УДК 681.3(045)

О. Д. Азаров
О. І. Черняк
Д. О. Смольц
Ж. А. Мельник

ПЛАТФОРМА ДЛЯ ЛОКАЛІЗАЦІЇ ВЕБ ТА МОБІЛЬНИХ ДОДАТКІВ

Вінницький національний технічний університет

Анотація

У статті описано актуальність локалізації, а також існуючі способи та шляхи її реалізації. Розглянуто дослідження та статистичні дані світових науково-популярних джерел. Наведено оптимальний спосіб вирішення мовної проблеми у вигляді запропонованої універсальної платформи для локалізації веб та мобільних додатків на основі багатошарової клієнт-серверної архітектури. Використано новітні технології та інструменти, що в свою чергу дозволило зробити розроблювану систему легкою для масштабування, надійною та кросплатформеною.

Ключові слова: локалізація, переклад, глобалізація, сервіс, лінгвістика, програмування, платформа, мережа.

Abstract

The article analyzes the problem of localization describes its relevance at present, as well as known methods and ways of its implementation. The latest researches and statistical data of world popular scientific sources are considered. The optimal way of solving the language problem is presented in the form of the proposed universal platform for localization of the Web and mobile applications based on the multilayered client-server architecture. Used the latest technologies and tools, which in turn allowed the developed system to be easy to scale, reliable and cross-platform.

Keywords: localization, translation, globalization, service, linguistics, programming, platform, network.

Вступ

Сучасні інформаційні технології увібрали в себе лавиноподібні досягнення електроніки, а також математики, філософії, психології та економіки. В умовах сучасного темпу розвитку IT суміжні сфери отримують поштовх до розповсюдження та популяризації:

- Наукова діяльність. Вчені по всьому світу отримують способи для комунікації, новітні програмні та апаратні інструменти для проведення досліджень, моделювання експериментальних умов, здійснення нових відкриттів.
- Освітні програми. Студенти та учні мають доступ до навчальних матеріалів із невичерпних джерел інформації з будь якої частини землі, що призводить до підвищення їх кваліфікації в будь-якій сфері.
- Бізнес. Люди мають змогу розвивати свій бізнес та робити продукт своєї діяльності доступним за межами власної країни.

Таких сфер діяльності як і прикладів використання інформаційних технологій у них можна назвати безліч і очевидним є те, що з IT людство прогресує у набагато швидшому темпі.

Актуальність локалізації

На даний момент, кількість нативних носіїв мови у світі не співпадає із кількістю продуктів цією мовою, що видно з діаграм, представлених на рисунках 1 і 2. На це є декілька причин:

- розвиненість країни, де спілкуються цією мовою
- складність мови
- освіченість населення і т.д.

На жаль, це призводить до труднощів та незручностей. На дану тему було проведено багато досліджень, з метою визначити на скільки комфортніше сприймати інформацію своєю мовою. Ось деякі висновки з них:

- 80% користувачів, які користуються мобільними додатками не англомовні [1];
- локалізовані версії сайтів та додатків мають на 42% більший коефіцієнт кліків та на 22% більше переходів ніж одномовні [2];
- 72% покупців стверджують, що вони здійснюють покупки лише у тих магазинах, де є опис товару їхньою мовою (Common Sense Advisory) [3]
- 56% покупців вважають наявність перекладу на їхню мову важливішою за ціну продукту (Common Sense Advisory) [3]

