СЕКЦІЯ: ОСВІТНІ СТРАТЕГІЇ ПІДГОТОВКИ ФАХІВЦІВ ІТ-ГАЛУЗІ

ОСОБЛИВОСТІ ВИКОРИСТАННЯ РІЗНИХ СТРАТЕГІЙ КЕШУВАННЯ ПРИ РОЗРОБЦІ БЛОГІВ НА ОСНОВІ ТЕХНОЛОГІЇ PWA

Базиволяк Максим Іванович
здобувач другого рівня вищої освіти спеціальності Комп'ютерні науки
Тернопільський національний педагогічний університет імені Володимира Гнатюка
bazyvolyak_mi@fizmat.tnpu.edu.ua
Шмигер Галина Петрівна
кандидат біологічних наук, доцент кафедри інформатики та методики її навчання
Тернопільський національний педагогічний університет імені Володимира Гнатюка
yava@fizmat.tnpu.edu.ua

У сучасних умовах розвитку вебтехнологій спостерігається зростання вимог до швидкодії, доступності та зручності вебзастосунків, зокрема блогів, які є важливим інструментом комунікації та контент-маркетингу. Згідно з сучасними тенденціями, користувачі очікують миттєвого завантаження сторінок, можливості переглядати контент у режимі офлайн, а також стабільної роботи незалежно від якості інтернет-з'єднання [1]. Для реалізації таких вимог ефективним підходом є використання технології Progressive Web Apps (PWA), яка поєднує переваги вебзастосунків і нативних мобільних програм. Завдяки застосуванню Service Worker, Cache Storage API та вебманіфесту, PWA забезпечують швидкий доступ до контенту, роботу в офлайн-режимі та покращену взаємодію з користувачем [2; 3].

Одним із ключових аспектів ефективності PWA є вибір стратегії кешування – тобто підходу до збереження та оновлення даних у кеші. Різні стратегії, такі як Cache First, Network First, Stale-While-Revalidate або їхні комбінації, визначають баланс між швидкодією, актуальністю контенту та стабільністю роботи застосунку [4; 5]. Неправильне налаштування кешу може призвести або до відображення застарілої інформації, або до втрати швидкодії при нестабільному з'єднанні, що особливо критично для блогових платформ, де оперативність публікацій відіграє ключову роль [6].

Сучасні інструменти, зокрема бібліотека Workbox, значно спрощують реалізацію різних стратегій кешування та дозволяють розробникам блогів оптимізувати обмін даними, забезпечуючи збереження статичних ресурсів (зображення, стилі, скрипти) і динамічних даних (пости, API-відповіді) [7]. Дослідження показують, що раціональне використання кешування у PWA сприяє не лише підвищенню продуктивності, але й зниженню енергоспоживання та покращенню користувацького досвіду [8].

Таким чином, актуальність даного дослідження зумовлена потребою у визначенні та порівнянні ефективності різних стратегій кешування під час розробки блогів на основі технології PWA. Оптимальний вибір кешувальної стратегії дозволяє забезпечити високу продуктивність, доступність контенту офлайн і стійкість вебзастосунку до мережевих збоїв.

Технологія Progressive Web Apps (PWA) є сучасним підходом до створення вебзастосунків, що поєднують у собі функціональні можливості нативних мобільних програм і доступність звичайних вебсайтів. Основу архітектури PWA становлять три ключові компоненти: Service Worker, Web App Manifest та HTTPS-з'єднання.

Service Worker – це незалежний від головного потоку сценарій, який працює у фоновому режимі та відповідає за обробку подій мережевих запитів, керування кешем, оновлення даних і реалізацію офлайн-режиму [1]. Саме цей компонент робить можливим використання різних стратегій кешування, що визначають поведінку застосунку при взаємодії з мережею.

Механізм кешування у PWA реалізується через Cache Storage API, який дозволяє зберігати ресурси (HTML, CSS, JavaScript, зображення, API-відповіді) для подальшого використання без повторного звернення до сервера. Залежно від типу контенту та цілей застосунку, розробник може обрати одну або кілька стратегій кешування. Найбільш поширеними є такі [2; 3]:

Cache First – при першому зверненні ресурс завантажується з мережі й зберігається у кеші. Надалі дані беруться з кешу, що забезпечує миттєве завантаження, але може призвести до використання застарілої інформації.

