DOI: https://doi.org/10.15276/ict.03.2026.03
УДК 004.415.2:004.738.5

Контрактний фреймворк адаптивних політик для прогресивних вебзастосунків в умовах нестабільних мереж

Приліпа Артем Олегович[1)]
ORCID: https://orcid.org/0009-0005-6633-8308; Artem.Prylipa@cs.khpi.edu.ua
Філатова Ганна Євгенівна[1)]
ORCID: https://orcid.org/0000-0003-1982-2322; Hanna.Filatova@khpi.edu.ua. Scopus Author ID: 56448583600
[1)] Національний технічний університет «Харківський політехнічний інститут», вул. Кирпичова, буд. 2, Харків, 61002, Україна

АНОТАЦІЯ

У роботі розглянуто перехід від прикладного методу адаптивних політик до повторно використовуваного фреймворку для прогресивних вебзастосунків, у яких нестабільна мережа впливає на читання, обсяг контенту та завершення записів. Попередній метод було реалізовано для одного застосунку, тому він не давав змоги перевірити повторне використання ядра, узгоджену активацію політики у віконному контексті та Service Worker і коректність відкладених операцій за відмов. Запропоновано контрактний програмний фреймворк, що поєднує спільне ядро зі специфікацією застосунку, яка містить реєстр операцій, політику, адаптери та можливості серверної частини. Експериментальне оцінювання охопило порівняння із зафіксованим еталоном поведінки, контрольоване внесення відмов і перевірку повторного використання ядра у двох вебзастосунках. Отримано повний збіг рішень та однакові браузерні результати для всіх пар основного експерименту, а часові витрати залишилися в установлених протоколом межах. Усі 54 виконання матриці відмов завершилися очікувано, а повторна доставка за атомарного серверного контракту не спричинила більш ніж однієї зміни стану на ключ ідемпотентності. Обидва вебзастосунки використали те саме ядро і пройшли спільні перевірки. Отримані результати підтверджують повторне використання фреймворку, однак перенесення висновку на інші серверні реалізації потребує окремої перевірки.

Ключові слова: комп'ютерна система; PWA; Service Worker; адаптивні політики; контрактний фреймворк; відкладені операції; нестабільні мережі

Для цитування: **Приліпа А. О., Філатова Г. Є. "Контрактний фреймворк адаптивних політик для прогресивних вебзастосунків в умовах нестабільних мереж". *Інформатика. Культура. Техніка.* 2026; Том 3 № 1 (3): 37–47. DOI: https://doi.org/10.15276/ict.03.2026.03**

Актуальність. Для прогресивних вебзастосунків (PWA) нестабільна мережа впливає не лише на затримку, а й на вибір стратегії читання, обсяг контенту та можливість завершити запис. Кешування покращує швидкодію, проте його ефект залежить від конфігурації, пристрою й сценарію та може збільшувати енергоспоживання [1], [2], [3]. Service Worker забезпечує програмоване перехоплення запитів, а HTTP визначає правила повторного використання кешованих відповідей [4], [5]. Тому адаптацію доцільно оцінювати за результатом операції, а не лише за транспортними метриками.

Для запису без мережі надійна черга має зберігати операцію до відновлення з'єднання та запобігати повторній зміні прикладного стану [6], [7]. Самоадаптивні системи, ActivFORMS і ASPLe пропонують контекстно-залежні рішення та повторне використання [8], [9], [10]. Водночас контрактний підхід і приховування змінних рішень вимагають явних передумов, умов коректності та оцінювання фактичних витрат перенесення [11], [12], [13]. Для PWA ця межа охоплює політику, два контексти виконання, локальне сховище й серверну частину.

Роботи про локально орієнтовані системи показують, що синхронізація після роботи без мережі потребує спроєктованого стану [14], [15]. Workbox зосереджується на формуванні черги та повторному надсиланні HTTP-запитів, PouchDB орієнтований на використання локальної документної бази даних і реплікації, а SWAPP використовує Service Worker як програмовану платформу [16], [17], [18]. Ці рішення не визначають узгодженої версіонованої політики для віконного контексту і Service Worker. ActivFORMS формалізує загальний цикл самоадаптації, а ASPLe забезпечує повторне використання на рівні методології. На відміну від зазначених підходів, запропонований фреймворк поєднує типізований реєстр операцій, узгоджену версіоновану політику для обох контекстів виконання, підтвердження постановки операції в чергу та серверний контракт ідемпотентності.

© Приліпа А., Філатова Г., 2026

Стаття опублікована у відкритому доступі відповідно до умов ліцензії CC BY (https://creativecommons.org/licenses/by/4.0/deed.uk)

Попередня робота авторів запропонувала браузерний засіб TinyML-профілювання, який у цій статті розглядається лише як замінна реалізація межі класифікатора [19]. У наступній роботі запропоновано прикладний метод адаптивних політик для одного вебзастосунку, що поєднував стратегії читання й запису, деградацію контенту та відкладення операцій [20].

Результати, отримані раніше для цього методу, у цій роботі не розглядаються як нові.

Невирішеним залишався перехід від одного прототипу до повторно використовуваного ядра. Не було перевірено узгоджену активацію політики у віконному контексті та Service Worker, відсутність хибного підтвердження відкладеного запису й незмінність ядра для іншої предметної області. Це зумовило потребу у контрактній моделі та окремій перевірці її поведінки за відмов.

Метою дослідження є розробка й оцінка фреймворку для адаптації прогресивних вебзастосунків до нестабільних мереж. У запропонованому фреймворку прикладні правила операцій відокремлено від спільних механізмів виконання шляхом розмежування реєстру операцій, політик, адаптерів і спільного ядра. Наукова новизна полягає у введенні контрактної моделі для PWA вебзастосунків, яка відокремлює незмінне ядро від специфікації застосунку, узгоджує версії політики у віконному контексті та Service Worker і пов'язує атомарну фіксацію відкладеного запису із серверною ідемпотентністю. Експериментальний внесок охоплює перевірку коректності в умовах відмов та оцінювання повторного використання ядра.

