DOI: https://doi.org/10.33216/1998-7927-2025-291-5-29-35

УДК 004.41:004.9

ІНТЕГРАЦІЯ WEB-СЕРВІСУ З СОЦІАЛЬНОЮ МЕРЕЖЕЮ

Ратов Д.В.

WEB SERVICE INTEGRATION WITH SOCIAL NETWORK

Ratov D.V.

Стаття присвячена розгляду створення web-сервісу, який організує роботу електронної черги і має функціонал взаємодії із соціальною мережею. Електронна черга є програмно-апаратним комплексом, що дозволяє формалізувати та оптимізувати керування потоком відвідувачів. У статті розглянуто програмну реалізацію системи електронної черги «Е-черга» з функціоналом, інтегрованим із соціальною мережею, може мати такі сфери застосування: операційні зали банків; центри виплат страхових компаній; клієнтські центри операторів стаціонарні оператори; державні установи (наприклад, податкові та реєстраційні служби, посольства та консульські установи); пенсійні фонди; медичні центри; туристичні компанії; візові центри; автосалони, автосервіси; нотаріальні та адвокатські контори; авіа- та залізничні каси; операційні зали відділень зв'язку; багатофункціональні центри надання державних та муніципальних послуг. Для ідентифікації користувачів в системі пропонується створення QR коду в персональних кабінетах, з подальшим адмініструванням даних завдяки сканеру QR коду. Зручним та швидким способом надання користувачам додаткового функціоналу є використання ботів соціальних мереж, які стали невід'ємною частиною будь-якого онлайн-сервісу, інтернет-магазину, інформаційного порталу або блогу. Розроблений програмний функціонал полягає як в отриманні контенту, так і в створенні процесів, які безпосередньо стосуються послуг і ресурсів. Додаткове використання функціоналу соціальних мереж для оповіщення дозволяють користувачам швидко та зручно отримувати актуальну інформацію в режимі реального часу без необхідності завантаження інтерфейсу веб-сервісу системи. Це створює гнучкість та зручність взаємодії з сервісом для користувачів, що забезпечує більш ефективне управління часом та ресурсами клієнтів. У зв'язку з цим інтеграція функціоналу web-систем управління потоками клієнтів із соціальними мережами стає актуальним завданням. Запропоновані методи реалізації системи при створенні сервісу використовують сучасний та перевірений стек технологій web програмування, що дозволяє створювати web-сервіси, які мають високий рівень безпеки при роботі в електронній системі.

Ключові слова: *соціальна мережа, електронна черга, авторизація, JavaScript.*

Вступ. У сучасному цифровому світі, де комп'ютери та смартфони стали невід'ємною частиною нашого повсякденного життя, електронні системи управління набувають все більшої популярності у вигляді web-сервісів або електронних порталів послуг [10]. Особливо такі сервіси важливі при організації ефективнішого управління соціальними процесами та оптимізації часу користувача.

Система електронної черги відрізняється від систем «виклику клієнта» тим, що дозволяє створити і дотримуватися алгоритму керування потоком, вести облік та статистику роботи операторів та інтенсивності потоку, що дозволяє ефективно планувати навантаження на операторів. Крім цього, у системі необхідно передбачити функції управління налаштуваннями системи та її виконавчими модулями [8]. Такі online системи, які представлені web-додатками, працюють і управляються в реальному часі і вимагає серйозного підходу до планування майбутніх функцій і можливостей програми [9], так як

кожна з обраних цільових платформ, як теоретично, так і на практиці, може створювати свої обмеження зумовлені як особливостями архітектури цільової платформи, так і компанією-власником цієї платформи.

Мета дослідження – використання модульного методу створення web-сервісу, який організує роботу електронної черги, що має функціонал взаємодії із соціальною мережею.

