by merging the physical and digital worlds, offering an engaging way to connect with the past [2].

Additionally, the app includes geolocation services, which help users discover nearby cultural sites, providing not only navigation but also rich historical context for each location. Users can explore interactive maps that highlight key points of interest, suggest routes for self-guided tours, and even provide audio or visual explanations as they move through the area.

A significant aspect of the project is its inclusivity. The app is designed to be accessible to a diverse audience, including individuals with disabilities. For instance, it will offer audio descriptions for visually impaired users and text-based guides for those with hearing impairments. This ensures that cultural heritage becomes more universally available, breaking down barriers to access [3].

The project also emphasizes personalization. By inputting their interests – such as architecture, local legends, or art – users can receive tailored recommendations and curated tour experiences. This not only enhances engagement but also makes the exploration process more meaningful and enjoyable.

In essence, the app is a tool for modernizing cultural tourism. It connects people to history through technology, creating a sustainable, interactive, and educational platform that aligns with the needs and expectations of today's travelers. The ultimate goal is to foster a deeper appreciation of cultural heritage while supporting sustainable tourism practices.

The digitalization of interactive cultural tours is a promising field that combines technology with cultural preservation and education. While challenges remain, ongoing advancements in digital tools and methodologies continue to enhance the scope and impact of these initiatives.

References

1. Balyk N., Shmyger G., Vasylenko Y., Skaskiv A. Study of augmented and virtual reality technology in the educational digital environment of the pedagogical university Innovative Educational Technologies, Tools and Methods for E-learning Scientific Editor Eugenia Smyrnova-Trybulska «E-learning». Katowice-Cieszyn, 2020. № 12. P. 305–313.

2. Lin C.-Y., Li C., Mahmood S., Guo F., Qian Z. Integrating Culture and Tourism: A Resilient Scale for Digital Transformation Innovation. *Journal of the Knowledge Economy*, 2024. № 15.

3. Rodrigues Gonçalves A., Peres Dorsch L. L., Figueiredo M. Digital Tourism: An Alternative View on Cultural Intangible Heritage and Sustainability in Tavira, *Sustainability*. Portugal, 2022. № 14(5).

ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕХАНІЗМІВ КЕШУВАННЯ В ПРОГРЕСИВНИХ ВЕБЗАСТОСУНКАХ

Базиволяк Максим Іванович
здобувач другого (магістерського) рівня вищої освіти зі 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,
shmyger@fizmat.tnpu.edu.ua

У сучасній веброзробці продуктивність і швидкість завантаження застосунків відіграють ключову роль у забезпеченні якісного користувацького

досвіду. Користувачі очікують миттєвого завантаження сторінок, безперебійного доступу до контенту навіть за нестабільного інтернет-з'єднання, а також мінімального споживання ресурсів пристрою. Прогресивні вебзастосунки (PWA) поєднують переваги традиційних вебсайтів і мобільних додатків, забезпечуючи швидке завантаження, офлайн-доступ і високу інтерактивність. Одним із найважливіших аспектів їхньої ефективної роботи є механізми кешування, які дозволяють зменшити затримки при завантаженні контенту, знизити навантаження на сервери та покращити загальну продуктивність.

У PWA використовуються різні підходи до кешування, зокрема Cache API, Service Workers, Workbox, а також серверні рішення, такі як Redis і Varnish. Кожен із цих методів має свої переваги, недоліки та сферу застосування. Однак вибір оптимальної стратегії значною мірою залежить від специфіки застосунку, моделі використання та вимог до швидкодії. Незважаючи на те, що кожен з цих механізмів має свої переваги та обмеження, їх комплексне порівняння у контексті прогресивних вебзастосунків залишається недостатньо дослідженим.

Дане дослідження є актуальним, оскільки воно дозволить розробникам та IT-фахівцям краще зрозуміти сильні та слабкі сторони різних методів кешування, що сприятиме ефективнішому вибору технологій для оптимізації продуктивності PWA. Результати аналізу можуть бути корисними для компаній, що розробляють високонавантажені вебзастосунки, а також для спільноти розробників, яка прагне вдосконалювати сучасні підходи до вебоптимізації.

Кешування у вебзастосунках – це процес збереження часто використовуваних даних для їхнього повторного використання без необхідності повторного завантаження з сервера [1]. Це дозволяє зменшити затримки при завантаженні сторінок, оптимізувати використання мережевих ресурсів та покращити загальну продуктивність вебзастосунків.

Принципи роботи кешування:

Клієнтське кешування – збереження ресурсів у браузері або на пристрої користувача: Cache API – дозволяє вебзастосунку зберігати та отримувати HTTP-запити і відповіді; Service Workers – проксі-сервери, які керують кешуванням і можуть обробляти запити навіть у режимі офлайн; Workbox – бібліотека, що спрощує реалізацію стратегій кешування у PWA.