Для досягнення мети поставлено чотири задачі:

(1) формалізувати межу між ядром і специфікацією застосунку;

(2) реалізувати контракти узгодженої політики, сховища та повторної доставки;

(3) визначити метрики й протокол експериментів E1-E3;

(4) оцінити збереження поведінки, накладні витрати, коректність за відмов і повторне використання ядра.

У роботі розглянуто три реалізації двох прогресивних вебзастосунків. Попередню реалізацію першого вебзастосунку позначено A0, а реалізацію цього самого вебзастосунку на запропонованому фреймворку позначено A1. Вебзастосунок у реалізаціях A0 та A1 призначений для реєстрації учасників подій і охоплює читання відомостей про події й учасників, одиничні та пакетні операції реєстрації, адаптацію контенту та відкладення записів без мережі. Другу реалізацію на тому самому ядрі фреймворку визначено як вебзастосунок A2. Вебзастосунок A2 призначений для обліку запасів і охоплює читання переліку й деталей позицій, версіоновані зміни кількості та стану, пакетну синхронізацію відкладених операцій. Отже, A0 і A1 є попередньою та фреймворковою реалізаціями одного вебзастосунку, які порівнюються в експерименті відповідності, а A1 і A2 є двома вебзастосунками на фреймворку, які використовуються для оцінювання повторного використання ядра.

Для перевірки запропонованого узагальнення сформульовано чотири гіпотези:

- гіпотеза про збереження поведінки (H1): реалізація A1 відтворює нормалізовані рішення та результати попередньої реалізації A0 у спільній області входів;
- гіпотеза про прийнятні накладні витрати (H2): контрактні механізми не збільшують часові витрати понад визначені протоколом межі;
- гіпотеза про коректність за відмов (H3): фреймворк не порушує умов коректності активації політики, надійної фіксації, ізоляції та повторної доставки у визначеній матриці відмов;
- гіпотеза про повторне використання ядра (H4): реалізації A1 і A2 використовують те саме ядро без залежних від застосунків змін і проходять спільні перевірки.

Для перевірки сформульованих гіпотез визначено три експерименти. В експерименті E1 попередню реалізацію A0 зіставлено з реалізацією на фреймворку A1 за нормалізованою поведінкою та часовими межами, що забезпечує перевірку H1 і H2. В експерименті E2 виконано контрольоване внесення відмов до контрактів політики, сховища, планувальника та серверної ідемпотентності для перевірки H3. В експерименті E3 вебзастосунки A1 і A2 підключено до одного зафіксованого ядра та перевірено відсутність залежних від застосунків

змін ядра і проходження спільних сценаріїв, що забезпечує перевірку H4. Для кожного експерименту наперед визначено входи, правила нормалізації, метрики та критерії прийняття.

Критеріями прийняття є повна відповідність у спільній області, дотримання часових меж, відсутність критичних порушень і відсутність змін ядра. Область висновків обмежена двома вебзастосунками, визначеними сценаріями та атомарним контрактом серверної частини.

Запропонована контрактна модель подає вебзастосунок як поєднання незмінного ядра фреймворку та специфікації застосунку. Модель формалізує активацію політики, збереження поведінкової відповідності, безпечне відкладення запису, серверну ідемпотентність і повторне використання ядра. Це дає змогу пов'язати архітектурні вимоги з вимірюваними критеріями експериментів.

Ключовим елементом запропонованої моделі є структурована специфікація застосунку, яка відокремлює правила поведінки від механізмів виконання:

$$D_i = \left\langle \mathrm{id}_i, R_i, P_{(i,v)}, A_i, B_i \right\rangle,$$

де id_i – визначає ідентичність та область дії; R_i – типізований реєстр операцій; $P_{(i,v)}$ – політику версії v; A_i – адаптери; B_i – можливості серверної частини.

Усі відмінності конкретного застосунку зосереджуються у специфікації, яка стає єдиною точкою його інтеграції з ядром.

Незмінні механізми подаються як ядро:

$$F = \langle V, C, K_W, K_{SW}, S, T \rangle,\ I_i = F \otimes D_i,$$

де V – є модулем перевірки; C – модулем перетворення політики на виконавчий знімок; K_W і K_{SW} – відповідають ядрам віконного контексту та Service Worker; S – відповідає підсистемі сховища; T – телеметрії. Оператор $\otimes$ – задає поєднання ядра зі специфікацією застосунку через визначені контракти, унаслідок чого результатом I_i є налаштований PWA, а не нова версія ядра. Правила поведінки конкретного застосунку до F не вносяться, а виконання цієї вимоги оцінюється за змінами зафіксованого ядра в експерименті E3.

Передумовою активації є повна структурна й семантична коректність специфікації застосунку:

$$valid(D_i) \in \{0,1\},\ \ valid(D_i) \Rightarrow active(P_{i,v}) = 0.$$

Функція $active(P_{i,v})$ позначає доступність політики для нових операцій. Часткове прийняття помилкової конфігурації забороняється, а в разі її відхилення зберігається останній коректний знімок. Експеримент E2 із внесення відмов перевіряє цю властивість на помилкових специфікаціях.

Після перевірки модуль перетворення створює пару незмінних знімків:

$$C(D_i) = \left(\hat{P}^W_{i,v}, \hat{P}^{SW}_{i,v} \right),\ canon\left(\hat{P}^W_{i,v} \right) = canon\left(\hat{P}^{SW}_{i,v} \right).$$

Верхні індекси позначають віконний контекст і Service Worker, а однакове нормалізоване подання обох знімків є передумовою активації версії політики. Самого збігу подань недостатньо, тому версіонований протокол додатково передбачає підтвердження версії та закріплення знімка за операцією. Це запобігає прийняттю рішень на змішаних версіях без вимоги спільної пам'яті між контекстами, якщо вхідні дані нормалізуються однаково. Коректність такого механізму під час оновлень політики й протокольних відмов перевіряється в експерименті E2.

