**Поздній О. В., Сало А.М.,**

Національний університет "Львівська політехніка"

кафедра електронних обчислювальних машин

**Програмна система керування вмістом мобільного додатку**

*© Поздній О. В., Сало АМ., 2020*

**Розглянуто проблему реалізації програмної системи керування вмістом мобільного додутку. Проаналізовані способи, технології та можливості реалізаціїї системи.**

**Ключові слова: мобільний застосунок, система керування вмістом.**

**Keywords: mobile application, content management system.**

**The problem of building a program content management system for a mobile application is considered. Approaches, technologies and opportunities to develop such system were also analyzed.**

**Вступ.** Ми живемо в час коли мобільні пристрої та мобільні додатки є невідємною частиною нашого життя. Сьогодні мобільні застосунки відіграють важливу роль на ринку інформаційних технологій та дають можливість використовувати сервіси та послуги через телефони та планшети. Проте розробка та супровід таких застосунків є скадним та дорогим процесом який часто стає недоступним для молодого бізнесу чи стартапу. Основною проблемою при розробці мобільних застосунків є супровід та розробка вмісту, який може змінюватися щогодини. Саме тому розробка зручної та простої системи керування вмістом мобільного застосунку є однією з осоновних задач при розробці мобільних застосунків. Дані системи дозволяють динамічно змінювати вміст додатків без необхідності завантаження великих пакетів даних. Все частіше та частіше виникають концепції та технології для створення гібридних мобільних застосунків. Такі системи поєднують можливості класичних додатків та веб-технологій та мають потенціал для створення гнучких, швидких систем керування вмістом як мобільних так і веб застосунків.

**Стан проблеми.** На сьогодні, жоден з фреймворків для розробки мобільних додатків не намагався фундаментально вирішити проблему прозорої інтеграції веб-технологій в нативні додатки, так як всі вони або повністю орієнтовані на системні можливості мобільних платформ, або цілком покладаються на HTML5. Коли говорять про майбутнє мобільних додатків, зазвичай все крутиться навколо питання про те, який з підходів переможе: заснований на HTML5 або на нативних API. Що характерно, в подібних міркуваннях не піднімається тема співіснування цих двох підходів, і, більш того, не розглядається ефект синергії, який, завдяки спільному використанню різних технологій, дозволить досягати результатів, які нелегко досягти, покладаючись лише на щось одне. Перш за все небхідно зрозуміти , чи нормально це - використовувати можливості HTML і JS в мобільних додатках, і про те, коли може стати в нагоді подібний підхід. Ось кілька ситуацій, коли змішування веб-технологій з нативними можливостями мобільних платформ може виявитися до речі.

Впершу чергу це може стати в нагоді для реалізації деяких частин. Наприклад, WebSocket - це технологія, спочатку орієнтована під веб. Для її використання можна застосувати вбудований в мобільну платформу веб-двигун (WKWebView для iOS і WebView для Android) замість установки сторонньої бібліотеки, яка просто «емулює» WebSocket. При такому підході не потрібно використовувати додаткові бібліотеки - досить застосовати стандартні технології. По-друге, для деяких новітніх технологій поки не створені їх надійні та стабільні мобільні реалізації. Однак, більшість цих технологій доступні в веб-середовищі. В результаті найефективніший спосіб їх інтеграції в мобільні додатки полягає в використанні відповідної JavaScript-бібліотеки. По-третє, при перенесенні веб застосунку на мобільну платформу виявляється, що деякі функції веб-проектів занадто складні для того, щоб відтворити їх з нуля для кожної з цільових мобільних платформ.Наприклад, одна зі сторінок якогось сайту може виявитися надто складною для того, щоб швидко конвертувати її в формат мобільного додатка, в той час як все інше наповнення сайту без проблем портується в мобільний додаток. У подібному випадку було б дуже до речі, якщо б більшу частину програми можна було б створювати засобами мобільної платформи, але якусь особливо складну сторінку можна було б прозоро інтегрувати в додаток в її початковому вигляді.

Нативні мобільні додатки - це потужний засіб подання інформації користувачам, як на телефонах, так і на інтерактивних дисплеях та пристроях, але іноді неникає потреба використовувати веб-технології, що можна зробити за допомогою стандартних веб-контейнерів. Однак, їх інтеграція в додатки - справа непроста. Ось що потрібно для прозорої інтеграції веб-технологій в нативні додатки:

1) Веб-контейнер повинен являти собою природну частину інтерфейсу мобільного додатка. Контейнер повинен виглядати так само, як будь-який інший компонент для користувача інтерфейсу додатка.

2) Батьківський додаток повинен мати можливість контролювати дочірній веб-контейнер. При такому підході додаток, частиною якого є елемент управління дляперегляду вмісту веб-сторінки, буде мати зручні засоби для впливу на нього через ресурси мобільного телефону

3) Дочірній веб-контейнер повинен мати можливість викликати системні події в батьківському додатку. У веб-контейнера, вбудованого в додаток, повинні бути засоби для звернення до нативних API.