Створення системи електронної черги є досить складним технологічним процесом, який вимагає детального планування і має свої особливості [1]. Додаток має бути кросплатформним та мати інтерфейс, який адаптується під характеристики пристрою. Web додаток буде складатися з HTML, CSS та Javascript файлів [11, 12].

Серед переваг даного типу програм можна виділити наступні:

- безліч додатків та інтерактивних компонентів можуть бути написані мовою JavaScript [14] для всіх мобільних платформ;
- програми можуть використовувати такі функції мобільного пристрою, як камера, акселерометр та інші;
- всі HTML, CSS та JavaScript файли можна оновити, не чекаючи затвердження нової версії програми.

При розробці web-програми для підтримки концепції MVC (Model-View-Controller) [7] дотримується поділ даних програми та логіки, що управляє, на три окремих компоненти: модель, уявлення і контролер. Дані JavaScript об'єктів та дані, отримані під час роботи програми, зберігатимуться у компоненті Model. Компонент Controller представлений центральним модулем System.js. Компонент View представлений інтерфейсом системи.

Архітектура програми. Для реалізації поставленого завдання керування потоками клієнтів було розроблено web-сервіс системи електронної реєстрації «Е-черга». Створений сервіс відповідає шаблону архітектури клієнт-сервер [1], який є одним з архітектурних шаблонів програмного забезпечення та став домінуючою концепцією у створенні розподілених мережних додатків та передбачає взаємодію та обмін даними між ними. До особливостей створеного web-програми можна віднести такі критерії:

- наявність всіх необхідних компонентів у одному комплексі програмного забезпечення;
- безпечне передавання даних та зашифрований формат даних для загального огляду;
- використання стандартної технічної бази для клієнт-серверних систем;
- висока швидкість доступу до даних, які можна отримати за допомогою Telegram бота або безпосередньо через веб-сервіс;
- гнучкість та масштабованість, а також можливість розширення конфігурування та нарощування функціоналу системи.

Взаємодія об'єктів клієнтської та серверної сторони. Схема процесу внутрішньої взаємодії між об'єктами JavaScript центрального модуля System.js та екземплярами серверних класів clServer, clTlgrm, clSession представлена на рис. 1. Для забезпечення безпеки системи на клієнтську сторону завантажується тільки інтерфейс системи (web interface). Усі вихідні дані із зашифрованими таблицями формуються та створюються на серверній стороні (об’єкт класу clServer), відповідно до статусу та повноважень користувача, які визначаються за переданими у запиті параметрами авторизації. Для обробки подій інтерфейсу та можливості створення запитів до сервера в програму завантажується центральний модуль System.js. Цей модуль містить об'єкт JavaScript, який створює діалог із сервером (за допомогою запитів await fetch), та забезпечує завантаження модулів з додатковою функціональністю (за допомогою об'єкта Promise).

Для здійснення запитів до сервера у форматі json [5] з боку об'єктів JavaScript використовуються запити прикладного рівня await fetch, що дозволяє створювати асинхронні запити [2]. У той же час за наявності у користувача статусу адміністратора система за допомогою асинхронного об'єкта Promise дозволяє завантажити модулі ctrl-md5.js і qrcode.js, які дають можливість створення хєш-коду і забезпечують роботу з qr-сканером.

На серверній стороні весь функціонал роботи з авторизацією, формуванням даних черги та кабінету користувача забезпечує клас clServer (рис. 1). У ньому також є функція checkHash, яка 128-бітним алгоритмом хешування перевіряє хеш-код надісланих даних від користувача. Конструктор цього класу відкриває доступ до центральної бази даних.

