УДК 004.9

*МУСАТОВ Д.С.*

# АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВЗАЄМОДІЇ З КРОСПЛАТФОРМЕННИМИ МЕСЕНДЖЕРАМИ НА ПРИКЛАДІ TELEGRAM

У даній статті розглянуто використання Microfrontend підходу до розробки клієнтських частин програмних систем на прикладі проектування програмної системи для взаємодії з кросплатформенними месенджерами на прикладі Telegram, а саме програмної системи для публікації контенту на канали месенджеру Telegram. Описано особливості Microfrontend підходу та детальну схему його роботи, включаючи зв'язки між компонентами клієнтської частини програмної системи. Визначено новизну пропонованого архітектурного рішення та його відмінності від існуючих рішень.
КЛЮЧОВІ СЛОВА: Telegram, Microfrontend, мікросервісна архітектура, Svelte, Module Federation.

This article considers the use of Microfrontend approach to the development of client side of software systems on the example of designing a software system for interaction with cross-platform messengers on the example of Telegram, namely a software system for publishing content on Telegram messenger channels. The features of the Microfrontend approach and the detailed scheme of its work are described, including the connections between the components of the client part of the software system. The novelty of the proposed architectural solution and its differences from existing solutions are determined.
KEYWORDS: Telegram, Microfrontend, microservice architecture, Svelte, Module Federation.

## Вступ

З розвитком людської цивілізації все більшою стає проблема складності керування інформацією у постійно зростаючому інформаційному просторі. Інформаційна лавина нової ери засобів масової інформації з кожним днем рухається все більш невпинно, через що ЗМІ розширили свої володіння не лише на публіцистичні видання та телебачення. Вони проникли і на простори мережі Інтернет. Поширення інформації не оминуло і системи миттєвого обміну повідомленнями (месенджери), що раніше використовувались лише для спілкування між людьми. Тепер месенджери теж являють собою окремі засоби масової інформації.

Метою даної роботи є створення програмного забезпечення для покращення процесу публікації дописів на канали месенджеру Telegram, а основним завданням є впровадження новітніх архітектурних підходів – Microfrontend підходу до розробки клієнтської частини програмної системи та мікросервісного підходу для розробки серверної частини програмної системи.

## 1. Загальний огляд месенджеру Telegram та його складових

Telegram – це кросплатформенна система миттєвого обміну повідомленнями (месенджер), що прив'язана до телефонного номеру користувача [1]. Робота месенджера побудована таким чином, що в ньому реалізована як функція миттєвого обміну повідомленнями, так і функції, притаманні примітивному мобільному зв'язку. Широке коло користувачів Telegram утворилось не лише внаслідок високої функціональності месенджеру, а й через високу безпеку даних, що реалізована у месенджері.

Telegram має високу функціональність: він застосовується не лише для підтримання зв'язку з іншими користувачами через листування або дзвінки, а й для поширення інформації широкому колу людей. Таку функцію надають можливості створення групових чатів та Telegram-каналів.

Telegram-канал являє собою інструмент для передачі повідомлень широкій аудиторії. Кількість людей, що можуть стати підписниками (постійними читачами) каналу не обмежена засобами Telegram, а створити такий канал може будь-який користувач. Система підтримки каналів включає не лише простоту керування ними, а й вбудовані статистичні інструменти, що допомагають адміністраторам каналів відслідковувати

зацікавленість аудиторії у своєму контенті.

Процес публікації допису на канали Telegram включає декілька етапів, серед яких виділяють підготовку ідеї допису, написання та редагування текстового матеріалу, підготовка графічного матеріалу за необхідності, власне процес публікації (об'єднання текстового та графічного матеріалів) та постпублікаційний аналіз. Для забезпечення процесу публікації дописів Telegram надає користувачам вбудовані інструменти для підготовки дописів. Серед цих інструментів виділяють текстовий та графічний редактори, файловий редактор тощо. Проте, кожен з цих інструментів має певні обмеження у своїй роботі, в зв'язку з чим незалежні розробники створюють сервіси, головним завданням яких є поліпшення процесу публікації дописів на канали Telegram.

## 2. Загальний огляд Microfrontend підходу

Microfrontend підхід – унікальний підхід до розробки клієнтських частин програмних систем, що використовує поділ на незалежні програмні компоненти для реалізації їх автономності та можливості асинхронної розробки клієнтських частин програмних систем [2].

Поява Microfrontend підходу як основної ланки розвитку сучасного процесу розробки клієнтських частин програмних систем обумовлена виникненням значних складнощів, зумовлених монолітністю застарілих підходів до розробки клієнтських частин програмних систем. Наслідком монолітності ставало виникнення значних проблем в роботі клієнтських частин програмних систем, важкості у їх тестуванні.

