УДК 004.415

О.А. КРАВЧУК
Хмельницький національний університет

ОГЛЯД ТА АНАЛІЗ АРХІТЕКТУРИ ПРОГРЕСИВНИХ ВЕБ-ДОДАТКІВ

В статті розглядається технологія Progressive Web Apps, що є найбільш сучасною тенденцією в програмуванні мобільних веб-додатків. Все більше великих компаній додають підтримку даної технології в своїх основних ресурсах в тому чи іншому вигляді. В статті розглянуто основні принципи роботи даної технології, висвітлено переваги перед «нативними» додатками мобільних пристроїв.

Ключові слова: PWA, прогресивний веб-додаток, новітня технологія, висока швидкодія, оптимізація, автономний додаток, мобільний додаток, гаджет, веб-розробка.

O.A. KRAVCHUK
Khmelnytsky National University

REVIEW AND ANALYSIS OF ADVANCED ARCHITECTURE OF WEB APPLICATIONS

In the article we review a new technology as call a Progressive Web Apps. This format for creating mobile sites attracted attention due to the relative ease of development and almost instantaneous interaction with the user. The primary purpose of PWA is to increase conversions, number of users, and the convenience of using Web-based applications on mobile devices. It is the most current trend in programming of mobile web applications. More and more big companies added support of this technology in their main websites The proposal to install a progressive application is only displayed when the web application meets certain criteria and the user has shown interest by re-visiting the site. Installing the application is instantaneous. All components that require a long boot already have been installed in the cache when the user first visits the site. Progressive applications are much smaller in size, since they effectively use browser capabilities. Pop-up messages, offline work and many other advanced features will work even if the visitor has never installed it. This article discusses the basic principles of this technology, highlighted the benefits to the "native" mobile applications.

Keywords: PWA, progressive web apps, progressive technology, high-speed mobile web, optimization, offline application, mobile application, gadget, web development.

Постановка проблеми. Уявіть, що ваш сайт взаємодіє з користувачем як додаток [2]. Тобто користувач може встановити його на будь-який гаджет, отримувати повідомлення і працювати з ним (навіть без інтернет-з'єднання). Це вже реальна технологія, яку активно просуває Google. В цій статті проаналізуємо її в деталях [4, 5].

Progressive Web Apps (PWA) Google анонсував ще в 2015 році. Цей формат створення мобільних сайтів привернув увагу завдяки відносній простоті розробки і майже миттєвій взаємодії з користувачем. Уже в травні 2016 на конференції розробників Google I / O The Washington Post продемонстрував свій мобільний гібридний сайт-додаток.

З'ясуємо детальніше, що саме являє собою PWA [3]. Це назва групи додатків, які використовують стек Web-технологій (JS + HTML + CSS) і дозволяють поєднати простоту використання Web сайту зі специфічними для програмних додатків операційної системи інтерфейсом і технічними можливостями.

Основне призначення PWA – збільшити конверсію, кількість користувачів і зручність використання Web-додатків на мобільних пристроях.

Progressive Web Apps є логічним продовженням Accelerated Mobile Pages. Таким чином, якщо ви раніше створювали AMP додатки, то вам однозначно варто оновити їх до норм PWA додатків. Якщо ви до цього нічого не чули про AMP, то це не стане для вас проблемою під час вивчення PWA.

Аналіз досліджень. Якщо у вас є смартфон, подивіться на іконки-квадрати на головному екрані. Вони називаються «нативними» або «рідними додатками». «Нативні» вони тому, що розроблені для операційної системи вашого смартфона (будь-то iOS або Android). Наразі створений прогресивний веб-додаток, який виглядає і поводиться так само, як мобільний додаток. А це означає, що його можна додати до головного екрана смартфона, йому дозволяється відправити push-повідомлення, отримувати доступ до апаратних засобів пристрою і працювати в автономному режимі. Progressive Web App працює так само стабільно при нестійкому з'єднанні або за відсутності мережі, так як це було б з повним доступом в Інтернет. Як же браузер може відкрити веб-сайт, не маючи доступу до інтернету, запитаєте ви? Вірите чи ні, ваш браузер тепер здатний на багато цікавих функцій, яких ви б не очікували всього пару років тому. Сайти, подібні whatwebcando.today, здатні аналізувати API вашого браузера і показувати, які з функцій доступні на мобільних пристроях і підтримуються браузером.