Для операції j фіксується контекст

$z_j = \left(s_j, r_j, u_j, k_j, v \right)$, після чого дія визначається як $a_j = \Delta F(D_i, z_j)$,

де s_j є – стабілізованим мережевим станом; r_j – визначенням операції; u_j – контекстом інтерфейсу; k_j – станом кешу, черги й надійності; v – версією політики.

Контекст і рішення залишаються незмінними до завершення операції. В експерименті E1 під час перевірки поведінкової відповідності та накладних витрат отримані дії порівнюються з еталоном поведінки, а в експерименті E2 із внесенням відмов перевіряється відсутність змішаних версій.

За однакових входів відповідність попередній реалізації оцінюється на множині спільних контекстів Z:

$$DA = \frac{1}{|Z|}\sum_{z\in Z}\mathbf{1}\left[\Delta_F(D,z) \equiv \pi_{\text{legacy}}(z)\right].$$

Множина Z, визначена до початку експерименту, містить 1440 контекстів. Вони утворюють декартів добуток множин станів мережі, зареєстрованих операцій і станів кешу. Для цих контекстів попередня реалізація A0 слугує еталоном поведінки. Позначення $\equiv$ означає збіг після застосування правил нормалізації. Під час нормалізації вилучаються нестабільні ідентифікатори, впорядковуються ключі й уніфікуються статуси завершення. Після цього порівнюються нормалізовані дії читання, контенту та запису. Індикатор дорівнює одиниці лише за повного збігу. Це основна метрика експерименту відповідності.

Безпечне підтвердження відкладеного запису вимагає

$$\text{returnedQueued}(o_j) \Rightarrow \text{commit}_{IDB}(o_j) = \text{success},$$

де o_j – позначає запис операції; commit_{IDB} – результат атомарної транзакції IndexedDB.

Для безпечного повтору серверна частина атомарно зберігає

$$M[k_j] = \left(canon(m_j), y_j\right) \text{ і } N_{\text{effect}}(k_j) \le 1,$$

де k_j – є ключем ідемпотентності; m_j – тілом операції запису; y_j – логічним результатом; N_{effect} – кількістю прикладних змін стану.

За однакових ключа й тіла серверна частина повертає збережений результат, тоді як інше тіло спричиняє конфлікт. Експеримент із внесенням відмов перевіряє дотримання цієї умови під час повторної доставки, яка сама може відбуватися багаторазово [6], [7].

Незмінність ядра для двох інтеграцій оцінюється як:

$$CI = 1 - \frac{\left|\Delta\text{Core}_1 \cup \Delta\text{Core}_2\right|}{L^{\text{core}}},$$

де ΔCore_i – є множиною доданих, видалених і модифікованих логічних рядків у визначеній межі; L^{core} – її зафіксований обсяг.

Логічним вважається непорожній рядок, що не починається з //. Модифікований рядок рахують один раз. Переміщення незмінного вмісту дає нуль LOC, але окремо фіксується як зміна шляху. Значення $CI = 1$ означає відсутність змін ядра, залежних від вебзастосунків A1 або A2.

Запропонований фреймворк реалізує контрактну модель як спільне ядро та обмежені точки розширення. Ядро відповідає за перевірку й активацію політики, маршрутизацію операцій, сховище, повторну доставку та телеметрію, а правила конкретного вебзастосунку залишаються у специфікації застосунку. Архітектура розмежовує застосунок, контур керування у віконному контексті, спільні контракти та Service Worker. Специфікація застосунку містить реєстр, політику, адаптери й можливості серверної частини. Віконний компонент координує мережевий контекст і життєвий цикл політики, а Service Worker виконує

зареєстровані стратегії читання й запису. Таким чином, спостереження, рішення та виконання мають окремі межі [8], [9], [11], [12].

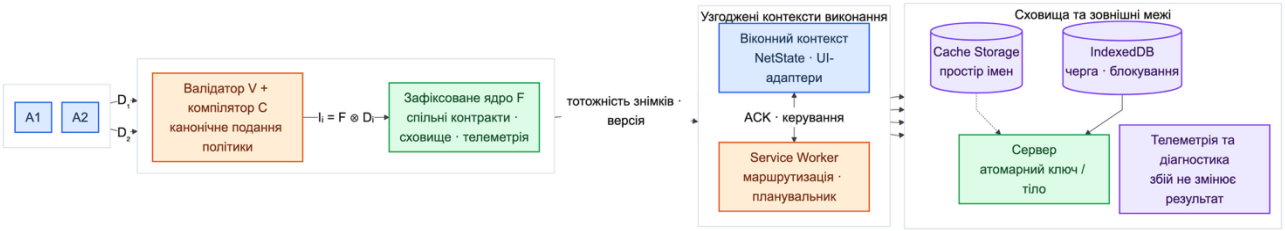

Рис. 1. **Використання фреймворку для вебзастосунків A1 і A2**
Джерело: **складено авторами**

Обидва вебзастосунки підключаються через відкриті точки входу й використовують те саме зафіксоване ядро. Відмінності зосереджуються в реєстрах, політиках і адаптерах, а не в спеціальних гілках ядра. Рис. 1 показує підключення специфікацій вебзастосунків до одного ядра F. Узгоджені знімки надходять до віконного контексту та Service Worker, а ізольовані сховища й серверна частина з атомарною реалізацією ідемпотентності обмежують наслідки відмов.

Реєстр пов'язує ідентифікатор операції з її видом, HTTP-методом, маршрутом, ключем кешу, схемою відповіді та доступністю без мережі. На цій основі формуються клієнтський виклик, зіставлення маршруту в Service Worker і допустимі стратегії. Невідомі маршрути відхиляються, а для невідомих операцій запису застосовується безпечний мережевий режим без використання черги.