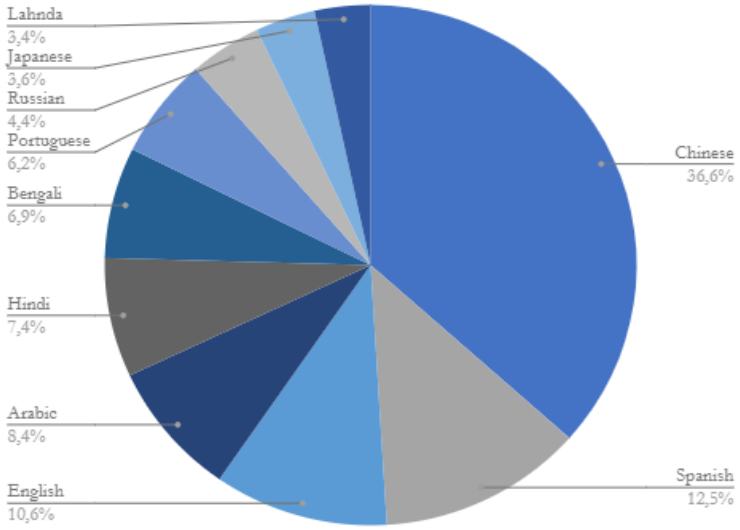

Рис. 1. Популярність мови у світі

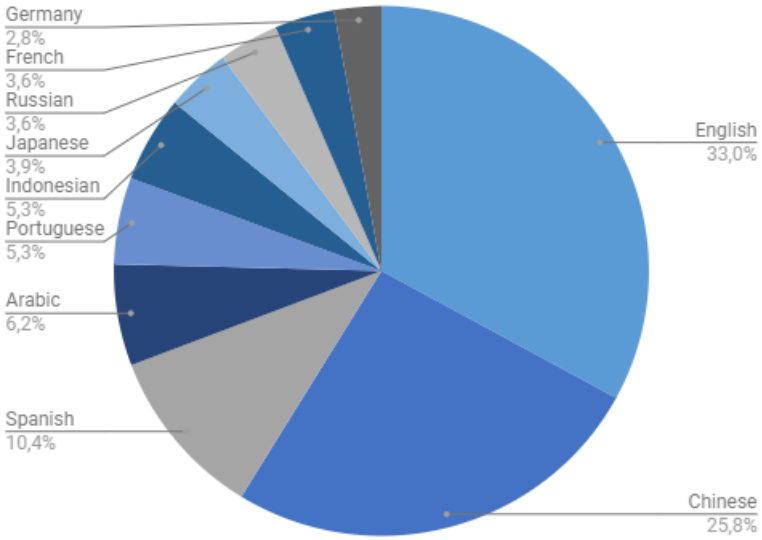

Рис. 2. Популярність мови в інтернеті

Наочно продемонструвати актуальність локалізації певного сайту можна за допомогою формули 1:

$$T = \frac{a_u - a_n}{k_w} \quad (1)$$

де a_u – це загальна кількість користувачів цього сайту; a_n – це кількість носіїв мови сайту серед користувачів; k_w – це коефіцієнт популярності мови в світі, який в свою чергу визначається за формулою 2:

$$k_w = \frac{L}{100} \quad (2)$$

де L – популярність мови у світі, яку можна побачити на рисунку 1.

Беручи до уваги формулу 1, можна зрозуміти, що якщо актуальність локалізації напряму залежить від кількості іноземних користувачів цього сайту та обернено від популярності мови у світі.

Зважаючи на це, стає зрозуміло, що локалізація – це невід'ємна частина успіху програмного продукту.

Способи та шляхи локалізації

Оскільки питання локалізації не нове, його вже давно намагаються вирішити. На даний момент існує декілька різних за складністю виконання та оптимізованістю варіантів реалізації.

Найпростіший варіант – створити декілька HTML сторінок (для кожної мови окремо) та додати можливість перемикати мову переходячи на іншу сторінку. Приклад реалізації такого сайту на рисунку 3.

```
<a href="file.html" lang="fr" hreflang="fr">Francais</a>
```

Рис. 3. Приклад реалізації сайту за допомогою кількох HTML сторінок

Даний метод дуже простий і не потребує ніяких особливих навичок. Але він зовсім не оптимізований. При переході на іншу мову доведеться повністю перезавантажувати всю HTML сторінку, а отже і всі ресурси. До того ж це неймовірно важко для розробників – вирішивши змінити верстку доведеться змінювати її в кожному HTML файлі.

