілей, тож дуже вчасно і ефективно ми використали його для тестування цього рішення.

Kubernetes — платформа з відкритим кодом для автоматизації розгортання, масштабування та управління контейнеризованими додатками. Розроблена та активно підтримується компанією Google. Цей інструмент має забезпечити гнучкість у розгортанні та можливість масштабування за потреби. Система автоматично зможе підлаштовуватись під навантаження, а також забезпечує необхідні оновлення і резервні копії станів застосунку. Для повноцінної роботи із Kubernetes потрібен спеціальний інструмент, який має підготувати образ застосунку для роботи. Оновлення цього образу у сховищі дасть команду кластеру на перезбір.

Postgres як реляційну базу даних було вибрано з огляду на її надійну роботу, підтримку шардінгу та простоту реплікацій у разі необхідності. У майбутньому можливе використання особливостей цієї бази даних, зокрема підтримки enum, для розподілу ролей або підтримки певного переліку мов, доступних до перекладу. Реляційна база даних має нормалізовану структуру даних, які будуть зберігатись у ній. Усі записи й зв'язки між даними досить очевидні та прямолінійні. Варіант NoSQL бази даних також розглядали, адже він мав би певну перевагу при зчитуванні: можна було простіше створити запит на отримання даних для проєкту у заданій мові, але процес запису і налаштування зв'язків потребував би більше дій і контролю.

На нинішньому етапі імплементації застосунку база даних має такі таблиці як структурні елементи:

– users містить дані про зареєстрованих користувачів системи — first_name, last_name, email, password_hash, created_at;
– organizations містить дані про створені користувачами організації — name, owner_id, created_at;
– projects містить дані про створені користувачами проєкти, які належать конкретній організації, — name, description, org_id, created_at;
– translations містить дані про листи перекладів, створені та доступні конкретному проєкту, — project_id, language, updated_at, created_at;
– translationKeys містить дані про перелік ключів і їх перекладів для конкретного листа перекладу — sheet_id, key, value, updated_at, created_at.

У процесі розвитку проєкту передбачено певні зміни у структурі та зв'язках між таблицями. Схему бази даних подано на рис. 3.

Щоб окреслити функціональні можливості вебзастосунку EchoLocal, розглянемо основні сторінки інтерфейсу. На момент написання доступною була лише одна роль користувача, який реєструється і має доступ до всього функціоналу. У подальшому заплановано введення більш гранулярного доступу.

Портал EchoLocal умовно розділений на дві частини залежно від статусу користувача — авторизований він чи ні. Головна сторінка доступна для всіх і має містити загальну інформацію про сервіс. Для неавторизованого користувача доступні сторінки входу та реєстрації (рис. 4).