Документ політики проходить структурну й семантичну перевірки, перетворення на виконавчий знімок та атомарну активацію. Помилковий документ відхиляється цілком, тому обидва контексти зберігають останній коректний або безпечний знімок. Версія активується лише після підтвердження збігу, а розпочата операція завершується з тим знімком, який отримала на початку. Мережеві спостереження також не змінюють зафіксованого стану операції. Стабілізатор застосовує гістерезис і строк придатності, після чого Service Worker відхиляє застарілі або дубльовані повідомлення. Для операції читання політика визначає мережево-кешову стратегію, строки свіжості, повтори та обмеження надійності. Service Worker виконує це рішення, зберігаючи семантику HTTP-кешування [4], [5].

Політика контенту окремо керує обсягом даних, ресурсами, відтворенням і доступними взаємодіями. Адаптер застосовує її до екрана, тоді як ядро не маніпулює прикладним інтерфейсом. Відкладення запису дозволено лише для зареєстрованої операції, доступної без мережі. Статус queued повертається після успішної атомарної транзакції IndexedDB, а помилка фіксації має статус *failed*.

Планувальник повторно перевіряє мережевий стан і отримує транзакційне блокування простору імен. Серверна частина атомарно зіставляє ключ ідемпотентності з тілом і результатом: повторний запит із тотожними ключем і тілом повертає збережений результат, а інше тіло спричиняє конфлікт. Це зумовлено потребою обмежити кількість прикладних змін стану, хоча мережевих доставок може бути кілька [6], [7]. Послідовність надійного відкладення операції, її повторної доставки та атомарного оброблення ключа ідемпотентності наведено на Рис. 2.

Кеші, таблиці IndexedDB, блокування й службові записи мають окремий простір імен. Очищення не видаляє сторонніх ресурсів, недоступність IndexedDB забороняє позитивне підтвердження відкладення, а збій телеметрії не змінює прикладного результату. Такі правила локалізують наслідки відмов і зберігають очікувану деградацію функціональності.

Відкритими залишаються специфікації застосунків, адаптери та інтеграційні входи, а внутрішні автомати, блокування й протоколи активації є закритими. Вебзастосунки A1 та A2 використовують однакові механізми ядра, а різні маршрути й правила контенту визначено в їхніх специфікаціях. Повторне використання оцінюється за відсутністю залежних від застосунків змін ядра та проходженням спільних контрактних сценаріїв.

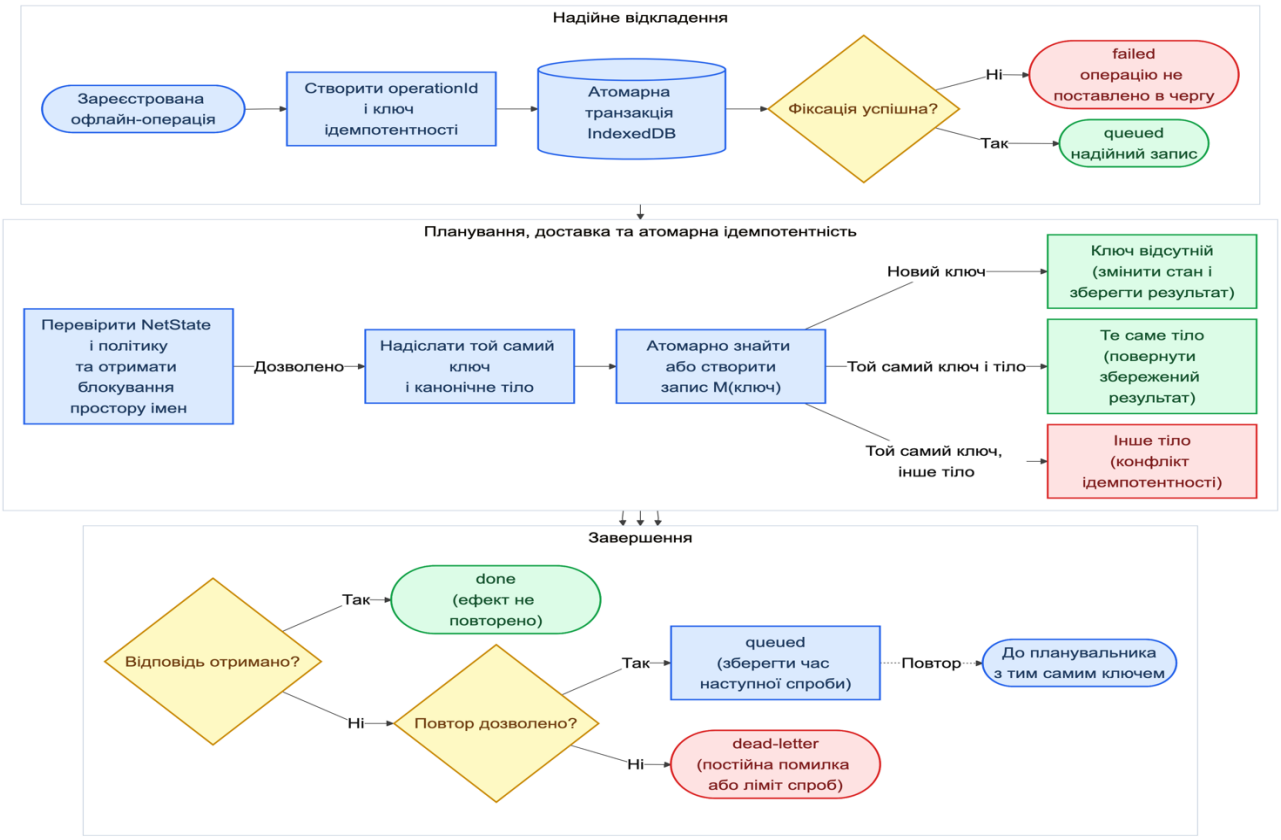

Рис. 2. **Протокол надійного відкладення та повторної доставки операції запису**
Джерело: **складено авторами**