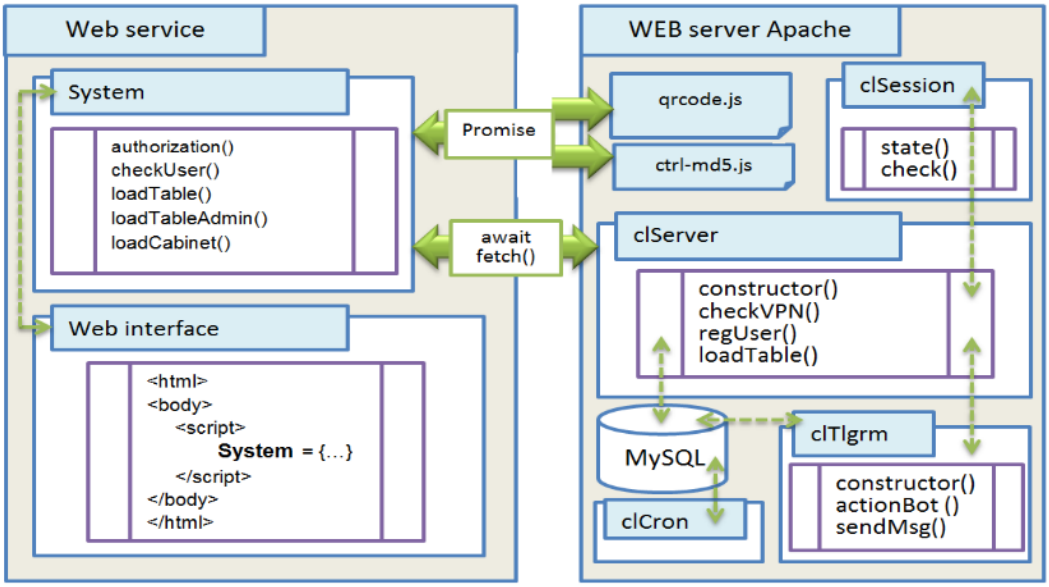

Рис. 1. Схема процесу взаємодії між об'єктами JavaScript та екземплярами серверного класу clServer, clTlgrm, clSession

Функціонал роботи з сесіями забезпечує клас clSession (рис. 1). Цей клас виконує запуск сесії з ідентифікатором користувача, встановлює тимчасову мітку видалення даних поточної сесії, встановлює тимчасову мітку і стежить за кількістю невірних авторизацій для поточної сесії.

Для здійснення взаємодії системи із соціальною мережею використовується функціонал класу clTlgrm (рис. 1), який також задіяний як скрипт web-хука для підключеного бота.

У методі класу sendMessageToId для надсилання повідомлення користувачеві або в певний канал соціальної мережі за заданим ідентифікатором використовується curl_init – стандартна функція запуску сесії cURL, яка запускає нову сесію та повертає дескриптор cURL. Цей дескриптор використовується для роботи з функціями запиту під час встановлення необхідних параметрів - curl_setopt, виконання необхідної операції - curl_exec, та завершення сеансу - curl_close.

Для організації періодичного автоматичного виконання скрипта на сервері використовується механізм Cron. Команда для виконання формується віджетом: wget –O /dev/null --no-check-certificate 'https://cnap.lg.ua/sitepro/php/Cron.php?method=loadYellowList'.

Весь функціонал автоматизації роботи інкапсульований у класі clCron. Із методів цього класу також відкривається доступ до центральної бази даних.

Для зберігання даних у моделі контенту використовується стандартний об'єкт localStorage. Асинхронне сховище даних дозволяє зберігати дані ключа-значення в мобільному пристрої, забезпечуючи постійне зберігання даних навіть після закриття програми. Механізм працює асинхронно, що дозволяє уникнути блокування інтерфейсу користувача при збереженні або отриманні даних. Для забезпечення необхідного захисту від злому у сховищі зберігаються лише дані для авторизації користувача. Використання асинхронного сховища забезпечує у концепції архітектури MVC [7] створення компонента “Model” (рис. 2). Функцію компонента Controller виконує центральний об'єкт System, а компонента View - інтерфейс системи.