З метою виправлення усіх недоліків монолітного підходу до фронтенд-розробки було засновано поділ монолітних клієнтських частин програмних систем на окремі мікрокомпоненти, що містив в собі варіант реалізації популярного на сьогоднішній день та практичного у застосуванні мікросервісного підходу до розробки серверних частин програмних систем. Поділ клієнтських частин на окремі компоненти (мікрофронтенди) дозволив втілити можливість незалежної, паралельної, асинхронної розробки цілісних клієнтських частин програмних систем [3].

## 3. Опис роботи клієнтської частини програмної системи при застосуванні Microfrontend підходу

Перш за все, слід надати визначення певних понять, що в подальшому будуть часто використовуватись в описі [4]:

1) Shell (Application Shell) – це центральний елемент клієнтської частини програмної системи, який доступний користувачеві. Можна сказати, що це та точка, в якій відбувається взаємодія користувача з програмною системою.
2) Host – це окремий компонент, який запускає у собі усі компоненти клієнтської частини програмної системи для майбутньої передачі їх у скомпонованому вигляді (Current Application Build) в Shell. Host за своєю суттю є середовищем розгортання системи, головним її модулем.
3) Current Application Build – це версія клієнтської частини програмної системи, що відображається користувачеві в Shell.
4) Module Federation – це плагін для Webpack 5, який дозволяє проводити експорт та обмін частинами окремих компонентів мікрофронтендів між собою та передавати їх в Host.
5) Remote module – це модуль, тобто найменша одиниця поширюваного коду, який компонент віддає іншому компоненту. Тобто частина мікрофронтенду, його компонент, що експортується в Host або інший мікрофронтенд.
6) Expose module – це модуль, тобто найменша одиниця поширюваного коду, який було імпортовано з іншого компонента. Тобто це частина мікрофронтенду, його компонент, що був імпортований з Host або іншого мікрофронтенду.
7) Route – це маршрут, або посилання, на необхідний модуль.

Тепер необхідно розглянути та описати загальну схему роботи клієнтської частини програмної системи, спроектовану за допомогою Microfrontend підходу.

Основним місцем дії в роботі системи можна визначити Host. Host є місцем збору та відтворення усіх необхідних для роботи

системи компонентів, експортованих з мікрофронтендів (Microfrontend Application на схемі). Відправною точкою для початку каскаду змін вважається дія користувача над Current Application Build, відображеним в Shell.

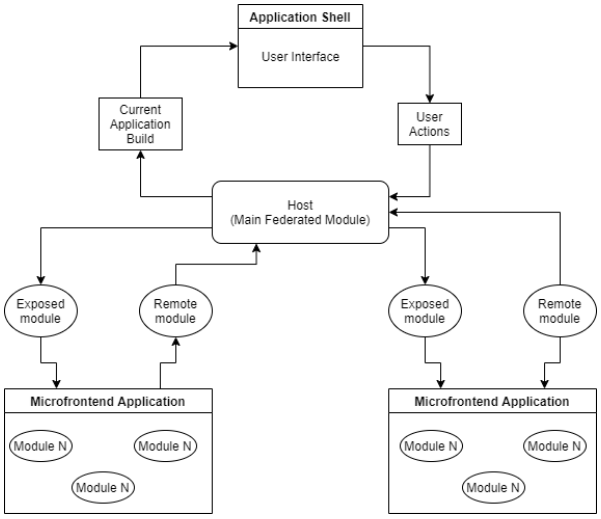


*Рис. 1. Схема робота клієнтської частини програмної системи, спроектованої з використанням Microfrontend підходу*

Як видно зі схеми, кожен мікрофронтенд включає у себе різноманітні модулі. Модулем у контексті Microfrontend підходу називають найменшу частину поширюваного коду, тобто компонент застосунку. Із зазначеного вище можна зробити висновок, що кожен мікрофронтенд є окремим незалежним застосунком, що має виконувати певні функції. Відповідно до потреб системи, в Host має бути поширений саме той компонент мікрофронтенду, який задовольняє потреби розробника та користувача. Host віддає запит на отримання необхідного йому компонента у відповідний мікрофронтенд, в якому цей компонент створено. Цей запит Host віддає мікрофронтенду у вигляді Exposed module. У даному випадку Exposed module представляє собою набір даних, які необхідні мікрофронтенду для відокремлення компоненту (модулю), який було запрошено. Коли Exposed module потрапляє всередину мікрофронтенду і обробляється ним, мікрофронтенд видає результат, а саме потрібний компонент. Результатом буде Remote module, що матиме певний Route. Remote module потрапляє у Host, обробляється ним та відображується у певному місці в структурі застосунку. Після отримання усіх компонентів необхідних для кінцевого збору користувацького інтерфейсу або після отримання зміненого компоненту Host передає Current Application Build в Shell, де її бачить користувач.