В експерименті E1 виконано диференційне порівняння реалізацій A0 і A1 [21], [22]. Межі для перевірки гіпотези H2 визначено до підтверджувального етапу за результатами незалежних пілотних запусків. Остаточні пороги становили 10 мс для одноразового запуску, 15 мс для набору з п’яти сценаріїв і 0,05 мс для вибору політики. Перші два значення обрано так, щоб додаткова затримка не перевищувала тривалість одного кадру за частоти 60 Гц (16,7 мс), а для вибору політики встановлено суворіший поріг через багаторазове виконання цієї операції. Гіпотезу H2 вважали підтриманою, якщо верхня межа 95 % довірчого інтервалу різниці між A1 і A0 не перевищувала відповідного порогу [23].

Браузерний етап складався з восьми блоків по 30 пар, тобто 480 незалежних браузерних контекстів. У кожній парі A0 і A1 отримували однакове початкове значення генератора, дані та стан кешу, а порядок запуску рандомізували. Методика експерименту E2 передбачала контрольоване внесення відмов до контрактів політики, сховища, планувальника та серверної ідемпотентності [24]. Для кожної первинної точки матриці було заздалегідь визначено детермінований очікуваний результат, а додаткові розклади охоплювали конкурентне оброблення черги та повтори серверної доставки. У експерименті E3 реалізації A1 і A2 підключали до одного зафіксованого ядра. Ідентичність ядра перевіряли за сукупним SHA-256-хешем визначеної межі, переліком змін вихідних файлів і проходженням спільних контрактних та браузерних сценаріїв.

Результати. В експерименті E1 оцінено збереження поведінки й часові витрати, в E2 — коректність контрактів у матриці відмов, а в E3 — повторне використання зафіксованого ядра у двох PWA. Первинними одиницями аналізу є зіставні контексти та пари браузерних запусків для E1, точки матриці й конкурентні розклади для E2, а також зміни визначеної межі ядра та результати спільних перевірок для E3.

Для оцінювання збереження поведінки диференційне випробування охопило 2400 контекстів. Із них 1440 належали до наперед визначеної спільної області й дали повний збіг рішень, тобто $DA = 1$. Інші 960 контекстів відповідали визначеним до аналізу відмінностям. У 240 зіставлених парах браузерних запусків нормалізовані результати також збіглися, тому

$OCSR = 1$. Отримані результати узгоджуються з тим, що реєстр операцій і сформована політика A1 відтворюють ту саму межу вибору дій, що й A0. Отже, результати підтримують гіпотезу про збереження поведінки (H1).

Для оцінювання накладних витрат використано попарні часові різниці між A0 і A1. У 240 попарних браузерних запусках найбільші верхні межі 95 % ДІ становили 8,27 мс для запуску, 12,41 мс для виконання набору сценаріїв і 0,00017 мс для вибору політики. Отримані результати не перевищили відповідних меж $\Delta_{\max}$, що складали 10, 15 і 0,05 мс у жодному блоці. Отже, результати підтримують H2 щодо дотримання визначених протокольних меж, але не встановлюють універсальної користувацької прийнятності часових витрат.

Додаткові метрики не змінили основного висновку E1. Для нормалізованого обсягу результатів медіана різниці та 95 % ДІ дорівнювали нулю. Засіб *performance.memory* також не зафіксував різниці. Одиничне порівняння складань показало 256 кбайти для A0 і 94 кбайти для A1. У всіх 60 парах результати збіглися, а порушень коректності не зафіксовано. Це розширює підтвердження гіпотези H1 без зміни висновку щодо гіпотези H2.

Часові оцінки, на яких ґрунтується висновок щодо гіпотези H2, наведено на Рис. 3. Рисунок показує попарні оцінки різниці та їхні 95 % довірчі інтервали, нормовані відносно $\Delta_{\max}$. Усі верхні межі залишилися нижчими від критерію прийняття, що узгоджується з підтримкою гіпотези H2.

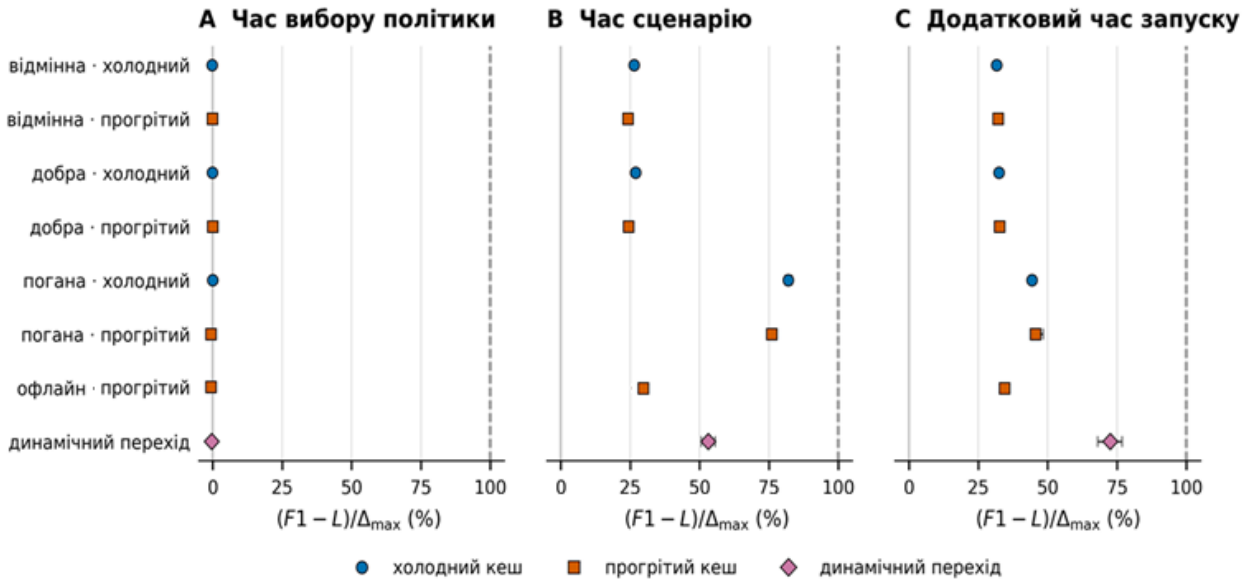

Рис. 3. **Нормовані часові витрати реалізації Check-in.**
Джерело: **складено авторами**

