УДК 004.42
DOI https://doi.org/10.32838/2663-5941/2020.2-1/35

Ячменьов Я.О.
Державний університет «Житомирська політехніка»

Левківський В.Л.
Державний університет «Житомирська політехніка»

Кравченко С.М.
Державний університет «Житомирська політехніка»

Гришкун Є.О.
Державний університет «Житомирська політехніка»

АВТОМАТИЗАЦІЯ ПРОЦЕСУ ІНТЕГРАЦІЇ MAGENTO 2 У WORDPRESS ЗА ДОПОМОГОЮ ПЛАГІНУ

У цій роботі розглянуто розробку нового програмного забезпечення у вигляді плагіну на базі CMS "WordPress" (PHP/MySQL) та з використанням API Magento 2 (PHP/MySQL) для здійснення автоматичної інтеграції вмісту Magento 2 у WordPress та ліквідації усіх можливих конфліктів між обома системами. Система інтеграції Magento 2 у WordPress створюється з метою максимального збереження можливостей WordPress, інтегрування до даного блогового рушія дизайну із Magento 2 (блоків, контейнерів, JS/CSS), вмісту Magento 2 (продуктів, категорій тощо), зменшення витрат на додаткові ресурси для розробки і підтримки блогу для наявного електронного магазину.

Проведений аналіз предметної області дозволив визначити основні аспекти інтеграції Magento 2 у WordPress. Огляд наявних аналогів та виявлення протиріч між наявними можливостями та потребами показав, що головними функціями майбутньої системи мають бути: повне збереження функціоналу WordPress, інтеграція із Magento версії 2.0–2.3 по дизайну та вмісту, наявність шорткодів та віджетів, наявність зручного GUI, наявність API для розширення та використання іншими системами. Для збереження масштабованості системи інтеграції було взято за основу подійно-орієнтовану архітектуру. Розглянуто алгоритми основних процесів системи та проаналізовано особливості функціонування ядра інтеграції. Наведено детальний розгляд деяких аспектів реалізації.

Розроблений плагін використовуватиметься для забезпечення електронних магазинів на Magento 2 сучасним та багатофункціональним блоґом на системі керування вмістом "WordPress" без додаткових витрат на команду розробників і дизайнерів із боку клієнта, окрім того, даний продукт може бути розділений на дві версії – повну (платна) та неповну (безкоштовна, публічна), що також принесе неабиякий зиск. Система може бути використана для забезпечення електронного магазину повнофункціональним блогом із мінімальними витратами на розробку, або навпаки, для забезпечення блогу електронним магазином.

Ключові слова: *Magento, Magento 2, інтеграція, WordPress, плагін, ядро, PHP.*

Постановка проблеми. Із розвитком мережі інтернет та просуванням її до широкого загалу різкими темпами почала прогресувати електронна комерція (надалі ЕК), що торкнулася всіх куточків нашого життя в наступних своїх проявах: електронний обмін інформацією, електронний рух капіталу, електронна торгівля, електронні гроші, електронний маркетинг, електронний банкінг, електронні страхові послуги тощо. Досить цікавим складником ЕК є електронні магазини. Нині їхній розвиток та потреби сучасного споживача сприяють виникненню великих фреймворків, систем керування вмістом (далі – СКВ), що мають розвинену спільноту програмістів у всьому світі. Прикладом можуть бути такі системи: WooCommerce (у вигляді плагіну під СКВ "WordPress"), Shopify, Magento, PrestaShop та інші. Проаналізувавши перелік платформ для електронних магазинів і збудувавши їхній рейтинг за використанням в порядку спадання, з'ясовується, що перші дві – WooCommerce та Magento [1]. WooCommerce має такий високий рейтинг завдяки дуже відомій і розповсюдженій СКВ для створення блогів – WordPress, але доцільність її використання для створення середнього чи великого магазину підлягає критиці. Щодо Magento, то ця платформа

розроблена саме для створення середньо-великих електронних магазинів.

Після створення достатньої кількості електронних магазинів на Magento 1 було виявлено потребу клієнтів у багатофункціональному блозі, який повинен розміщуватися на одному ж сервері із Magento і мати схожий чи подібний дизайн без залучення великих ресурсів. Варіанти із модулем під Magento 1 одразу не підходили, бо мали недостатньо функціоналу порівняно із WordPress блоґом, погану пошукову оптимізацію сайту (далі ПОС) тощо. Для Magento 1 магазинів був доступний варіант із використанням вже готового плагіну інтеграції з WordPress блогом, що дозволяв перенести дизайн Magento 1 платформи і показувати продукти зі сторони WordPress блогу [2].

Після приходу на ринок електронної комерції нової Magento 2, де було перероблено архітектуру, систему кешування тощо, обране вище рішення із плагіном під WordPress перестало бути актуальним через неможливість роботи із оновленою Magento самого плагіну, а його розробник повідомив аудиторію, що не знає способів ефективного поєднання Magento 2 і WordPress [3].