Важливо зазначити також деякі нюанси:

1) Реалізація експорту та імпорту компонентів між мікрофронтендами та Host забезпечується плагіном Module Federation.
2) Під віддачою та отриманням компонентів розуміється їх експорт та імпорт всередині програмної системи, а не перехід одного компонента з мікрофронтенду в Host і навпаки.
3) Враховуючи особливості організації компонентів при застосуванні Microfrontend підходу, з кожною новою дією користувача Current Application Build змінюється лише в тому компоненті, над яким користувач виконав дію, тобто відсутня необхідність перезавантаження усіх компонентів системи при виконанні користувачем певних дій. Саме цей аспект Microfrontend підходу обумовлює швидкодію процесу розробки клієнтських частин програмних систем.
4) Враховуючи те, що окремі компоненти містяться в різних мікрофронтендах, над кожним окремим мікрофронтендом може працювати окрема команда розробників, використовуючи свій окремий стек технологій, що забезпечує процес незалежної асинхронної паралельної розробки.
5) Перелічені у пунктах 3 та 4 особливості забезпечують одну з головних переваг Microfrontend підходу – можливість горизонтального масштабування процесу розробки, тобто здатність системи збільшувати функціонал, не змінюючи свою структуру. Умовно кажучи, можна створити нову функцію, виконувану системою, та підключити її до вже існуючої клієнтської частини без необхідності переробляти усю систему. Однією з небагатьох потрібних змін буде лише вибір розміщення доданого компоненту.

## 4. Новизна архітектурного рішення при створенні програмної системи.

Новизна архітектурного рішення при

створенні програмної системи для взаємодії з кросплатформенними месенджерами на прикладі Telegram, або програмної системи для публікації контенту на канали месенджеру Telegram полягає в наступному:

1) Спроектована система є першою такою системою, побудованою з використанням Microfrontend підходу до розробки клієнтської частини програмної системи. Дана система є першим прикладом застосування цього підходу для реалізації майбутнього програмного рішення.
2) Спроектована система реалізує у собі можливість горизонтального масштабування програмної системи, яку забезпечує Microfrontend підхід, що виливається у появу нових функцій, які раніше не використовувались в існуючих рішеннях.
3) Використання таких технологій розробки як Svelte та Effector, що раніше не використовувались для реалізації таких програмних системи, забезпечує вищі показники ефективності, продуктивності та стабільності системи, аніж в існуючих рішеннях.

## Заключення.

Метою даної роботи стало впровадження новітнього архітектурного рішення для побудови програмного забезпечення для взаємодії з кросплатформенними месенджерами на прикладі Telegram.

Було наведено загальний опис предметної області (месенджер Telegram та процес публікації дописів на його канали), вивчено усі особливості використання Microfrontend підходу до розробки клієнтської частини програмної системи та детально описано структуру клієнтської частини програмної системи при застосуванні Microfrontend підходу. Крім того, було виокремлено, в чому саме полягає новизна впровадженого архітектурного рішення до проектування системи.

## Список літератури.


1. Telegram (2021). *Telegram FAQ*. https://telegram.org/faq/.
2. Richards M., & Ford N. (2020). *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Media.
3. Gears M. (2020). *Micro Frontends - extending the microservice idea to frontend development.* https://micro-frontends.org/.
4. Webpack (2020). *Module federation | webpack*. https://webpack.js.org/concepts/module-federation/#use-cases.


УДК 004.055

*Калініченко В. С.,*
*Ковтунець О.В.*

## ВИМІРЮВАННЯ ЕФЕКТИВНОСТІ ВЕБ-ДОДАТКУ. АНАЛІЗ НАЙВАЖЛИВІШИХ ПОКАЗНИКІВ ЕФЕКТИВНОСТІ

Професійне управління продуктивністю в Інтернеті стало окремою галуззю, і це справедливо, оскільки продуктивність життєво важлива для підвищення коефіцієнта конверсії, утримання користувачів та інших показників. Проте процес вимірювання веб-продуктивності може набувати різних форм, і за всіх доступних показників веб-продуктивності може бути складно зрозуміти, які з них слід відстежувати. Ця стаття стане оглядом способів вимірювання веб-продуктивності, а також надасть додаткову інформацію та інструменти, які допоможуть покращити веб-продуктивність вашого сайту.
КЛЮЧОВІ СЛОВА: ОПТИМІЗАЦІЯ, ЧАС ВІДПОВІДІ, ПРОДУКТИВНІСТЬ.

Professional online performance management has become an industry of its own, and rightfully so, as performance is vital to improving conversion rates, user retention, and more. However, the process