Припустимо, ваш браузер підтримує всі ці функції. Але в смартфонах так багато нативних додатків, які використовуються щодня, що люди не зможуть просто кинути все це і переключитися на PWA. Але, ймовірно, і тут ми помиляємося. Дослідження, проведене в 2015 році marketingland.com показало дивовижну статистику: мобільні користувачі витрачають 80% часу на своїх пристроях, використовуючи тільки три пріоритетних нативних додатки. Отже, безліч нативних додатків не використовувалися. Доводиться проходити процедуру скачування, а в подальшому видалення додатків, які не стали актуальні для вас з часом. Пріоритетними для більшості з нас залишаються кілька мобільних додатків, такі як Facebook, Instagram, поштовий додаток або інші. Кількість завантажень нових нативних додатків з року в рік падає. За останні роки вона впала на 20% і ця цифра продовжує збільшуватися. У 2016 році число завантажених додатків стало менше, ніж число віддалених.

Формулювання цілей статті. Мета дослідження – огляд та аналіз нових можливостей мобільного інтернету при створенні та використанні прогресивних веб-додатків.

Виклад основного матеріалу дослідження. Але що насправді являють собою PWA? Чи потрібно

вам такий додаток? І, якщо ви його створите, як забезпечити йому високі позиції в пошуковій видачі? Отже, таке поняття, як «Прогресивний веб-додаток» є трохи розпливчатим і дещо суперечливим. Але в цілому, ідея полягає в тому, що веб-сторінка з відповідними модифікаціями згідно зі стандартом розробленим Google, поводилась, як звичайна програма встановлена з App Store. Спочатку сторінка відкривається, як зазвичай у вікні браузера, але якщо вона володіє функціоналом PWA, користувач зможе дозволити браузеру «Додати на робочий стіл» цю сторінку. Раніше даний функціонал використовувався в якості закладок.

Коли встановлений таким чином веб-додаток буде запущено з робочого столу, інтерфейс браузера буде захований, а сторінка відкриється як звичайно встановлений додаток. Отже, PWA є спробою створити баланс між інтернетом і нативним середовищем.

«Progressive Web Apps» повноцінно використовують сучасні можливості Інтернету і надають користувачеві відчуття роботи з додатком. Цікавою властивістю PWA є те, що завдяки їм відпадає потреба створювати додатки. Мобільна версія вашого сайту сама стає додатком. Ось недавно, наприклад, ми з колегами вели досить цікаву дискусію на тему того, що сьогодні потрібніші бренду: сайт з додатком або PWA.

При цьому враховувалася можливість надсилати повідомлення, синхронізація, робота в режимі офлайн, інтерфейс, що створює враження "рідного" додатка, а також опція встановлення ярлика на робочому столі пристрою. Ці можливості були раніше недоступні для сайтів. Але завдяки тому, що нові браузери підтримують HTML5 і JavaScript, ми можемо приступати до створення такого функціоналу.

За інформацією Google PWA включають в себе дві складові: Service Workers і Application Shell Architecture (архітектура оболонки додатку). Розглянемо детальніше ці дві складові додатку.

Одна з найважливіших проблем, від якої страждали користувачі веб-додатків – це робота в умовах втрати зв'язку. Найкращий в світі веб-додаток забезпечить жахливий користувальницький досвід, якщо ви не зможете його завантажити. Робилося багато спроб створення технологій, які б вирішили цю проблему.

Попередньою спробою була технологія AppCache, і вона здавалася гарною ідеєю, тому що дозволяла дійсно просто вказати ресурси для кешування. Однак, ця технологія робить багато припущень про те, що ви намагаєтеся зробити і потім не спрацьовує, коли ваш додаток працює не в точності з цими припущеннями.

Технологія Service Workers повинна в підсумку вирішити проблеми, як спіткали AppCache. Синтаксис Service Worker складніший, ніж той же AppCache, але компромісом є те, що ви можете, за допомогою JavaScript, контролювати AppCache; це дозволяє прогнозувати поведінку додатку з високим ступенем деталізації, що дозволяє вирішувати проблеми кешування. Використовуючи Service Worker ви можете без зусиль отримати додаток, використовуючи, в першу чергу, кешовані ресурси, представляючи тим самим поведінку за замовчуванням в автономному режимі, до того як буде отримано по мережі більше даних (такий підхід називається Offline First). Так зазвичай працюють нативні додатки, що часто є причиною вибору користувача на їх користь.