Для оцінювання коректності за відмов виконано 54 детерміновані перевірки в межах визначеної матриці. Усі вони завершилися очікувано, а критичних порушень умов коректності не виявлено. Первинна частина охопила 30 точок: усі 7 із 7 помилкових специфікацій, 7 із 7 протокольних відмов, 7 із 7 відмов сховища або черги та 9 із 9 випадків доставки або серверної частини завершилися відповідно до заздалегідь визначених результатів. Кожній точці відповідав окремий тест, тому повнота стосується визначеної матриці, а не всіх можливих відмов. Перевірка перед активацією не допустила помилкової специфікації, зіставлення знімків і підтвердження версії запобігли змішаним версіям політики, фіксація до повернення *queued* усунула хибне підтвердження, а ізоляція простору імен обмежила наслідки очищення. Серверний контракт забезпечив $N_{\text{effect}}(k) = 1$. Ці результати узгоджуються з окремими умовами прийняття для кожної небезпечної зміни стану та підтримують гіпотезу про коректність за відмов (H3) у межах визначеної матриці. Додатково конкурентне оброблення

черги та повтори серверної доставки перевірено у 24 розкладах: 12 початкових значень генератора поєднували з двома порядками виконання. Порушень не спостерігалося, що узгоджується з блокуванням простору імен та атомарним усуненням повторного виконання.

Для оцінювання повторного використання проаналізовано 10 224 логічні рядки визначеної межі ядра. Залежних від двох досліджених застосунків змін не виявлено: $\Delta\text{Core}_1 = \Delta\text{Core}_2 = 0$ і $CI = 1$. Ідентичність ядра перевіряли у трьох контрольних точках: до інтеграції A1, до інтеграції A2 та після неї. Сукупний SHA-256-хеш визначеної межі в усіх випадках залишився однаковим.

Обидві інтеграції пройшли спільні контрактні й браузерні перевірки без спеціальних точок розширення. Такий результат узгоджується з розміщенням різних правил поведінки в реєстрах, політиках і адаптерах за незмінного ядра.

У вебзастосунку A2 успішно перевірено сценарії читання з мережею, деградації контенту, запису без мережі, повторної доставки, конфлікту ідемпотентності та оновлення політики. Сукупність структурних і поведінкових результатів підтримує гіпотезу про повторне використання ядра (H4) для двох досліджених PWA. Порівняння наведено в таблиці, а перевірка на вебзастосунках з інших предметних областей має уточнити межі цієї властивості. Таблиця зіставляє структурну незмінність ядра та проходження спільних перевірок для обох застосунків. Однакові результати за різних реєстрів і політик узгоджуються з тим, що предметні відмінності залишилися за межами визначеного ядра.

Таблиця. **Повторне використання ядра в вебзастосунках A1 та A2**

Показник	A1	A2
Зміна ядра (логічні LOC)	0	0
Реєстр (логічні LOC)	184	153
Політика (логічні LOC)	1119	824
Адаптери (логічні LOC)	1051	937
Специфікація + адаптери (логічні LOC)	2354	1914
Публічні операції	10	7
Спеціальні точки розширення	0	0
Частка проходження контрактних тестів	100 %	100 %
Частка проходження браузерних тестів	100 %	100 %
Індекс незмінності ядра CI	100 %	100 %

Джерело: **складено авторами**

Фреймворк потенційно переноситься на браузерні PWA зі скінченним реєстром операцій і обмеженими відкладеними записами, які можна атомарно зберігати в IndexedDB, якщо предметні відмінності локалізуються в реєстрі, політиці й адаптерах без зміни ядраСерверна технологія може відрізнятися, але має атомарно пов’язувати ключ ідемпотентності, тіло операції та результат. Повтор з однаковими ключем і тілом повертає збережений результат, інше тіло спричиняє конфлікт, а конкурентні повтори спричиняють не більш як одну зміну стану.Синхронізація з кількома рівноправними вузлами або на основі безконфліктних реплікованих структур даних, великі чи потоково передавані тіла запитів, а також зовнішні операції, повторне виконання яких може спричиняти додаткові зміни стану, перебувають поза перевіреною областю. Для кожної нової інтеграції необхідно повторно перевірити незмінність ядра та проходження спільних перевірок, а часові показники оцінювати окремо.

Висновки. У роботі запропоновано контрактний фреймворк, що відокремлює правила конкретного вебзастосунку від спільних механізмів перевірки й активації політики, відкладення запису, ізоляції сховища та серверної ідемпотентності. Фреймворк оцінено за