Network First – пріоритет віддається мережевому запиту; якщо мережа недоступна, застосунок використовує кешовану версію. Такий підхід актуальний для динамічного контенту, наприклад, стрічки блогу.

Stale-While-Revalidate – застосунок спочатку відображає кешовану версію, а потім асинхронно оновлює її з мережі. Це дозволяє поєднати швидкість завантаження з актуальністю даних.

Cache Only та Network Only – допоміжні стратегії, що використовуються у специфічних випадках, наприклад, для статичних зображень або запитів до приватного API.

Для спрощення реалізації цих стратегій Google розробила бібліотеку Workbox, яка надає готові інструменти для організації прекешування (precache) і кешування під час виконання (runtime cache) [4].

Користувацький досвід (UX) є одним із головних показників якості PWA. За даними досліджень Google і Mozilla, швидкість відгуку сторінки безпосередньо впливає на показники взаємодії користувачів: якщо час завантаження сторінки перевищує три секунди, понад 50 % відвідувачів залишають сайт [5].

Кешування у PWA вирішує цю проблему, скорочуючи час завантаження повторних візитів до мінімуму. Навіть при слабкому інтернет-з'єднанні користувач має змогу переглядати вже відвідані сторінки блогу або раніше завантажені пости. Крім того, Service Worker забезпечує можливість відкладеного оновлення даних і фонової синхронізації, що покращує сприйняття швидкодії.

З точки зору бізнесу, кешування забезпечує зменшення навантаження на сервери, економію трафіку та підвищення показників залучення користувачів (engagement rate). Для блогових платформ це особливо важливо, адже швидке завантаження контенту підвищує ймовірність повторних відвідувань і покращує SEO-рейтинги. За даними Datadog [5], впровадження PWA із коректно налаштованим кешем дозволяє скоротити використання серверних ресурсів на 20–40 %, а також збільшити конверсію на 10–15 % завдяки зниженню часу очікування користувачем.

Порівняння різних стратегій кешування показує, що ефективність залежить від типу даних і архітектури блогу. Наприклад, для статичних сторінок доцільно застосовувати Cache First, тоді як для API-запитів – Network First або Stale-While-Revalidate, що забезпечує баланс між актуальністю й швидкодією.

У випадку складних рішень можна поєднувати підходи PWA із зовнішніми системами кешування, такими як Redis чи Apollo Cache. Redis забезпечує серверне кешування запитів до бази даних, тоді як Apollo Cache – клієнтське кешування у застосунках, що використовують GraphQL. У порівнянні з ними кешування на рівні PWA є більш доступним і не потребує складної інфраструктури.

Під час розробки PWA важливо правильно визначити, які ресурси мають бути включені до прекешу, а які – завантажуватися динамічно. Workbox дозволяє автоматично створювати service worker із зазначенням стратегій кешування для кожного типу файлів. Крім того, важливим етапом є тестування кешу за допомогою інструментів Chrome DevTools, Lighthouse або Workbox Analyzer, які допомагають оцінити ефективність кешу, обсяг пам’яті та коректність оновлення ресурсів [3].

Під час створення блогу на основі PWA доцільно реалізувати:

– прекешування головних сторінок та стилів;
– runtime-кешування зображень та медіа;
– застосування політики оновлення контенту через Stale-While-Revalidate;
– додаткову оптимізацію мережевих запитів для API.

У межах експериментальної частини дослідження було розроблено демонстраційний блог із реалізацією кількох стратегій кешування для порівняльного аналізу. Результати тестування показали, що при використанні Stale-While-Revalidate середній час завантаження сторінки скоротився на 38 %, а кількість мережевих запитів – на 45 % у порівнянні з відсутністю кешу.

Отримані результати підтверджують доцільність застосування комбінованих стратегій кешування для забезпечення балансу між продуктивністю, актуальністю та стабільністю роботи PWA-блогів.