Service worker це подієво-керований процес, що реєструється на рівні джерела. Він приймає форму JavaScript-файлу, який може контролювати веб-сторінку / сайт, з яким він асоціюється, перехоплювати і модифікувати навігацію та запити ресурсів, дуже докладно кешувати ресурси, й так само надавати повний контроль над тим, як додаток поводиться в ситуації, коли мережа недоступна.

Service worker запускається в контексті workera, тому він не має доступу до DOM і запускається в окремому потоці від основного JavaScript, який керує вашим додатком так, що не блокує його. Він покликаний бути повністю асинхронним і, як наслідок, API, такі як синхронні XHR і localStorage, не можуть бути використані в service worker.

Service workers запускаються тільки поверх HTTPS з міркувань безпеки. Оскільки мати відкриті широкій публіці мережеві запити може бути вкрай небезпечно і призвести до атак.

Коли користувач вперше запросить доступ до сайту / сторінки, контрольованої server worker, той моментально буде завантажений та спочатку реєструє себе. Якщо реєстрація пройшла успішно, service worker буде завантажений клієнтом і спробує встановитися / активуватися для всіх URL, доступних користувачеві, або підмножини, зазначеної ним. Після цього він буде завантажуватися кожні 24 години або близько того. Він спробує встановитися заново, якщо процес буде визнаний новим, або відмінним від існуючого service worker (визначається на основі побайтового порівняння).

Якщо вже існують доступні service worker, нова версія встановлюється у фоновому режимі, але буде все ще не активна – в даний момент вона буде називатися worker в очікуванні. Він активується тільки тоді, коли більше немає інших сторінок для завантаження, та все ще використовують старий service worker. Як тільки таких сторінок більше не залишиться, новий service worker активується (стане активним worker).

Визначимо основні моменти застосування Service workers:

- фонова синхронізація даних;
- відповідь на запити від зовнішніх джерел;
- отримання централізованого оновлення для даних, складних для розрахунків, таких як геолокація або гіроскоп;
- збірка клієнтської частини і управління залежностями для CoffeeScript, less, CJS / AMD модулів розробки;
- поліпшення продуктивності, наприклад, за рахунок попереднього завантаження ресурсів, які знадобляться користувачу в найближчому майбутньому (підвантаження кількох наступних картинок в фотоальбомі);
- реакції на вхідне повідомлення: запускає service worker для відправки повідомлень користувачам, щоб оповістити їх про надходження нового контенту.

В майбутньому service workers будуть здатні на багато більше корисних речей для web-платформ, наблизивши їх до функціонування нативних додатків.

Другою частиною прогресивного веб-додатку, як було зазначено вище, є Application Shell Architecture (архітектура оболонки додатку) – це мінімальний набір HTML, CSS і JavaScript, необхідних для відображення головної сторінки додатку.

Оболонка додатку (Application Shell) є секретом надійної продуктивності. Уявіть, що оболонка додатку – це пакет, який ви опублікували б в магазині додатків, якби розробляли «рідний» додаток. Інтерфейс користувача при цьому залишається локальним, а контент динамічно завантажується через API.

Цей метод завантаження контенту надає неймовірно високу швидкість прийому даних. Ми можемо практично моментально відобразити перед очима користувача то, як виглядає наш сайт але без контенту. Потім сторінка завантажить контент і все готово.

Коли ви заходите в Інтернет і відкриваєте веб-сайт, ви чекаєте завантаження всієї головної сторінки. Це включає в себе не тільки динамічний контент сторінки, але і всі зображення, шрифти, таблиці стилів, JavaScript, що використовуються на сторінці і більшість з них залишаються незмінними незалежно від того, скільки разів ви відкриваєте сайт. Так чому б не кешувати все це? Коли PWA запускається вперше, воно поміщає всі статичні ресурси і оболонку додатку в кеш. Наступного разу, коли додаток буде запущено, воно підтягне статичну інформацію додатку безпосередньо з кешу, що значно зменшить час умовного завантаження для користувача. Якщо ви коли-небудь намагалися відкрити веб-сайт не в 3G-з'єднанні, ви зрозумієте, що я маю на увазі.