Аналіз останніх досліджень і публікацій. У статті Івана Сергійовича Степури [4] викладено процес створення сайту Інтернет-медіа «Грінченко-інформ», призначеного для публікації творчих доробків студентів спеціальності «Журналістика». Визначено вимоги до цього ресурсу, порівняно системи керування контентом, описано особливості реалізації сайту та додаткових плагінів, проаналізовано статистику використання ресурсу. В роботі [5] було встановлено, що саме за допомогою плагіну iThemes Security можна подолати будь-які загрози безпеки сайту. Плагін iThemes Security створений для запобігання будь-яким спробам несанкціонованого доступу до WordPress за допомогою різноманітних методів. У статті Євгена Язвинського [6] визначено шляхи підвищення конверсії для будь-яких магазинів, що функціонують у сегменті B2C & B2B, на прикладі системи управління контентом Magento CMS. Наводяться рекомендації стосовно впливу окремих рішень на підвищення показника конверсії. Основними рекомендаціями є: постійний пошук інноваційних рішень; тестування та налаштований зворотній зв'язок з клієнтами, а також переймання передового досвіду західних колег, які є більш обізнаними у сфері електронної комерції; використання бенчмаркінгових заходів на шляху вдосконалення власного ресурсу.

У розглянутих працях авторами застосовано технології WordPress або Magento для розв'язання певних задач, але матеріалів, які б вирішували проблему здійснення автоматичної інтеграції вмісту Magento 2 у WordPress, не знайдено.

Постановка завдання. Після ретельного аналізу ситуації з відсутністю рішення для забезпечення електронного магазину на платформі Magento 2 багатофункціональним блогом було поставлене завдання розробки плагіну під WordPress. Він дозволить ефективно поєднати вміст системи Magento 2 у блозі, створеному на WordPress, що матиме подійно-орієнтовану архітектуру [7] для легшої інтеґрації із плагінами інших WordPress розробників.

Виклад основного матеріалу дослідження. Під час проектування архітектури інтеграції обох систем та її програмування було виявлено конфлікт на базі суперглобальної PHP змінної "$_SERVER" [8], без вирішення якого подальша інтеграція була б неможливою. Тому розглянемо ядро інтеграції, а саме два базових алгоритми, в одному з яких вирішується фундаментальний конфлікт систем.

Основною частиною системи є ядро інтеґрації Magento 2 у WordPress у вигляді статичного класу "M2I_External". Клас "M2I_External" виконує не тільки роль монолітного ядра інтеграції, він є своєрідним мостом між WordPress і Magento, що забезпечує вирішення конфліктів між даними системами. Також цей клас зберігає у собі найважливіші об'єкти Magento, за допомогою яких можливе подальше розширення функціоналу інтеграції. Це ядро вміщує в собі методи, що здійснюють:

− додавання елементів із різних сторінок Magento;

− переведення Magento помилок у WordPress помилки та попередження;

− ініціалізацію Magento;

− збір основних об'єктів для написання API інтеґрації;

− оптимізацію середовища до запуску Magento;

− переклад тексту системою Magento;

− повний запуск Magento;

− вибір потрібної крамниці Magento;

− надання додаткової інформації про посилання Magento, версію цієї системи тощо.

Плагін інтеграції Magento 2 у WordPress повинен мати щонайкраще ядро, яке буде легке у використанні, що дозволить реалізувати увесь подальший функціонал плагіну відповідно до вимог користувача. Варто навести основні методи, що будуть використовуватися найбільше (табл. 1).

Під час реалізації системи інтеграції було найбільш точно відпрацьовано етапи ініціалізації та повного запуску Magento в середовищі WordPress, що дозволило надалі у масштабувати систему інтеграції.

Розглянемо алгоритм ініціалізації Magento в плагіні інтеграції (рис. 1).

Спершу створюється об'єкт для збору помилок, адже ініціалізація зазвичай робиться лише один раз, а під час наступної ініціалізації об'єкт для збору помилок має бути новим, щоб мати можливість проаналізувати різницю. Далі перевіряється функція перекладу в системі WordPress. Якщо клієнт замінив її на варіант, запропонований документацією плагіну інтеграції [9], то ця функція вважається оптимізованою для використання із системою Magento. Інакше генерується попередження, що функція перекладу неоптимізована, а ініціалізація завершується. Після успішної перевірки на функцію перекладу ініціалізується коренева директорія Magento: перевіряється правильність шляху, за умови AJAX виклику зі сторінки базових налаштувань береться директорія, передана із даними POST запиту. Далі ініціалізується Bootstrap об'єкт – один із головних об'єктів Magento, без якого неможливий жоден запуск системи, окрім того, він створює ObjectManager – об'єкт, що дозволяє створювати екземпляри будь-яких класів Magento. Своєю чергою ObjectManager є сховищем екземплярів вже створених об'єктів, а тому він реалізовує твірний шаблон [10] проектування "Object pool" і є дуже важливим для подальшої роботи Magento системи. Під час ініціалізації Bootstrap об'єкту також створюється об'єкт автозавантажувача усіх класів Magento. Якщо Bootstrap об'єкт було ініціалізовано успішно, то його екземпляр буде збережено, ось чому існує перевірка на наявність цього об'єкту. У разі відсутності цього об'єкту процес ініціалізації Magento завершується. За наявності цього