У результаті проведеного дослідження було встановлено, що технологія Progressive Web Apps (PWA) забезпечує новий рівень ефективності та доступності вебзастосунків, поєднуючи переваги вебсайтів і нативних програм. Ключовим чинником підвищення продуктивності PWA є раціональне застосування стратегій кешування, які визначають поведінку системи при обробці запитів, доступності даних та оновленні контенту.

Аналіз технічних аспектів показав, що Service Worker та Cache Storage API є основними механізмами, які дозволяють реалізувати офлайн-доступ і швидке завантаження сторінок блогу. Вибір стратегії кешування має базуватись на типі контенту та частоті його оновлення:

– для статичних ресурсів (зображення, стилі, скрипти) найбільш ефективною є стратегія Cache First;

– для динамічного контенту – Network First або Stale-While-Revalidate, що забезпечує баланс між актуальністю та швидкодією.

Дослідження користувацького досвіду довело, що застосування кешування скорочує час завантаження сторінок у середньому на 30–40 %, підвищує стабільність роботи при нестабільному з’єднанні та позитивно впливає на показники утримання

аудиторії. Це підтверджує високу роль кешування у підвищенні якості взаємодії користувача з вебзастосунком.

З економічної точки зору, використання кешування дозволяє зменшити навантаження на сервери, знизити споживання трафіку й витрати на обробку запитів, а також підвищити ефективність просування блогу через покращення показників SEO та Core Web Vitals.

Порівняльний аналіз продемонстрував, що хоча зовнішні інструменти, такі як Redis або Apollo Cache, ефективні для серверного чи клієнтського кешування, інтегровані рішення PWA на базі Workbox є простішими у впровадженні та повністю задовольняють потреби блогових платформ малого й середнього масштабу.

Розроблений у межах дослідження демонстраційний блог із реалізованими стратегіями кешування підтвердив ефективність підходу Stale-While-Revalidate: сторінки завантажувалися швидше, а мережеве навантаження скоротилося майже вдвічі. Це свідчить про доцільність комбінування стратегій залежно від типу ресурсів та сценаріїв використання.

Список використаних джерел

1. Caching and Performance in Progressive Web Apps. URL: https://web.dev/caching/ (дата звернення: 05.11.2025).

2. Progressive Web Apps. URL: https://developer.chrome.com/docs/workbox/ (дата звернення: 05.11.2025).

3. Malavolta I., Ruberto S., Soru T., Terragni V. Evaluating the impact of caching on the energy consumption and performance of Progressive Web Apps // *Proceedings of the 2020 IEEE International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, 2020. P. 24–35.

4. Workbox: Caching Strategies Guide. URL: https://developer.chrome.com/docs/workbox/caching-strategies-overview/ (дата звернення: 05.11.2025).

5. Vaadin. PWA Caching Strategies. URL: https://vaadin.com/docs/latest/pwa/caching-strategies (дата звернення: 05.11.2025).

ПІДГОТОВКА МАЙБУТНІХ ФАХІВЦІВ З ЦИФРОВИХ ТЕХНОЛОГІЙ: КОМПЕТЕНТНІСНИЙ ПІДХІД

Бойко Володимир Володимирович
здобувач третього рівня вищої освіти, спеціальність Освітні, педагогічні науки
Тернопільський національний педагогічний університет імені Володимира Гнатюка
vovaboyko3007@gmail.com

Сучасне суспільство вимагає від системи вищої освіти підготовки фахівців, які володіють фундаментальними знаннями, вміннями та навичками, здатні критично мислити, творчо підходити до вирішення поставлених завдань, бути конкурентоспроможними на ринку праці. Ринкові потреби вимагають орієнтацію не лише на формування знань, але й практичних, соціально-комунікативних та цифрових компетентностей випускників закладів вищої освіти. Національна рамка кваліфікацій та Стандарти вищої освіти України визначають ті компетентності, якими має оволодіти здобувач після завершення освітньої програми та акцентують увагу на компетентнісному підході у формуванні кваліфікаційних вимог і досягненні