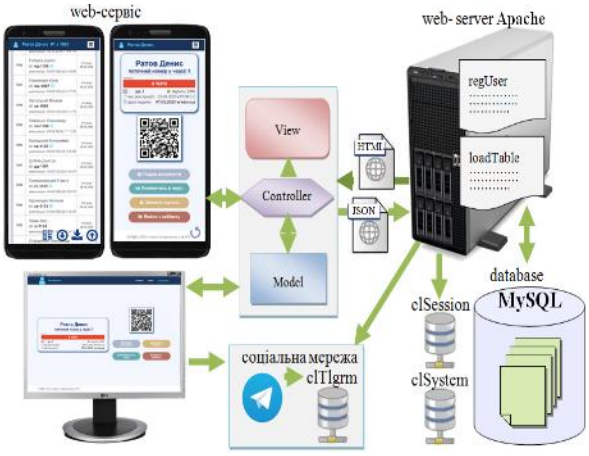

Рис. 2. Схема організації підтримки концепції MVC у контексті web-додатку

Графічний інтерфейс користувача. У інтерфейсі користувача (UI) додаток електронної черги розділений на 4 сторінки з відображенням даних, що спрощують взаємодію користувача з додатком (рис. 3):

1. інформаційна сторінка з довідковою системою (рис. 3 - a);
2. сторінка авторизації користувача (рис. 3 – b);
3. сторінка черги (рис. 3 – c, d, e);
4. сторінка кабінету користувача (рис. 3 – f).

Рис. 3. Графічний інтерфейс користувача:
a – інформаційна сторінка з довідковою системою;
b – сторінка авторизації користувача;
c, d, e – сторінка черги; f – сторінка кабінету користувача

У програмі створена система управління доступом (Access Control Systems), яка дозволяє контролювати доступ до ресурсів та сервісу, встановлювати права доступу для різних користувачів чи груп користувачів. Для цього в модулі System.js створено методи авторизації та аутентифікації, які забезпечують доступ лише авторизованим користувачам. На сторінці авторизації (рис. 3 - b) дані, введені користувачем до відправки на сервер, згідно з дозволеною політикою, проходять валідацію методом модуля System.js - autorization.

У разі запиту на постановку в чергу переданий пакет даних у форматі json [13] в кілька етапів обробляється на сервері. При вході до кабінету система керування доступом робить додатковий запит на перевірку параметрів сесії для запобігання можливості зламування пароля перебором.

Сторінка з таблицею черги для користувача без прав адміністратора (рис. 3 – c) завантажується у зашифрованому вигляді. Кнопки в нижній частині користувача створюють можливість швидкої навігації за списком. При авторизації адміністратора вся таблиця черги завантажується у відкритому форматі (рис. 3 – d) з додатковими можливостями пошуку, зміни та внесення нового користувача (рис. 3 – e). Крім того, у нижній панелі з'являється кнопка включення камери в режимі сканера qr –коду. На сторінці кабінету користувача завантажується його qr –код та вся інформація поточного користувача, який зареєстрований у черзі (рис. 3 – f). Користувач на цій сторінці має можливість змінити свій пароль, зробити запит на зміни свого порядкового номера в черзі, зафіксувати подачу документів або вийти з кабінету.

Програмний сканер QR коду. Модуль QR сканера завантажується у програму лише для користувачів зі статусом «адміністратор». Для завантаження модуля QR сканера використано об'єкт Promise [3]. Для роботи бібліотеки сканера додаток завантажує скрипт qrcode.js (рис. 4).

```
return new Promise( (resolve, reject) => {
    resolve( $$.loadScript( 'js/qrcode.js' ));
})
.then( () => {
    $$.scanQrCode();
})
.finally(() => {
})
```

```
$$.scanQrCode = function(){
    let html5QrcodeScanner = new Html5QrcodeScanner("reader",
        { fps: 10, qrbox: 250 });

    html5QrcodeScanner.render( ( decodedText, decodedResult ) => {
        html5QrcodeScanner.clear();
        window.location.href =
            'https://cnap.lg.ua/php/System.php?data=${decodedText}';
    });
}
```

```
public function setUser()
{
    $_SESSION['qr_pib']   = $this->mysqli->real_escape_string( $this->_PARAM['PIB'] );
    $_SESSION['qr_date']  = date('Y-m-d');
    $_SESSION['qr_time']  = date('H:i:s');

    header("Location: https://cnap.lg.ua/%D0%A7%D0%B5%D1%80%D0%B3%D0%B0/");
}
```