Серверне кешування – збереження оброблених даних на стороні сервера: Redis – високошвидкісне сховище ключ-значення для збереження динамічного контенту [2]; Varnish – зворотний проксі-сервер, що кешує HTML-сторінки для швидкої доставки [3].

Кешування на рівні мережі – використання CDN (Content Delivery Network) для розповсюдження кешованого контенту між серверами по всьому світу.

Вплив кешування на швидкодію вебресурсів: зменшення часу завантаження – кешовані дані можуть завантажуватися безпосередньо з локального сховища, а не з сервера; зниження навантаження на сервер – зменшується кількість запитів до серверної частини вебзастосунку; підвищення доступності – вебзастосунки можуть працювати навіть без підключення до

мережі; оптимізація використання мережевого трафіку – особливо важливо для мобільних користувачів і повільних з'єднань.

Таблиця 1

Порівняння клієнтських механізмів кешування

Параметр	Cache API	Service Workers	Workbox
Може працювати автономно	✅ Так	❌ Ні (потребує Service Worker)	❌ Ні (потребує Service Worker)
Перехоплення запитів	❌ Ні	✅ Так	✅ Так
Стратегії кешування	⚠ Обмежені	✅ Різні варіанти (cache-first, network-first тощо)	✅ Автоматизовані
Автоматичне оновлення кешу	❌ Ні	⚠ Частково (потрібно реалізовувати вручну)	✅ Так
Офлайн-робота	❌ Обмежена	✅ Так	✅ Так
Простота використання	⚠ Середня	❌ Складна (потрібен ручний код)	✅ Дуже проста
Гнучкість	✅ Висока	✅ Висока	⚠ Обмежена (залежить від Workbox API)

Cache API – найкраще підходить для базового кешування ресурсів без складних стратегій. Service Workers – основний механізм для реалізації гнучких стратегій кешування та офлайн-роботи. Workbox – найкраще рішення для тих, хто хоче швидко налаштувати кешування без складної розробки. Залежно від потреб застосунку, можна комбінувати ці механізми для досягнення максимальної продуктивності та зменшення затримок у PWA.

На основі проведеного дослідження можна сформулювати такі практичні рекомендації щодо ефективного використання механізмів кешування у прогресивних вебзастосунках:

1. Використовувати Service Workers для офлайн-роботи та зменшення навантаження на сервер: реалізація Cache API для збереження основних ресурсів (HTML, CSS, JS, зображень) і забезпечення швидкого доступу до них; використання стратегії кешування, такі як cache-first (для статичних ресурсів) та stale-while-revalidate (для динамічного контенту); налаштування Workbox для автоматизованого кешування та оновлення ресурсів.

2. Оптимізувати серверне кешування для зменшення затримок і навантаження: використовувати Redis для кешування часто запитуваних API-даних, що зменшить час відповіді сервера; налаштувати Varnish Cache як зворотний проксі-сервер для кешування HTML-сторінок і прискорення доставки контенту; використовувати CDN (Cloudflare, Akamai, Fastly) для глобального кешування статичних ресурсів.

3. Обирати відповідну стратегію кешування залежно від типу контенту: Cache-first – для статичних ресурсів (CSS, JavaScript, зображення); Network-first – для даних, що часто оновлюються (новини, повідомлення); Stale-while-revalidate – для контенту, який можна оновлювати у фоновому режимі (статті, каталоги

товарів); No-cache або network-only – для даних, що критично важливо завжди отримувати свіжими (особисті повідомлення, платіжна інформація).

4. Використовуйте контроль кешування на рівні заголовків HTTP: визначення часу життя кешу через Cache-Control (max-age, must-revalidate, immutable); налаштовуйте ETag для контролю оновлень ресурсів; використання Expires для визначення часу дії кешованих ресурсів.

5. Регулярно перевіряти та оновлюйте кешовані дані: реалізація механізму автоматичного оновлення кешу при виході нових версій застосунку; використання background sync для оновлення кешу при відновленні з'єднання; додавання механізму видалення застарілих кешованих даних для зменшення використання пам'яті пристрою.

6. Тестувати ефективність кешування: використання Lighthouse або PageSpeed Insights для аналізу ефективності кешування; тестування продуктивності кешованих ресурсів у Chrome DevTools (Application → Cache Storage Service Workers); аналіз мережевого трафіка та частоти оновлення кешованих ресурсів.

Застосування цих рекомендацій допоможе значно підвищити швидкодію прогресивних вебзастосунків, покращити користувацький досвід і зменшити навантаження на серверну інфраструктуру.