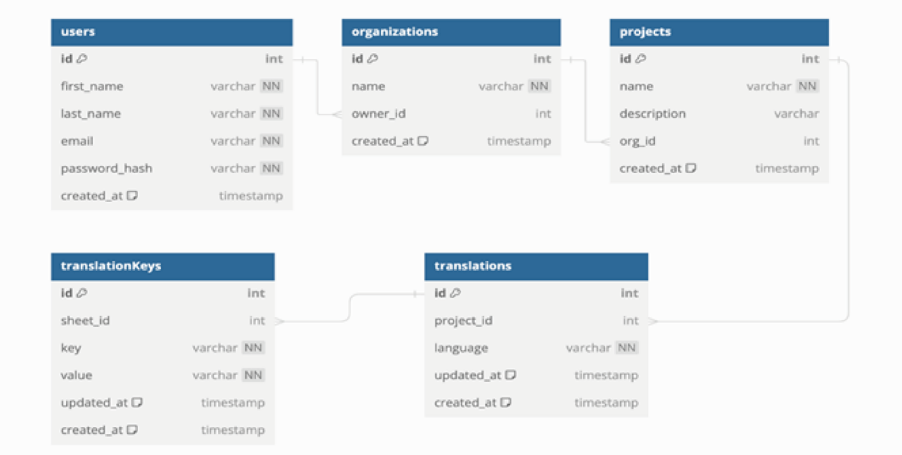

Рис. 3. Схема бази даних

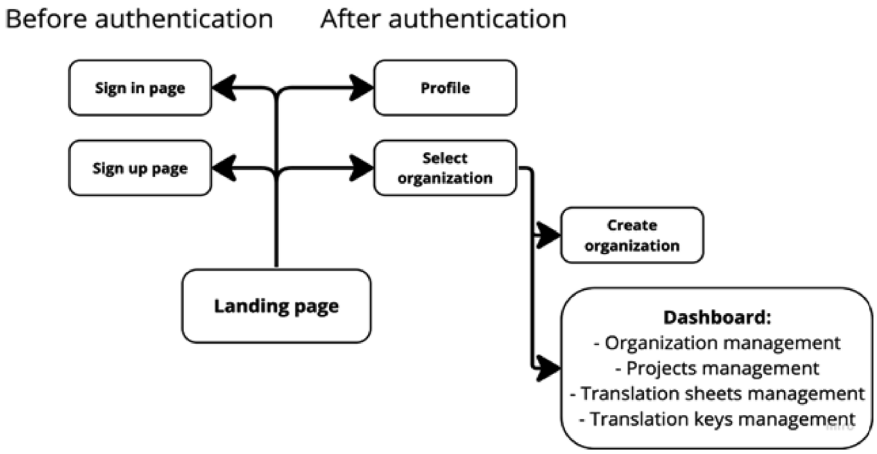

Рис. 4. Головна сторінка

Якщо користувач авторизований, то він має повний доступ до системи — до сторінки профілю, вибору та створення організації та панелі керування — сторінка, яка містить все необхідне для взаємодії з сервісом. Далі розглянемо кожну сторінку та її функціональність окремо для більшого розуміння загальної функціональності та можливостей системи.

На головній сторінці відображається простий інтерфейс із логотипом EchoLocal та меню навігації, яке дає змогу неавторизованому користувачеві зареєструватись або увійти до системи, а авторизованому — переходити між сторінками системи. Варто зазначити, що візуальна частина сторінки не готова і показує лише те, що сторінка у процесі розроблення, а також кнопку перевірки статусу сервера.

Після входу в систему користувач потрапляє на сторінку вибору організації. Тут можна вибрати наявну організацію або створити нову.

Сторінка налаштувань організації дає можливість користувачу керувати організацією, зокрема видаляти її.

На панелі керування користувач може налаштовувати проєкти і переклади. Це головна сторінка сервісу, де відбувається основна взаємодія і доступний такий функціонал: вибір проєкту (можливість вибрати проєкт зі спадного списку), список перекладів (перегляд наявних перекладів для вибраного проєкту), вибір листа перекладу (вибір конкретного листа перекладу для детальної роботи із ним), додавання нового перекладу (користувач може додати новий ключ перекладу та його значення), зміна та видалення наявного перекладу.

Користувач може створювати проєкти з власним набором мов і деталями та редагувати їх у подальшому.

Огляд функціональних можливостей EchoLocal дає уявлення про те, як саме користувач взаємодіє із системою і які можливості вона надає для управління процесом локалізації.

Переваги використання EchoLocal

EchoLocal підтримує різні технології та платформи, що дає можливість легко інтегрувати сервіс у будь-який проєкт. Завдяки можливостям масштабування він зручний як для малих стартапів, так і для великих компаній із багатьма проєктами та мовами. Всі ключі локалізації та переклади зберігаються в одному місці, що забезпечує легкий доступ та управління. Централізований контроль дає змогу уникнути дублювання зусиль і помилок, забезпечуючи єдність контенту. Завдяки можливостям синхронізації в реальному часі всі зміни в локалізації миттєво відображаються в проєкті. API сервісу дозволяє автоматизувати процес завантаження та оновлення перекладених текстів, знижуючи потребу в ручній праці. Власник проєкту може легко додавати та керувати ролями членів команди, забезпечуючи ефективну співпрацю між розробниками, перекладачами та менеджерами. Інтуїтивно зрозумілий інтерфейс дає змогу кожному члену команди швидко зрозуміти свої задачі та розпочати роботу. Наявність коментарів і контексту для перекладачів значно підвищує якість перекладів, зменшуючи кількість помилок та непорозумінь. Можливість затвердження перекладів менеджерами забезпечує додатковий контроль якості.

Сервіс EchoLocal може стати хорошим інструментом для управління локалізацією, який забезпечує гнучкість, ефективність та високу якість перекладів. Він дає змогу знизити витрати й час на локалізацію, підвищуючи при цьому загальну ефективність роботи команди. Використовуючи його, компанії можуть легко адаптувати свої продукти під нові ринки, розширюючи свою аудиторію та збільшуючи доходи.

Майбутні напрями розвитку та удосконалення EchoLocal

Звичайно, у рамках цього дослідження не було реалізовано всі ідеї. Основний наголос було зроблено на дослідження проблеми та перевірки методів можливого покращення і підвищення ефективності крос-командної співпраці у роботі над локалізацією вебзастосунків.