Рис. 4. Динамічне завантаження скрипта QR сканера, подія обробки QR сканера, параметри фільтра в об'єкті сесії

В об'єкті Promise, після завантаження модуля QR сканера, викликається метод scanCode (рис. 4), який використовує наскрізний інтерфейс користувача і функціональність сканування. Для спрацювання сканера в об'єкті Html5QrcodeScanner відстежується подія render – розпізнання коду (рис. 5 – a, b).

Отримані з QR-коду дані в параметрі get-запиту відправляються на сервер, де прикріплюються до об'єкта сесії у вигляді параметрів фільтра поточного користувача і перевантажують сторінку черги з урахуванням цих параметрів фільтра (рис. 4).

Рис. 5 Результат роботи сканера QR-коду: a – сторінка з QR кодом; b – сторінка персонального кабінету користувача; c – сторінка з додатковими даними

На сторінці користувача з правами адміністратора завантажиться інформація про користувача, qr-код якого був отриманий під час сканування на попередньому кроці (рис. 5 – c).

Інтеграція сервісу із соціальною мережею. Для інтеграції сервісу із соціальною мережею реєструється новий бот у соціальній мережі Telegram. Для цього боту з ім'ям @BotFather надсилається команда створення нового бота. Після цього створюється новий бот з унікальним токеном (наприклад, 2555729766:BBC16Fjruw123).

Для підв'язки свого скрипта до створеного бота використовується налаштування Webhook за допомогою GET-запиту до програмного інтерфейсу соціальної мережі: https://api.telegram.org/bot2555729766:BC16Fjruw123setwebhook?url=https://cnap.lg.ua/sitepro/php/TelegramBot.php

Скрипт TelegramBot.php, є джерелом обробки повідомлень і містить клас clTlgrm, який приймає дані надіслані з Telegram (від підключеного через налаштування Webhook бота), здійснює певні дії та відповідає користувачеві шляхом відповідного запиту до BOT API Telegram. Конструктор класу створено таким чином, що клас використовується як самим ботом, так і web-сервісом для надсилання відповіді користувачеві або посилання повідомлення в певний канал соціальної мережі. При взаємодії користувача з роботом у класі clTlgrm використовується метод getAutorize, який за допомогою запиту до бази даних дозволяє встановити факт авторизації користувача в системі та виявити його статус. Метод sendGenericMenu дозволяє створити меню бота (рис. 6 – a), за допомогою якого користувач може перейти на певну сторінку web-сервісу системи (авторизація, черга, кабінет), а також отримати актуальну інформацію в режимі реального часу без необхідності завантаження веб-інтерфейсу самої системи (кнопка «Стан»).