**Постановка задачі.**

 Запропонувати підхід для побудови системи керування вмістом мобільного застосунку з використанням веб технології. Запропунувати метод для реалізації двосторонньої взаємодію додатку і веб-контейнера.

**Розв’язання задачі.**

Для розвязання данної задачі було вирішено використовувати систему для керування вмістом Wordpress, та розробити мобільний застосунок на операційній системі Android. Такі застосунки дозволяють використовувати технологію як веб контейнер. Що дозволяє відображати вміст у форматі веб.

Jasonette - це опенсорсний проект, спрямований на створення крос-платформних нативних додатків, заснованих на JSON-розмітці. Фреймворк Jasonette схожий на веб-браузер, але, замість того, щоб перетворювати HTML-код в веб-сторінки, він перетворює JSON-розмітку в нативні додатки для iOS і Android. Схожість з браузером полягає і в тому, що будь-який додаток, заснований на Jasonette, має однаковий базовий код, який займається інтерпретацією різних наборів JSON-даних в процесі формування фінального додатку. Розробнику не потрібно працювати з кодом бібліотеки. Процес створення додатка полягає в підготовці його JSON-опису, яке дозволяє Jasonette, в реальному часі, перетворювати веб-сторінки в нативні додатки. Суть Jasonette також полягає в створенні стандартної мови розмітки, що дозволяє описувати крос-платформні мобільні додатки. В даному випадку нам потрібна мова розмітки, який може точно описати варіанти взаємодії між батьківським додатком і дочірнім веб-контейнером.



Рис 1. Структурна схема

Для того щоб цього досягти, я скористався технологією JSON-RPC, яка дозволила налагодити взаємодію додатку і веб-контейнера. Так як все в Jasonette виражається у вигляді JSON-об'єктів, цілком природним було використовувати стандартний формат JSON-RPC в якості комунікаційного протоколу.

Для того щоб отримати можливість викликати JS-функції, що знаходяться в веб-контейнері, ми оголошуємо дію $ agent.request.



Рис 2. Виклик Агента

Іноді потрібно динамічно впроваджувати JavaScript код в веб-контейнери вже після того, як в них завантажений вихідний HTML-код. Єдиний спосіб взаємодії мобільного застосування та веб-контейнера полягає у використанні API $ agent. Але якщо ви не можете змінити HTML-вміст, єдиний спосіб додавання інтерфейсу $ agent в веб-контейнер полягає в динамічному впровадженні скриптів. Раніше веб-контейнер міг обробляти клацання по посиланнях лише двома способами, перебуваючи в одному з двох режимів. Завдяки можливостям нового веб-контейнера можна задати в $ jason.body.background атрибут action, який містить опис механізму обробки клацань по посиланнях.



Рис 3. Впровадження скриптів

**Висновки.** У даній роботі запропоновано підхід для створення системи керування вмістом мобільного додатку через веб-технології. У даному випадку мова йде про справжні гібридні додатки, коли у одного і того ж додатка може одночасно бути кілька нативних уявлень і кілька веб-уявлень. Крім того, одне подання такого додатка може мати кілька нативних компонентів для користувача інтерфейсу і веб-контейнер, причому, все це виводиться в одному нативном макеті. Також було розроблено структурну схему та наведені приклади релізації поєднання нативних та веб-компонентів.

**Література**

1. Mahesh Panhale., Beginning Hybrid Mobile Application Development. Apress, 2015. PAP.
2. ISBN 10: 1484213157
3. Stark J., Jepson B., MacDonald B., Building Android Apps with HTML, CSS, and JavaScript: Making Native Apps with Standards-Based Web Tools. ISBN-10: 1449316417
4. 3.Берко А.Ю. Застосування маркетингових методів для аналізу життєвого циклу комерційного web-контенту / А. Ю. Берко, В. А. Висоцька / Вісн. Нац. ун-ту «Львівська політехніка» «Комп’ютерні науки та інформаційні технології». – Львів, 2011. –№ 699. – С. 3–12.
5. Горнаков С. Г. Программирование мобильных телефонов на Java: “ДМК Пресс”, 2007. - 331 c.
6. Брюс Еккель. Философия java. 4-е издание: “Питер”, 2009 - 638 с.
7. Mark L. Murphy. Beginning Android: “Apress”, 2010 — 416 c. Paperback – September 26, 2011
8. Хашими С., Коматинени С., Маклин Д. Разработка приложений для Android: “Питер”, 2011. - 738 с.
9. Nigel George., Beginning Django CMS., 1st ed. Edition ISBN-10: 1484216709
10. Brian Fling., Mobile Design and Development: Practical concepts and techniques for creating mobile sites and web apps (Animal Guide) 1st Edition. ISBN-10: 0596155441