Наступний спосіб – використання файлів локалізації. Файлом локалізації може виступати файл у форматах JSON, XML, YAML, CSV та інші [4]. Головна вимога до такого файлу наявність пари ключ-значення, де ключ буде ідентифікатором стрічки і буде однаковим для кожної мови, а значення буде перекладом стрічки на певну мову. Приклади файлів локалізації у форматі JSON (див. рисунок 4) та XML (див. рисунок 5).

```
"ukrainian": {
  "greeting": "Привіт!",
  "excuse": "Вибач"
},
"esperanto": {
  "greeting": "Hello!",
  "excuse": "Sorry!"
}
```

Рис. 4. Файл локалізації у форматі JSON

Наступним кроком файл локалізації під'єднується до HTML сторінки. Це можна зробити декількома способами (за допомогою PHP, JavaScript та/або Ajax). Використання файлу локалізації суттєво спрощує процес розробки, підтримки та використання продукту: при необхідності змінити структуру документу потрібно зробити мінімум змін, при зміні мови перезавантажується лише файл локалізації (розмір якого в десятки менший ніж розмір цілого HTML файлу). Приклад використання файлу локалізації за допомогою AJAX (див. рисунок 6). Як видно з прикладу потрібно звернутись до потрібно поля по ключу і в залежності від обраної мови підставиться потрібний переклад.

```
<language name="Ukrainian">
  <data name="greeting">
    <value>Привіт!</value>
  </data>
  <data name="excuse">
    <value>Вибач</value>
  </data>
</language>
<language name="English">
  <data name="greeting">
    <value>Hello!</value>
  </data>
  <data name="excuse">
    <value>Sorry!</value>
  </data>
</language>
```

Рис. 5. Файл локалізації у форматі XML

```
$(document).ready(function(){
    $('#div1').text(local.greeting);
    });
```

Рис. 6. Приклад використання файлу локалізації за допомогою AJAX

Архітектура системи

В основу розроблюваної системи покладена клієнт-серверна архітектура, яка має ряд переваг відносно відомих програмних засобів локалізації (зниження об'єму даних, що передаються, ізоляція доступу користувача до даних та логіки обробки запитів, можливість додавання клієнтів іншого типу) [5]. У свою чергу клієнт та сервер також не монолітні. Для підсилення абстракції, зменшення кількості коду та збільшення продуктивності вони побудовані на основі патернів проектування [6]. Крім того, вони поділені на шари, що підсилює інкапсуляцію, робить систему інтуїтивно зрозумілою та спрощує процес її тестування. Далі наведено запропоновану авторами структуру шарів клієнта і сервера.

Шари клієнта:

- шар представлення – відповідає за зовнішній вигляд додатка;
- проміжний шар – формує запити для сервера, та обробляє відповіді.

Шари сервера:

- проміжний шар – являє собою "ворота", для доступу до сервера та перевіряє права користувача;
- шар логіки – містить у собі всю обробку даних та взаємодіє із віддаленим сервером для машинного перекладу;
- шар роботи з даними – ізолює стандартні операції роботи з даними (створення, редагування, читання та видалення).

Окремим модулем архітектури представлене сховище, що містить реляційну базу даних для збереження всіх даних у відношеннях, нереляційну документо-орієнтовану базу даних для збереження текстів та перекладів, а також сховище бінарних файлів та пошуковий сервер.

Окрім того особливістю запропонованої системи є використання сторонніх хмарних сервісів, а саме Firebase для аутентифікації користувача в системі та Google Translate API для здійснення машинного перекладу.

Програмна реалізація

Для реалізації серверної частини був обраний .NET Core Framework та мова C#. Особливістю даного фреймворку є кросплатформеність та оптимізація порівняно із .NET Framework. У якості ORM використовується Entity Framework Core [7], який спрощує процес взаємодії із реляційною базою даних (створення таблиць із класів моделей, підтримка міграцій) та MongoDB Driver для NoSQL бази даних. Обмін даними реалізований за допомогою надсилання HTTP запитами серіалізованих даних у форматі JSON, що набагато оптимізованіший ніж його класичний аналог XML.

Для реалізації клієнта використовувався Angular 6. Клієнт запрограмований на TypeScript, який є наступною сходинкою розвитку Javascript і на відміну від свого "батька" має строгу типізацію, краще реалізує концепції ООП та дозволяє підтримувати та масштабувати код значно легше. Для інтерфейсу користувача використовується HTML + SASS.