У результаті дослідження механізмів кешування у прогресивних вебзастосунках (PWA) було проведено порівняльний аналіз різних підходів до кешування та оцінено їхню ефективність у підвищенні продуктивності вебресурсів.

Кешування є ключовим фактором у забезпеченні швидкодії PWA. Воно дозволяє значно зменшити час завантаження, знизити навантаження на сервери та покращити користувацький досвід, особливо при нестабільному інтернет-з'єднанні або в офлайн-режимі. Клієнтські механізми кешування (Cache API, Service Workers, Workbox) забезпечують ефективне управління ресурсами на боці браузера. Service Workers дозволяють реалізовувати різні стратегії кешування, а Workbox спрощує їх інтеграцію, автоматизуючи оновлення кешу та керування мережею. Серверні рішення, такі як Redis і Varnish, ефективно зменшують навантаження на сервер. Redis використовується для кешування динамічних даних у пам'яті, тоді як Varnish покращує доставку HTML-сторінок, що прискорює відображення контенту. Контроль кешування через HTTP-заголовки є важливим аспектом ефективного управління ресурсами. Регулярне тестування та моніторинг кешування допомагають підтримувати високу продуктивність PWA. Інструменти, такі як Lighthouse, Chrome DevTools та PageSpeed Insights, дозволяють аналізувати ефективність кешу та виявляти можливі проблеми.

Правильне поєднання клієнтського та серверного кешування дозволяє досягти оптимального балансу між швидкістю завантаження, актуальністю контенту та навантаженням на інфраструктуру. Використання сучасних механізмів кешування є необхідною складовою розробки продуктивних і надійних прогресивних вебзастосунків.

Список використаних джерел

1. PWA Caching. URL: https://web.dev/learn/pwa/caching (available at: 30.03.2025).
2. What is Redis? URL: https://www.ibm.com/think/topics/redis (available at: 30.03.2025).
3. Introduction to Varnish. URL: https://varnish-cache.org/intro/ (available at: 30.03.2025).

ОНЛАЙН-ОСВІТА: АНАЛІЗ СУЧАСНИХ ПЛАТФОРМ ДЛЯ ДИСТАНЦІЙНОГО НАВЧАННЯ

Банцур Ірина Олександрівна
здобувач першого (бакалаврського) рівня вищої освіти спеціальності Середня освіта (Інформатика),
ернопільський національний педагогічний університет імені Володимира Гнатюка,
bantsur_io@fizmat.tnpu.edu.ua
Барна Ольга Василівна
кандидат педагогічних наук, доцент кафедри інформатики та методики її навчання,
Тернопільський національний педагогічний університет імені Володимира Гнатюка,
barna@tnpu.edu.ua

Одним із трендів сучасної освіти є дистанційна освіта, яка відповідає на потреби сучасного суспільства в мобільності, гнучкості та доступності знань.

Завдяки дистанційному навчанню непомітні географічні та соціальні бар'єри, користувачі мають доступ до провідних університетів та організацій. Платформи для дистанційного навчання, серед яких Moodle, Google Classroom, Zoom, дозволяють організувати свій час найзручніше, створюючи нове кооперативне навчання та практичне закріплення знань [4, с. 100–101].

Незважаючи на переваги, онлайн-освіта вимагає високого рівня самодисципліни, хорошого технічного забезпечення та новітніх педагогічних підходів. Важливими залишаються взаємодія між студентами та викладачами, удосконалення сучасних методик навчання та контроль якості освітнього процесу [1]. Тому виникає необхідність аналізу переваг та недоліків платформ для підтримки онлайн освіти, що і є метою даного дослідження.

Сучасні цифрові технології мають потужний вплив на теперішнє життя та значною мірою впливають на наше навчання. Результатом такого впливу є розвиток онлайн-освіти, що визначається потребою в адаптації навчального процесу до викликів (пандемія, військові дії, соціальна нерівність, тощо) [3, с. 44].

Онлайн-освіта є невід'ємною частиною сьогоднішнього освітнього процесу, що дозволяє усім доєднатися до знань і забезпечити індивідуальне навчання. Однією з важливих тенденцій є широке впровадження хмарних технологій в освітній процес. Наприклад, у Данії з 2013 року цифрові навчальні платформи (DLP) є обов'язковими. Ця навчальна платформа містить різні елементи для створення навчальних курсів. Вчителі можуть ними ділитися, змінювати, також такі курси є доступними для наукових педагогічних досліджень [3, с. 44]. Латвійські дослідники провели опитування серед вчителів щодо їхнього ставлення до використання навчальних платформ, було опитано 705 вчителів. За