Ми визначили такі напрями покращення сервісу:

1. Можливість спільної роботи декількох користувачів на перекладом одного проєкту.
2. Введення ролей користувачів і закріплення за ними певних можливостей, які доступні на порталі, для покращення контролю за доступом. Це стане більш актуальним зі збільшенням аудиторії та розмірів проєктів.
3. Суттєвим покращенням сервісу було б додавання перекладу у реальному часі. Це можливо зробити з ускладненням підключення сервісу та проєкту за допомогою WebSockets або long-polling підходів. Можна запропонувати створення спеціально налаштованих бібліотек для популярних технологій, які вже мали такий функціонал.
4. Можливість попереднього перегляду сторінки з певною локалізацією. Це допомогло б виявити проблеми зі структурою сторінки, довжиною тексту та інші можливі невідповідності.

Висновки

У цій статті розглянуто різні підходи до локалізації сучасних вебзастосунків, їхні переваги та недоліки, а також концепцію нового сервісу для локалізації — EchoLocal. Сервіс покликаний покращити процес і зайняти свою нішу серед подібних рішень на ринку. Детально представлено архітектуру прототипу та функціонал сервісу.

Сервіс EchoLocal було розроблено з урахуванням основних недоліків наявних підходів, він надає такі переваги:

1. Підтримує різні технології та платформи, що дає змогу легко інтегрувати сервіс у будь-який проєкт. Прототип не залежить від технологій і платформ, адже надає API для роботи, яка своєю чергою може бути побудована різним чином — налаштована вручну або із використанням сторонніх бібліотек. Завдяки можливостям масштабування, EchoLocal підходить як для малих стартапів, так і для великих компаній із багатьма проєктами та мовами.
2. Усі ключі локалізації та переклади зберігаються в одному місці, що забезпечує легкий доступ та управління. Це допомагає уникнути повторення помилок і неефективної співпраці, забезпечуючи цілісність контенту.

3. Можна легко додавати та керувати ролями членів команди, забезпечуючи ефективну співпрацю між розробниками, перекладачами та іншими менеджерами. Інтуїтивно зрозумілий інтерфейс дає змогу кожному члену команди швидко зрозуміти свої задачі та розпочати роботу.
4. Наявність нотаток і контексту для перекладачів значно підвищує якість перекладів, зменшуючи кількість помилок та непорозумінь. Можливість затвердження перекладів менеджерами забезпечує додатковий контроль якості.
5. Завдяки можливостям синхронізації в реальному часі, всі зміни в локалізації миттєво відображаються в проєкті. API сервісу дає можливість автоматизувати процес завантаження та оновлення перекладених текстів.

EchoLocal, як сучасний інструмент для управління локалізацією, забезпечує гнучкість, масштабованість і високу якість перекладів, що робить його ідеальним рішенням для компаній, які прагнуть ефективно локалізувати свій контент і вийти на нові ринки. Однак подальші дослідження та розвиток технологій залишаються важливими для подолання майбутніх викликів.

Список літератури

1. Глибовець А. М. Агентно-базовані програмні системи пошуку та аналізу інформації : монографія / А. М. Глибовець. — Київ : Видавничий дім «Києво-Могилянська академія», 2019. — 281 с.
2. Документація сервісу Contentful [Електронний ресурс]. — Режим доступу: https://www.contentful.com/help/contentful-101/.
3. Документація сервісу Gtranslate [Електронний ресурс]. — Режим доступу: https://gtranslate.io/docs/58-gtranslate-tdn-documentation.
4. Документація сервісу Localazy [Електронний ресурс]. — Режим доступу: https://localazy.com/docs/general/getting-started-with-localazy.
5. Документація Gatsby [Електронний ресурс]. — Режим доступу: https://www.gatsbyjs.com/docs.
6. Документація GitHub Actions [Електронний ресурс]. — Режим доступу: https://docs.github.com/en/actions.
7. Документація Google Cloud Artifact Registry [Електронний ресурс]. — Режим доступу: https://cloud.google.com/artifact-registry/docs.
8. Документація Google Cloud Kubernetes Engine [Електронний ресурс]. – Режим доступу: https://cloud.google.com/kubernetes-engine/docs.
9. Документація Google Cloud SQL [Електронний ресурс]. — Режим доступу: https://cloud.google.com/sql/docs.
10. Документація Next.js [Електронний ресурс]. — Режим доступу: https://nextjs.org/docs.
11. Інтернаціоналізація та локалізація [Електронний ресурс]. — Режим доступу: https://qalight.ua/baza-znaniy/internatsionalizatsiya-ta-lokalizatsiya/.
12. Офіційний сайт Docker [Електронний ресурс]. — Режим доступу: https://www.docker.com/.
13. Офіційний сайт GitHub [Електронний ресурс]. — Режим доступу: https://github.com/.
14. Офіційний сайт Google Cloud Platform [Електронний ресурс]. — Режим доступу: https://cloud.google.com/.
15. Офіційний сайт Kubernetes [Електронний ресурс]. — Режим доступу: https://kubernetes.io/uk/.
16. Cheng R. How to choose a localization approach for your React application [Electronic resource] / Raymond Cheng. — 2022. — Mode of access: https://www.plasmic.app/blog/react-localization.
17. Krukowski I. Internationalization vs. localization (i18n vs l10n): The differences [Electronic resource] / Ilya Krukowski. — 2022. — Mode of access: https://lokalise.com/blog/internationalization-vs-localization/.