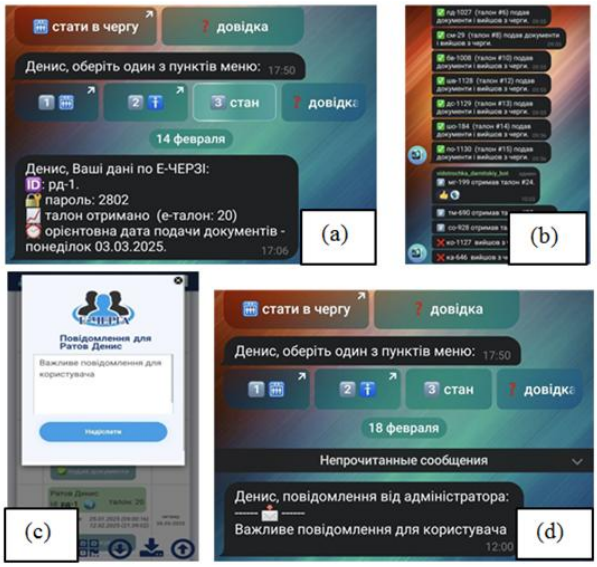

Рис. 6. Меню та діалог з повідомленнями бота, інтегрованого з web-сервісом: a – сторінка спілкування користувача з ботом; b – сторінка з надісланими повідомленнями з web-системи в соціальну мережу; c – сторінка створення повідомлення з web-системи обраному користувачеві; d – діалог з ботом у соціальній мережі

Для надсилання повідомлення з класу clTlgrm використовується get-запит до програмного інтерфейсу соціальної мережі: $url = "https://api.telegram.org/bot{$botToken}/sendMessage?chat_id={$chatId}&text={$message}&message_thread_id ={$message}, де $chatId – ідентифікатор користувача чи каналу, $message – повідомлення, кодоване функцією urlencode(), $messageThreadId – ідентифікатор додаткової теми каналу.

Для запуску нової сесії та отримання дескриптора cURL на сервері використовується стандартна функція запуску сесії curl_init. Далі працює обробка запиту до зовнішнього прикладного програмного інтерфейсу по ланцюжку curl_setopt-curl_exec-curl_close.

Тут curl_setopt – функція встановлення необхідних параметрів запиту, curl_exec – функція виконання необхідної операції,curl_close - функція завершення сеансу (рис. 7).

Такий функціонал дозволяє надіслати повідомлень як користувачу (при спілкуванні користувача з ботом (рис. 6 – a)), так і в канал або певну тему каналу (при надсиланні повідомлень з web-системи в соціальну мережу (рис. 6 – b)). Крім того, у адміністратора з'являється можливість надсилати повідомлення з web-системи (рис. 6 – c) обраному користувачеві в його діалог з ботом у соціальній мережі (рис. 6 – d) за допомогою методу sendMesTlgr серверного класу clTlgrm (рис. 7).

```php
$curl = curl_init();
curl_setopt($curl, CURLOPT_URL, $url);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
if( $params ) {
    curl_setopt($curl, CURLOPT_HTTPHEADER,
            ['Content-Type: application/x-www-form-urlencoded']);
    curl_setopt($curl, CURLOPT_POST, true);
    curl_setopt($curl, CURLOPT_POSTFIELDS, http_build_query( $params ));
}
$output = curl_exec( $curl );
curl_close( $curl );
public function sendMesTlgr()
{
    include_once "clTlgrm.php";
    $arPIP = explode(' ', $this->_PARAM['PIBTlgrUser']);
    $objTlgrm = new clTlgrm( false );
    $objTlgrm->sendMessageToId( $this->_PARAM['idTlgrUser'], implode("\n", [
        "{$arPIP[1]}, повідомлення від адміністратора:",
        "------ ✉ ------",
        $this->_PARAM['message']
    ]));
    $this->res->status  = 'successfully';
    $this->res->message = 'Повідомлення надіслано';
    return json_encode( $this->res );
}
```

Рис. 7. Отримання дескриптора cURL на сервері і серверна функція надсилання повідомлень із сервісу до соціальної мережі

Висновки. Описана у статті електронна система управління потоками клієнтів допомагає змінити та підвищити якість обслуговування, дозволяє організувати запис відвідувачів на прийом за часом та датою. Система електронної черги дозволяє на основі отриманих у процесі роботи даних оптимізувати обслуговування або розробити нові методики управління потоками, а також створює можливість оперативного внесення коректив в автоматизацію управління процесом. Результатом застосування системи електронної черги є покращення загального клімату обслуговування та вищий коефіцієнт роботи персоналу установи.