збереженням поведінки, часовими витратами, коректністю за відмов і повторним використанням зафіксованого ядра. У зіставлених парах браузерних запусків A0 і A1 результати повністю збіглися, а верхні межі часових різниць залишилися нижчими від установлених протоколом критеріїв. Найпомітніші витрати спостерігалися під час узгодження політики між віконним контекстом і Service Worker. Ці результати підтримують гіпотезу про збереження поведінки (H1) у визначеній спільній області та гіпотезу про накладні витрати (H2) за встановлених меж. Ключовим результатом роботи є те, що всі детерміновані виконання матриці відмов завершилися очікувано, критичних порушень умов коректності не виявлено, а повторна доставка за атомарного серверного контракту не спричинила більш ніж однієї прикладної зміни стану на ключ ідемпотентності. Сукупність цих спостережень підтримує гіпотезу про коректність за відмов (H3) у перевірених сценаріях. Для вебзастосунків A1 та A2 сукупний SHA-256-хеш визначеної межі ядра залишився однаковим у трьох контрольних точках, а обидві інтеграції пройшли спільні перевірки. У межах проаналізованих логічних рядків залежних від цих застосунків змін ядра не виявлено. Отримані дані підтримують гіпотезу про повторне використання ядра (H4) для двох досліджених інтеграцій і не обґрунтовують універсальність поза ними. Отримана підтримка стосується вебзастосунків із зареєстрованими операціями читання, обмеженими ідемпотентними записами без мережі, явною політикою контенту та визначеною областю дії Service Worker. Серверна реалізація при цьому має відтворювати атомарний контракт ідемпотентності. Інші предметні області, серверні технології й моделі синхронізації потребують окремої перевірки. Наступним кроком має бути перевірка перенесення фреймворку в інших серверних реалізаціях.

Конфлікт інтересів: Автори декларують, що не мають конфлікту інтересів, зокрема фінансового, особистого, авторського чи будь-якого іншого характеру, який міг би вплинути на дослідження, а також на результати, опубліковані в цій статті

Фінансування: Дослідження проводилося без фінансової підтримки

Доступність даних: Код моделювання, використаний для одержання наведених результатів, подано у відкритому репозиторії, дані модельних реалізацій, узагальнені результати, параметри експерименту та сформовані графічні матеріали доступні в репозиторії https://github.com/prylipa-artem/policies-framework

Використання засобів штучного інтелекту: Під час підготовки рукопису використовувався інструмент штучного інтелекту (ШІ) OpenAI GPT-5.6 Sol для граматичної та стилістичної перевірки тексту. Усі фрагменти, опрацьовані за його допомогою, були перевірені й відредаговані автором. ШІ не використовувався для генерації тексту, формул і таблиць, формування наукових положень, отримання результатів або формулювання висновків. Наукові положення, результати та висновки є власним інтелектуальним внеском авторів

СПИСОК ЛІТЕРАТУРИ

1. Malavolta, I., Procaccianti, G., Noorland P. & Vukmirovic, P. “Assessing the Impact of Service workers on the energy efficiency of progressive Web Apps”. *MOBILESoft*. 2017. p. 35–45. DOI: https://doi.org/10.1109/MOBILESoft.2017.7.

2. Malavolta, I., Chinnappan, K., Jasmontas, L., Gupta, S. & Karam Soltany, K. A. “Evaluating the Impact of caching on the energy consumption and performance of progressive Web Apps”. *MOBILESoft*. 2020. p. 109–119. DOI: https://doi.org/10.1145/3387905.3388593.

3. de Munk, O., Scoccia, G. L. & Malavolta, I. “The state of the art in measurement-based experiments on the mobile web”. *Information and Software Technology*. 2022; 149: 106944, https://www.scopus.com/pages/publications/85131100664.
DOI: https://doi.org/10.1016/j.infsof.2022.106944.

4. “World Wide Web consortium, service workers nightly”. – Available from: https://www.w3.org/TR/service-workers/. – [Accessed: 6 August 2025].

5. Fielding, R. T., Nottingham, M. & Reschke, J. *HTTP Caching*. *RFC 9111*. 2022. DOI: https://doi.org/10.17487/RFC9111.

6. Mukherjee, S., Raj, N. J., Govindraj, K., Deligiannis, P., Ravichandran, C., Lal, A., Rastogi, A. & Krishnaswamy, R. “Reliable state machines: A Framework for programming reliable cloud services”. *ECOOP*. 2019; 134 (18): 1–18. DOI: https://doi.org/10.4230/LIPIcs.ECOOP.2019.18.

7. Zhang, H., Cardoza, A., Chen, P. B., Angel, S. & Liu, V. “Fault-tolerant and transactional stateful serverless workflows”. *OSDI* 20. 2020. p. 1187–1204. – Available from: https://www.usenix.org/conference/osdi20/presentation/zhang-haoran. – [Accessed: 6 August 2025].

8. Wong, T., Wagner, M. & Treude, C. “Self-adaptive systems: A systematic literature review across categories and domains”. *Information and Software Technology*. 2022; 148: 106934. DOI: https://doi.org/10.1016/j.infsof.2022.106934.

9. Weyns, D. & Iftikhar, M. U. “ActivFORMS: A Formally Founded Model-Based Approach to Engineer Self-Adaptive Systems”. *ACM Transactions on Software Engineering and Methodology*. 2023; 32 (1): 12, https://www.scopus.com/pages/publications/85152593093. DOI: https://doi.org/10.1145/3522585.

10. Abbas, N., Andersson, J. & Weyns, D. “ASPLe: A methodology to develop self-adaptive software systems with systematic reuse”. *Journal of Systems and Software*. 2020; 167: 110626, https://www.scopus.com/pages/publications/85084955201.
DOI: https://doi.org/10.1016/j.jss.2020.110626.

11. Meyer, B. “Applying “Design by Contract””. *Computer*. 1992; 25 (10): 40–51. DOI: https://doi.org/10.1109/2.161279.

12. Parnas, D. L. “On the criteria to be used in decomposing systems into modules”. *Communications of the ACM*. 1972; 15 (12): 1053–1058. DOI: https://doi.org/10.1145/361598.361623.

13. Chen, X., Usman, M. & Badampudi, D. “Understanding and evaluating software reuse costs and benefits from industrial cases – A systematic literature review”. *Information and Software Technology*. 2024; 171: 107451. DOI: https://doi.org/10.1016/j.infsof.2024.107451.

14. Hellerstein, J. M. & Alvaro, P. “Keeping CALM: When distributed consistency is easy”. *Communications of the ACM*. 2020; 63 (9): 72–81. DOI: https://doi.org/10.1145/3369736.