References

Cheng, R. (2022). *How to choose a localization approach for your React application.* https://www.plasmic.app/blog/react-localization/.

Contentful service documentation. (n.d.). ttps://www.contentful.com/help/contentful-101/.

Gatsby documentation. (n.d.). https://www.gatsbyjs.com/docs.

GitHub Actions documentation. (n.d.). https://docs.github.com/en/actions.

Google Cloud Artifact Registry documentation. (n.d.). ttps://cloud.google.com/artifact-registry/docs.

Google Cloud Kubernetes Engine documentation. (n.d.). https://cloud.google.com/kubernetes-engine/docs.

Google Cloud SQL documentation. (n.d.). https://cloud.google.com/sql/docs.

Gtranslate service documentation. (n.d.). https://gtranslate.io/docs/58-gtranslate-tdn-documentation/.

Hlybovets, A. M. (2019). *Ahentno-bazovani prohramni systemy poshuku ta analizu informatsii* (p. 281). Vydavnychyi dim “Kyievo-Mohylianska akademiia” [in Ukrainian].

Internatsionalizatsiia ta lokalizatsiia. (n.d.). https://qalight.ua/baza-znaniy/internatsionalizatsiya-ta-lokalizatsiya/.

Krukowski, I. (2023). *Internationalization vs. localization (i18n vs l10n): What’s the difference?* https://lokalise.com/blog/internationalization-vs-localization/.

Localazy service documentation. (n.d.). https://localazy.com/docs/general/getting-started-with-localazy/.

Next.js documentation. (n.d.). https://nextjs.org/docs.

Official Docker website. (n.d.). https://www.docker.com/.

Official GitHub website. (n.d.). https://github.com/.

Official Google Cloud Platform website. (n.d.). https://cloud.google.com/.

Official Kubernetes website. (n.d.). https://kubernetes.io/.

V. Vereta, V. Tkachenko

AUTOMATED LOCALIZATION OF APPLICATIONS IN MICROSERVICE ARCHITECTURE

The article provides an in-depth analysis of various tools and services available for the localization and translation of web applications, as well as methods and approaches for their integration and scaling. It examines existing solutions for managing web service localization, highlighting their strengths and weaknesses. The article also describes an architecture that allows for the quick adaptation of services to meet the diverse needs of different users and projects. This architecture facilitates efficient integration with various platforms, while ensuring ease of scaling.

Based on this architecture, the EchoLocal web service was developed. EchoLocal enhances interaction among participants in the web application localization process, allowing for optimal integration into distributed microservice architectures common in modern applications. It supports the easy deployment within a closed ecosystem, making it suitable for organizations with specific security or operational requirements. EchoLocal enables users to integrate and manage localizations of their products on one centralized platform in real-time.

EchoLocal helps reduce localization costs by simplifying the adaptation of content for different language markets. It increases the efficiency of communication between all participants in the localization process, fostering stronger collaboration. By centralizing localization management, EchoLocal streamlines workflows for developers, translators, and project managers alike.

In conclusion, the article underscores the importance of efficient localization in today's global market and presents EchoLocal as a robust solution to many challenges in web application localization. Implementing the described architecture allows organizations to significantly improve their localization processes, leading to faster market entry and a more enhanced user experience across different regions.

Keywords: localisation, internationalisation, globalisation, microservice architecture, Kubernetes, Google Cloud Platform, GCP, CI/CD.

Матеріал надійшов 18.09.2024