Згідно з отриманою статистикою реєстрації в електронній черзі “Е-черга” протягом першого дня (рис. 8 – a) та протягом трьох тижнів (рис. 8 – b), найбільша кількість зареєстрованих спостерігалась у перші хвилини початку реєстрації.

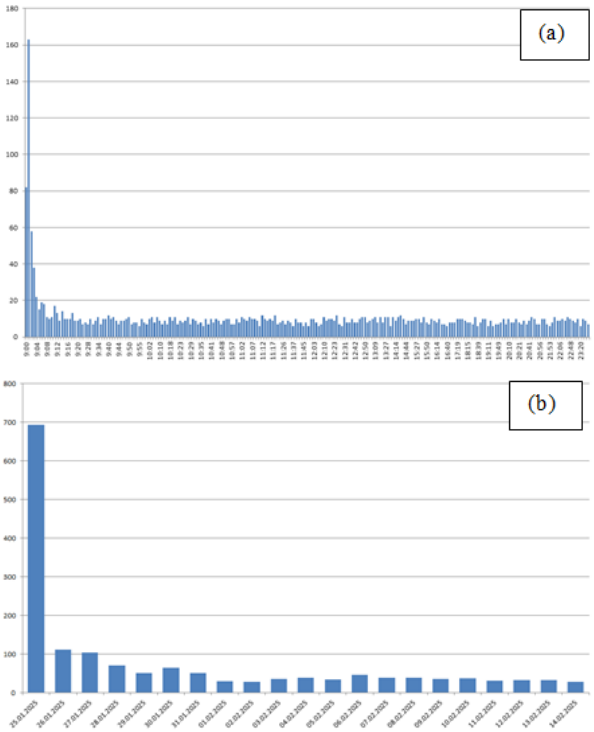

Рис. 8. Статистика реєстрації в додатку “Е-черга”: a – протягом першого дня; b – протягом 21 дня

Запропоновані методи реалізації системи при створенні сервісу використовують сучасний та перевірений стек технологій web програмування, що дозволяє створювати web-сервіси, які мають високий рівень безпеки при роботі в системі.

Література

1. Архітектура Клієнт - сервер. URL: https://ua.frwiki.wiki/wiki/Client-serveur (дата звернення: 27.06.2025).
2. Крейн Д., Паскарелло Еге. Ajax у дії. М: Вид. будинок "Вільямс", 2006.
3. D. Herman, Effective JavaScript, Pearson Education. Addison Wesley, 2012.
4. ECMAScript Language Specification - ECMA-262 Edition 5.1. URL: https://262.ecma-international.org/5.1/. (дата звернення: 27.06.2025).
5. ECMA International.. ECMA-404 The JSON Data Interchange Standard. URL: http://www.json.org/ (дата звернення: 27.06.2025).
6. Герберт Шілдт. Java 8. Повний посібник. М: Вид. будинок "Вільямс", 2015-1376 с.
7. Архітектура MVC. URL: http://www.gwtproject.org/articles/mvp-architecture.html (дата звернення: 27.06.2025).
8. Ratov D.: Architectural paradigm of interactive interface module in the cloud technology model. Applied Computer Science. 2020. Vol. 16, No. 4. P. 48-55.

9. Ratov D.: Integration with the software interface of the com server for authorized user. Applied Computer Science. 2021. Vol. 17, No. 2, P. 5–13.
10. Ратов Д. В. Модель модуля інтерфейсу користувача інформаційної web-системи. Математичні машини і системи. 2020. № 4. С. 74–81.
11. Stoyan Stefanov:. JavaScript. Patterns. 2010. O'Reilly Media. P. 234.
12. Vscode.dev - Visual Studio Code for the Web. URL: https://code.visualstudio.com/blogs/2021/10/20/vscode-dev (дата звернення: 27.06.2025).
13. T. Marrs, JSON at work : practical data integration for the web, O'Reilly Media 2017.
14. JavaScript data types and data structures, MDN Web Docs, Mozilla Foundation. URL: https://developer.mozilla.org/enUS/docs/Web/JavaScript/Data_structures (дата звернення: 27.06.2025).