15. Kleppmann, M., Wiggins, A., van Hardenberg P., McGranaghan, M. “Local-first software: You own your data, in spite of the cloud”. *Onward*. 2019. p. 154–178. DOI: https://doi.org/10.1145/3359591.3359737.

16. “Chrome for Developers, workbox-background-sync”. – Available from: https://developer.chrome.com/docs/workbox/modules/workbox-background-sync/. – [Accessed: 6 August 2025].

17. “PouchDB Project, About PouchDB”. – Available from: https://pouchdb.com/learn.html. – [Accessed: 6 August 2025].

18. Chinprutthiwong, P., Huang, J. & Gu, G. “SWAPP: A new programmable playground for web application security”. *31st USENIX Security Symposium*. 2022. p. 2029–2046. – Available from: https://www.usenix.org/conference/usenixsecurity22/presentation/chinprutthiwong. – [Accessed: 6 August 2025].

19. Prylipa, A. & Filatova, A. «Клієнтський TinyML профайлер мережевих характеристик у веббраузері». *Системи управління, навігації та зв’язку*. 2025; 4 (82): 114–120. DOI: https://doi.org/10.26906/SUNZ.2025.4.114.

20. Prylipa, A. O. & Filatova, H. E. “Adaptive content optimization and degradation policies for web applications under unstable network conditions”. *Optoelectronic Information-Power Technologies*. 2026; 51 (1): 7–17. DOI: https://doi.org/10.31649/1681-7893-2026-51-1-7-17.

21. Park, J., An, S., Youn, D., Kim & G., Ryu, S. “JEST: N+1-Version differential testing of both JavaScript engines and specification”. *ICSE*. 2021. p. 13–24. DOI: https://doi.org/10.1109/ICSE43902.2021.00015.

22. Chen, T. Y., Kuo, F.-C., Liu, H., Poon, P.-L., Towey, D., Tse, T. H. & Zhou, Z. Q. “Metamorphic testing: A review of challenges and opportunities”. *ACM Computing Surveys*. 2018; 51 (1): 4. DOI: https://doi.org/10.1145/3143561.

23. Efron, B. “Bootstrap methods: Another look at the Jackknife”. *The Annals of Statistics*. 1979; 7 (1): 1–26. DOI: https://doi.org/10.1214/AOS/1176344552.

24. Chen, H., Dou, W., Wang, D. & Qin, F. “CoFI: Consistency-guided fault injection for cloud systems”. *ASE*. 2020. p. 536–547. DOI: https://doi.org/10. https://doi.org/1145/3324884.3416548.

Отримано 23.06.2026
Отримано після перегляду 29.07.2026
Прийнято 12.08.2026
DOI: https://doi.org/10.15276/ict.03.2026.03
UDC 004.415.2:004.738.5

A contract-based adaptive policy framework for progressive web applications under unstable network conditions

Artem O. Prylipa[1)]
ORCID: https://orcid.org/0009-0005-6633-8308; Artem.Prylipa@cs.khpi.edu.ua
Anna E. Filatova[1)]
ORCID https://orcid.org/0000-0003-1982-2322; Hanna.Filatova@khpi.edu.ua. Scopus Author ID: 56448583600
[1)] National Technical University "Kharkiv Polytechnic Institute", 2, Kyrpychova St. Kharkiv, 61002, Ukraine

ABSTRACT

The paper examines the transition from an application-specific adaptive-policy method to a reusable framework for progressive web applications in which unstable network connectivity affects data retrieval, content volume, and the completion of write operations. The previous method was implemented for a single application and therefore did not allow evaluation of core reuse, consistent policy activation across the window context and the Service Worker, or the correctness of deferred operations under failures. A contract-based software framework is proposed that combines a shared core with an application specification containing an operation registry, a policy, adapters, and backend capabilities. The experimental evaluation included comparison with a recorded behavioral baseline, controlled fault injection, and verification of core reuse across two web applications. Complete decision agreement and identical browser outcomes were obtained for all pairs in the main experiment, while timing overhead remained within the protocol-defined limits. All 54 fault-matrix executions produced the expected outcomes, and repeated delivery under an atomic backend contract caused no more than one state change per idempotency key. Both web applications used the same core and passed the shared checks. The results confirm the reusability of the framework; however, transferring this conclusion to other backend implementations requires separate validation.

Keywords: computer system; PWA; Service Worker; adaptive policies; contract-based framework; deferred operations; unstable networks

ІНФОРМАЦІЯ ПРО АВТОРІВ

Приліпа Артем Олегович - аспірант кафедри Комп'ютерної інженерії та програмування. Національний технічний університет «Харківський політехнічний інститут», вул. Кирпичова, 2. Харків, 61002, Україна
***Галузь досліджень*:** комп'ютерна інженерія

Artem O. Prylipa - PhD Student, Department of Computer Engineering and Programming. National Technical University "Kharkiv Polytechnic Institute", 2, Kyrpychova St. Kharkiv, 61002, Ukraine
ORCID: https://orcid.org/0009-0005-6633-8308; Artem.Prylipa@cs.khpi.edu.ua
Research field: computer engineering

Філатова Ганна Євгенівна - доктор технічних наук, професор кафедри Комп'ютерної інженерії та програмування. Національний технічний університет «Харківський політехнічний інститут», вул. Кирпичова, 2. Харків, 61002, Україна
***Галузь досліджень*:** комп'ютерна інженерія

Anna E. Filatova - Doctor of Engineering Sciences, Professor, Department of Computer Engineering and Programming. National Technical University "Kharkiv Polytechnic Institute", 2, Kyrpychova St. Kharkiv, 61002, Ukraine
ORCID: https://orcid.org/0000-0003-1982-2322; Hanna.Filatova@khpi.edu.ua. Scopus Author ID: 56448583600
Research field: computer engineering