Окрім кешування завантажень PWA керують також відповіді. Це, можливо, найпотужніша річ, на яку здатні PWA. Вони можуть кешувати не тільки статичні компоненти оболонки додатку, а й цілі відповіді з GET-запитів. Припустимо, ви вчора відвідали новинне PWA, щоб прочитати новини. Якщо ви відкриєте його сьогодні, миттєво отримаєте новинний канал вчорашнього дня, доки додаток завантажує новий контент в фоновому режимі, динамічно впроваджуючи його в канал, який ви вже переглядаєте. Якщо ви не можете отримати свіжий контент, наприклад, тому, що перебуваєте в автономному режимі, то залишитеся з вчорашнім каналом, але принаймні не отримаєте ніяких помилок. Існують різні алгоритми кешування, які реалізуються в залежності від мети застосування.

Розглянемо ці алгоритми:

- Cache with fallback to network – «кеш з резервом для мережі». Використовується, якщо ви створюєте автономний додаток. Якщо відповідь вже знаходиться в кеші, вона буде передана користувачеві, і онлайновий запит ніколи не буде виконано. Якщо відповідь ще не кешували, додаток спробує завантажити її он-лайн і потім помістить в кеш. Цей підхід слід використовувати для контенту, який змінюється дуже рідко або не змінюється взагалі;
- Network with fallback to Cache – «Мережа з відкотом до кешу». Це підхід, при якому он-лайн користувачі завжди отримують актуальну он-лайн-версію, а автономні користувачі отримують кешовану версію. Слід використовувати його для ресурсів, які часто оновлюються;
- Cache and Network race – «Кеш і мережева гонка». Це коли ви шукаєте відповідь в кеші, одночасно запрошуючи он-лайн-контент. Спочатку ви показуєте кеш відповідь користувача, а потім замінюєте його новим контентом відразу після його появи, або додаєте новий контент поверх кешованих сторінок, таких як Facebook і Twitter.

Коротко розглянемо процес інсталяції прогресивного веб-додатку на пристрій. По суті, встановлення веб-додатку – це додавання «закладки» на домашній екран або в програму запуску додатків. Є деякі досить очевидні речі, які ви, як розробник, повинні надати браузеру, щоб той міг вважати сайт додатком: назва, іконки – тобто ті атрибути, що наявні й в нативного додатку. Всі ці дані пристрій зчитує з маніфесту додатку.

Специфікація маніфесту пропонує вам стандартний спосіб зробити це за допомогою файлу JSON. Просто зробіть посилання на файл маніфесту в HTML-сторінці наступним чином:

```
<link rel="manifest" href="/manifest.json">
```

Маніфест веб-додатку надає інформацію про додаток у форматі JSON-файлу. Мета маніфеста – встановити веб-додаток на домашній екран пристрою, надаючи користувачеві більш швидкий доступ і більше можливостей.

Маніфест веб-додатку є частиною колекції веб-технологій поряд з іншими потужними можливостями, такими як доступ до додатка в режимі оф-лайн і попередженням користувача за допомогою push-повідомлень про зміну вмісту додатку. Наведемо типовий приклад маніфесту веб-додатку:

```
    {
"lang": "ua",
"dir": "ltr",
"name": "Повна назва веб додатку",
"description": "Опис веб-додатку
"short_name": "Коротка назва",
"icons": [{
    "src": "icon/lowres.webp",
    "sizes": "64x64",
    "type": "image/webp"
{
    "src": "icon/hd_hi",
```

```
        "sizes": "128x128"
    }],
    "scope": "/test/",
    "start_url": "/test/start.html",
    "display": "fullscreen",
    "orientation": "landscape",
    "theme_color": "aliceblue",
    "background_color": "red",
    "screenshots": [{
        "src": "screenshots/test.jpg",
        "sizes": "640x480",
        "type": "image/jpeg"
    }
}
```

Розглянемо більш детально основні ключі маніфесту. По-перше, нашому додатку потрібна справжня назва. Для цього використовуються ключі name і short_name.

```
{
    "name": " Повна назва веб-додатку ",
    "short_name": "Коротка назва"
}
```

Ключ short_name служить назвою додатку при відображенні в умовах обмеженого простору (наприклад, під значком на домашньому екрані телефону). Ключ name може бути трохи довший, відображаючи назву програми повністю. Також він служить додатковою інформацією для користувача, який шукає ваш додаток на телефоні. Так що набравши «Повна» або «назва веб-додатку», користувач зможе знайти додаток на своєму пристрої.