References

1. Architecture Client-Server. URL: https://ua.frwiki.wiki/wiki/Client-serveur (date accessed: 06/27/2025).
2. Crane D., Pascarello E.: Ajax in action. 2006. M.: Ed. house "Williams".
3. D. Herman, Effective JavaScript, Pearson Education. Addison Wesley, 2012.
4. ECMAScript Language Specification - ECMA-262 Edition 5.1. URL: https://262.ecma-international.org/5.1/ (date accessed: 06/27/2025).
5. ECMA International. ECMA-404 The JSON Data Interchange Standard. URL: http://www.json.org/ (date accessed: 06/27/2025).
6. Herbert Schildt: The Complete Guide. 2015. M.: Ed. house "Williams". P. 1376.
7. MVC architecture. URL: http://www.gwtproject.org/articles/mvp-architecture.html (date accessed: 06/27/2025).
8. Ratov D.: Architectural paradigm of interactive interface module in the cloud technology model. Applied Computer Science. 2020. Vol. 16, No. 4. P. 48-55.
9. Ratov D.: Integration with the software interface of the com server for authorized user. Applied Computer Science. 2021. Vol. 17, No. 2, 5–13.
10. Ratov D.: Model of user interface module of information web system. Mathematical machines and systems. Київ. 2021. No. 4, 74–81.
11. Stoyan Stefanov:. JavaScript. Patterns. 2010. O'Reilly Media. P. 234.
12. Vscode.dev - Visual Studio Code for the Web. URL: https://code.visualstudio.com/blogs/2021/10/20/vscode-dev (date accessed: 06/27/2025).
13. T. Marrs, JSON at work : practical data integration for the web, O'Reilly Media 2017.
14. JavaScript data types and data structures, MDN Web Docs, Mozilla Foundation. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures (date accessed: 06/27/2025).

Ratov D.V. Web service integration with social network

The article is devoted to the consideration of the creation of a web-service that organizes the work of an electronic queue and has the functionality of interaction with a social network. An electronic queue is a software and hardware complex that allows you to formalize and optimize the management of the flow of visitors. The article considers the software implementation of the electronic queue system "E-queue" with functionality integrated with a social network, which can have the following areas of application: bank operating rooms; insurance company payment centers; operator client centers, stationary operators; state institutions (for example, tax and registration services, embassies and consular offices); pension funds; medical centers; travel companies; visa centers; car dealerships, car services; notary and law offices; airline and railway ticket offices; operating rooms of communication departments; multifunctional centers for the provision of state and municipal services. To identify users in the system, it is proposed to create a QR code in personal accounts, with subsequent data administration using a QR code scanner. A convenient and fast way to provide users with additional functionality is to use social network bots, which have become an integral part of any online service, online store, information portal or blog. The developed software functionality consists of both receiving content and creating processes that directly relate to services and resources. Additional use of social network functionality for notifications allows users to quickly and conveniently receive up-to-date information in real time without the need to download the system's web service interface. This creates flexibility and convenience of interaction with the service for users, which provides more effective management of customer time and resources. In this regard, the integration of the functionality of web-systems for managing customer flows with social networks becomes an urgent task. The proposed methods for implementing the system when creating a service use a modern and proven stack of web programming technologies, which allows you to create web services that have a high level of security when working in an electronic system.

***Keywords**: social network, electronic queue, authorization, JavaScript.*

Ратов Денис Валентинович – к.т.н., доцент кафедри інформаційних технологій та програмування, Східноукраїнський національний університет імені Володимира Даля (м. Київ), ratov@snu.edu.ua

Стаття подана 23.05.2025.