Для збереження даних використовуються MS SQL (всі реляційні дані), MongoDB (документи з текстом та перекладами), Azure Blob Storage (картинки проектів та вкладені до тексту файли). Також для швидкого пошуку по документах використовується пошуковий сервер ElasticSearch, для розгортання якого був використаний Docker.

Завдяки бібліотеці SignalR досягнуто обмін даних між сервером і клієнтом у реальному часі (без оновлення сторінки).

Машинний переклад реалізований за допомогою хмарного сервісу – Google Translate API, дані на який пересилають за допомогою HTTP запитів.

Використовуючи Firebase, як постачальника аутентифікації, з'явилась можливість безпечної авторизації в системі за допомогою електронної пошти або акаунтів соцмереж: Google та Facebook [8].

Також в програму додано спостерігання за прогресом виконання перекладу. Щоб реалізувати таку можливість доцільно було звернутись до формули 3.

$$P = \frac{\sum (l - n_i)}{s \cdot l} \cdot 100\% \qquad (3)$$

де l – це кількість мов у проекті; n – це кількість неперекладених мов для стрічки; s – це кількість стрічок у проекті; i – це певна стрічка.

Висновки

Усі описані засоби дозволили зробити систему сучасною, надійною, кросплатформеною, багатофункціональною, а головне – корисним інструментом для власників веб та мобільних додатків та перекладачів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Google Play improves all around, still trails iOS in revenue [Електронний ресурс]. – Режим доступу: https://www.androidauthority.com/q2-2014-app-stats-show-google-play-ios-improve-407106/.
2. Language Localization: Pushing Mobile Ad Conversion Across the Globe [Електронний ресурс]. – Режим доступу: http://www.adotas.com/ 2014/04/language-localization-pushing-mobile-ad-conversion-across-the-globe/.
3. Survey of 3,000 Online Shoppers Across 10 Countries Finds that 60% Rarely or Never Buy from English-only Websites [Електронний ресурс]. – Режим доступу: http://www.commonsenseadvisory.com/Default.aspx?Aid=21500&Contenttype=ArticleDet&tabID=64.
4. Why JSON Is Better Than XML [Електронний ресурс]. – Режим доступу: https://blog.cloud-elements.com/json-better-xml.
5. Bass, Len. Software architecture in practice / Len Bass, Paul Clements, Rick Kazman. 3rd ed. p. cm. – (SEI series in software engineering). Includes bibliographical references and index. ISBN 978-0-321-8 1 573-6.
6. Fowler, Martin, 1963- Patterns of enterprise application architecture / Martin Fowler. p. cm. Includes bibliographical references and index. ISBN 0-321-12742-0.
7. Entity Framework Core [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/en-us/ef/core/.
8. Firebase by platform [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/?hl=ru.

Азаров Олексій Дмитрович – докт. техн. наук, професор, професор кафедри обчислювальної техніки, декан ФІТКІ, Вінницький національний технічний університет, м. Вінниця.
Черняк Олександр Іванович – канд. техн. наук, доцент, доцент кафедри обчислювальної техніки, Вінницький національний технічний університет, м. Вінниця, e-mail: chernyak@vntu.edu.ua.
Смольц Дмитро Олександрович – студент групи 1КІ-15б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: dima.smolts@gmail.com.
Мельник Жанна Анатоліївна – студентка групи 1КІ-15б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: 1ki.15bmelnyk@gmail.com.

Oleksiy D. Azarov – Doct. Sc. (Eng.), Professor, Professor of the Computer Techniques Chair, Dean of the ITKI faculty. Vinnytsia National Technical University, Vinnytsia.
Oleksandr I. Chernyak – Cand. Sc. (Eng.), Assistant Professor of the Computer Techniques Chair, Vinnytsia National Technical University, Vinnytsia, e-mail: chernyak@vntu.edu.ua.
Dmytro Smolts – Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: dima.smolts@gmail.com.
Zhanna Melnyk – Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: 1ki.15bmelnyk@gmail.com.