Якщо ви не вкажете назву, то браузер може використовувати <meta name = "application-name"> , або дані елементу <title>. Але будьте уважні: деякі браузери можуть вимагати вказати назву по іншому, ваш додаток може втратити статус «прогресивний веб-додаток».

Замість звичайної іконки браузера, у вашого веб-додатка повинна бути іконка, яка буде з ним асоціюватися. Для цього в маніфесті є ключ icons. Він приймає список іконок, їх розмірів і форматів. Це робить процес вибору іконки дуже ефективним, оскільки у них з'являється адаптивне рішення, яке дозволяє уникнути непотрібних навантажень і допомагає іконці завжди виглядати відмінно на широкому діапазоні пристроїв й розширень екрану.

Якщо ви не вкажете ключі іконок, браузер буде шукати запасні варіанти: <link rel = "icon">, <favicon.ico> або, якщо не знайде їх, може навіть використовувати скріншот вашого сайту.

Додатки при запуску повинні мати можливість контролювати своє відображення на екрані. Якщо це гра, то їй, ймовірно, потрібно бути в повноекранному режимі і в горизонтальній орієнтації. Для цього формат маніфесту надає вам два ключа.

```
{
    "display": "fullscreen",
    "orientation": "landscape"
}
```

Іноді під час запуску програми вам потрібно, щоб користувач завжди потрапляв на певну сторінку. Ключ start_url дає можливість це вказати.

```
{
    "start_url": "/start.html"
}
```

Нативні додатки мають чіткі межі: як користувач, ви впевнені, що коли ви відкриваєте нативний додаток, він не відчинить заборонені дії непомітно для вас. Найчастіше вам гранично ясно, що ви переключилися з одного нативного додатку на інший. Зазвичай ці візуальні підказки надає операційна система. З Інтернетом все інакше: це величезна гіпертекстова система, в якій веб-додаток може охоплювати кілька доменів: ви можете з легкістю перейти з gmail.com на docs.google.com і користувач навіть цього не помітить. На практиці, ідея існування кордонів додатків є абсолютно непристосованою для вебу.

В Інтернеті ми знаємо, що покидаємо область однієї програми та переходимо до іншої тільки завдяки веб-дизайнерам, які були досить добрі, щоб зробити їм унікальний помітний дизайн. У випадках, коли це не так, безліч користувачів виявляються обмануті сайтами, що маскуються під інші.

Формат маніфесту вирішує цю проблему, дозволяючи вказувати «область адреси» для застосування. Ця область вказує межі для застосування. Це може бути або домен, або директорія на цьому домені та більш складні функції, які можуть вам стати в нагоді, наприклад, можливість вказати бажану орієнтацію пристрою і чи потрібен вам повноекранний режим.

Як й інші веб-ресурси, маніфест веб-додатку повинен бути доступний для будь-якого веб-браузера або пошукового робота. Якщо розробник веб-додатку хоче сповістити пошукових роботів про заборону на сканування файлу, він може зробити це включивши маніфест веб-додатку в файл robots.txt. Це описано докладніше в протоколі robots.txt.

Отже, ви в захваті від ідеї прогресивного веб-додатку. Але якщо ви створите його, як забезпечити йому високі позиції? Як і з будь-якою новою фронтенд-технологією, тут виникають питання з SEO-видимістю.

Існує кілька проблем, з якими ви можете зіткнутися, якщо плануєте побудувати сайт з використанням архітектури Application Shell. По-перше, вам фактично доведеться використовувати той чи

інший JS-фреймворк або бібліотеку, таку як Angular або React. В цьому випадку вам не завадить вивчити деякі SEO-поради, що стосуються Angular.JS або React. Якщо ви використовуєте щось інше, буде необхідно попередньо візуалізувати сторінки на сервері, а потім зчитувати їх через додаток, коли вони завантажаться. Це дозволить вам використовувати все найкраще, що є в цих інструментах, при цьому надаючи дані, які будуть розпізнаватися Google і іншими пошуковими системами. Незважаючи на недавнє твердження Google, що вони вдосконалюють візуалізацію додатків подібного типу, на практиці ми все ще зустрічаємо безліч прикладів збою при зборі «важких» JS-даних.

Висновок. Припустимо, що ви приєдналися до світу просунутих «фронтендових» JS-технологій, тоді для дії за принципом PWA вам буде потрібно використовувати CSS і JS, необхідні для роботи сторінки, поряд з HTML. Тобто не просто включати теги <script> з атрибутом <code> src, а вбудовувати весь файл.

Зрозуміло, це означає, що ви збільшите розмір сторінки, але, з іншого боку, сторінка буде завантажуватися моментально. Більш того, при моментальному надання всіх JS (необхідних для зчитування) і CSS (необхідних для надання сенсу дизайну), браузер зможе візуалізувати ваш контент та забезпечити його коректне відображення й швидку роботу.

Одна з переваг Progressive Web App полягаю в тому, що його дуже легко виявити, як звичайний веб-сайт – ви знайшли його у пошуковому сервісі, клацнули посилання, щоб відкрити і все, у вас є додаток на пристрої, готовий до використання. При цьому браузер сам запропонує вам додати іконку на робочий стіл. Якщо ви згодні з цим, ви побачите значок нового додатку на головному екрані телефону, перебуваючи поруч з рідними додатками.

При наявності певних складнощів прогресивні веб-додатки мають значну більшість переваг. Так само як і з пошукових систем користувачі можуть переходити на прогресивні програми з посилань в соцмережах під час перегляду контенту, або, безпосередньо, з пошукової видачі соцмережі. Погодьтеся, це набагато зручніше, ніж конкурувати за перегляди з двома мільйонами доступних додатків на IOS App Store або Google Play Store.

Пропозиція встановити прогресивний додаток показується тільки тоді, коли веб-додаток відповідає певним критеріям і користувач продемонстрував інтерес за допомогою повторного відвідування сайту.

Встановлення додатку відбувається миттєво. Всі компоненти, які вимагають тривалого завантаження, вже були встановлені в кеш при першому відвідуванні сайту користувачем.

Прогресивні додатки значно менші за розміром, так як вони ефективно використовують можливості браузера. Спливаючі повідомлення, робота в автономному режимі та багато інших функцій прогресивного додатку будуть працювати, навіть якщо відвідувач ніколи його не встановлював.

Отже, визначимо області, де використання прогресивного веб-додатку надає найбільші переваги в порівнянні з нативними додатками та веб-ресурсами:

- контент, який регулярно оновлюється, на кшталт біржових зведень, швидко мінливих цін або рівня запасів, тощо;
- чат або платформа коментарів з необхідністю оновлення даних, а також повідомлень про нові повідомлення;
- аудиторія, якій властиво витягувати дані, а потім завантажувати їх в оф-лайн режимі, по типу новинних програм або блогу, який публікує багато статей в день;
- сайт з регулярно оновлюваним контентом, який користувачі можуть перевіряти по кілька разів на день.

З розвитком концепції PWA, веб-розробники зможуть швидко створювати та розгортати додатки, які виглядають і працюють однаково добре на всіх платформах.

Література

1. Пасічник В.В. Глобальні інформаційні системи та технології (моделі ефективного аналізу, опрацювання та захисту даних) / Пасічник В.В., Жежнич П.І., Кравець Р.Б., Пелещишин А.М. Тарасов Д.М. – Львів : Вид-во Національного університету "Львівська політехніка", 2006. – 350 с.

2. Особливості web-додатків [Електронний ресурс]. – Режим доступу : http://sites.znu.edu.ua/webprog/lect/1191.ukr.html

3. Прогресивна завантаження web-додатків [Електронний ресурс]. – Режим доступу : http://it-ua.info/news/2016/08/15/progresivna-zavantazhennya-web-dodatkv-za-dopomogoyu-podlu-kodu.html

References

1. Pasichnyk V.V. Hlobalni informatsiini systemy ta tekhnolohii (modeli efektyvnoho analizu, opratsiuvannia ta zakhystu danykh) / Pasichnyk V.V., Zhezhnych P.I., Kravets R.B., Peleshchyshyn A.M. Tarasov D.M. – Lviv : Vyd-vo Natsionalnoho universytetu "Lvivska politekhnika", 2006. – 350 p.

2. Features of web-applications [Electronic resource]. - http://sites.znu.edu.ua/webprog/lect/1191.ukr.html

3. Progressive download of web-applications [Electronic resource]. – http://it-ua.info/news/2016/08/15/progresivna-zavantazhennya-web-dodatkv-za-dopomogoyu-podlu-kodu.html

Рецензія/Peer review : 09.09.2017 р. Надрукована/Printed :25.10.2017 р.
Рецензент: стаття прорецензована редакційною колегією