Таблиця 1

Основні методи ядра інтеграції

Метод	Опис та призначення
add_handle(handle, [pageUrl : string = '/']) : void	Додає симуляцію обраної сторінки в Magento по handle з можливістю задання pageUrl за потреби. Цей метод може бути використаний для довантаження ще неіснуючих блоків і контейнерів, адже ядро інтеграції усталено завантажує вміст головної сторінки.
can_launch() : bool	Перевіряє: чи може Magento бути повністю запущеною.
get_app() : \Magento\Framework\App\Http | null	Віддає об'єкт \Magento\Framework\App\Http, якщо він ще не був проініціалізований, то null.
get_base_url([type : string = 'static']) : string	Віддає різні види базових посилань Magento. Використовується для інтеграції вмісту.
get_bootstrap() : \Magento\Framework\ App\Bootstrap | null	Віддає об'єкт \Magento\Framework\App\Bootstrap, що використовується для отримання ObjectManager, призначення якого було описано в розділі 2.3. Якщо Bootstrap непроініціалізований – метод віддає null.
get_error_helper() : M2I_Error_Helper | null	Віддає контейнер, що може містити у собі помилки. Дозволяє перевірити: чи ядро зіткнулося із проблемами під час інтеграції.
get_layout() : \Magento\Framework\View\ LayoutInterface | null	Віддає об'єкт, що дозволяє вибрати усі блоки та контейнери. Дуже корисний об'єкт для інтеграції вмісту. Віддає null за відсутності повного запуску Magento.
get_response() : \Magento\Framework\ App\ResponseInterface | null	Віддає об'єкт, що містить у собі увесь вміст Magento, а саме HTML/CSS/JS код, дуже корисний об'єкт для подальшого розбору такого вмісту на окремі частини і комплексної інтеграції цих частин до вмісту WordPress. Повертає null за відсутності повного запуску Magento.
get_store() : \Magento\Store\Model\Store | null	Віддає об'єкт крамниці Magento. Якщо сталася помилка, то генерує попередження в стилі WordPress та повертає null.
init([is_ajax : bool = false]) : void	Проводить ініціалізацію системи Magento. Без виклику цього методу використання будь-якого API Magento є неможливим.
is_mage_runs_from_root() : bool	Перевіряє налаштування запуску Magento в режимі "root". Якщо Magento запущено в режимі "root", то всі посилання будуть із префіксом "pub".
launch() : void	Проводить повний запуск Magento.
was_launched() : bool	Перевіряє чи повний запуск Magento був проведений успішно.

об'єкту ініціалізація продовжується. Виконується модифікація середовища веб-серверу методом оптимізації суперглобальної змінної $_SERVER для безконфліктної роботи Magento, а попередній стан $_SERVER зберігається. Далі ініціалізується об'єкт App (повна назва \Magento\Framework\App\Http) через виклик "createApplication(…)" методу Bootstrap об'єкту. Відновлюється попередньо збережений стан $_SERVER. Наприкінці встановлюється інформація про можливе здійснення повного запуску Magento записом булевого значення "true" в поле класу M2I_External з назвою "can_launch".

Розглянемо алгоритм повного запуску Magento в плагіні інтеграції (рис. 2).

В алгоритмі повного запуску Magento варто відзначити деякі моменти. Перевірка на те, чи був вже здійснений запуск, дозволяє зберегти час виконання завантаження системи WordPress, бо така перевірка може бути відсутня на якомусь етапі під час зовнішнього використання. Очищення full_page кешу є дієвим лише для Magento нижче 2.2.5 версії і повинно бути видалене в наступних версіях плагіну. Збереження об'єкту Layout в ядрі потрібне для майбутнього отримання блоків і контейнерів при інтеграції вмісту Magento у WordPress. Збереження об'єкту крамниці (Store) потрібне для передачі правильних посилань та інших можливих перевірок. Встановлення потреби перекладу зі сторони Magento потрібне для ліквідації невідповідності параметрів між функцією перекладу WordPress та функцією перекладу Magento, що насправді міститься вже у функції перекладу WordPress, але перемикається параметром, ось чому ще під час ініціалізації Magento перевірка на оптимізацію внутрішньої функції WordPress для перекладів є найпершою умовою.

Розглянемо основні класи системи (табл. 2).

У процесі роботи системи відбувається базова інтеграція Magento 2 у WordPress, а тому є можливим написання прикладної інтеграції у процедурному стилі для забезпечення усіх вимог користувача і API плагіну інтеграції, а також для використання поза системою.

У головному файлі плагіну "magento2-integration.php" реалізовано ряд дій та фільтрів, а

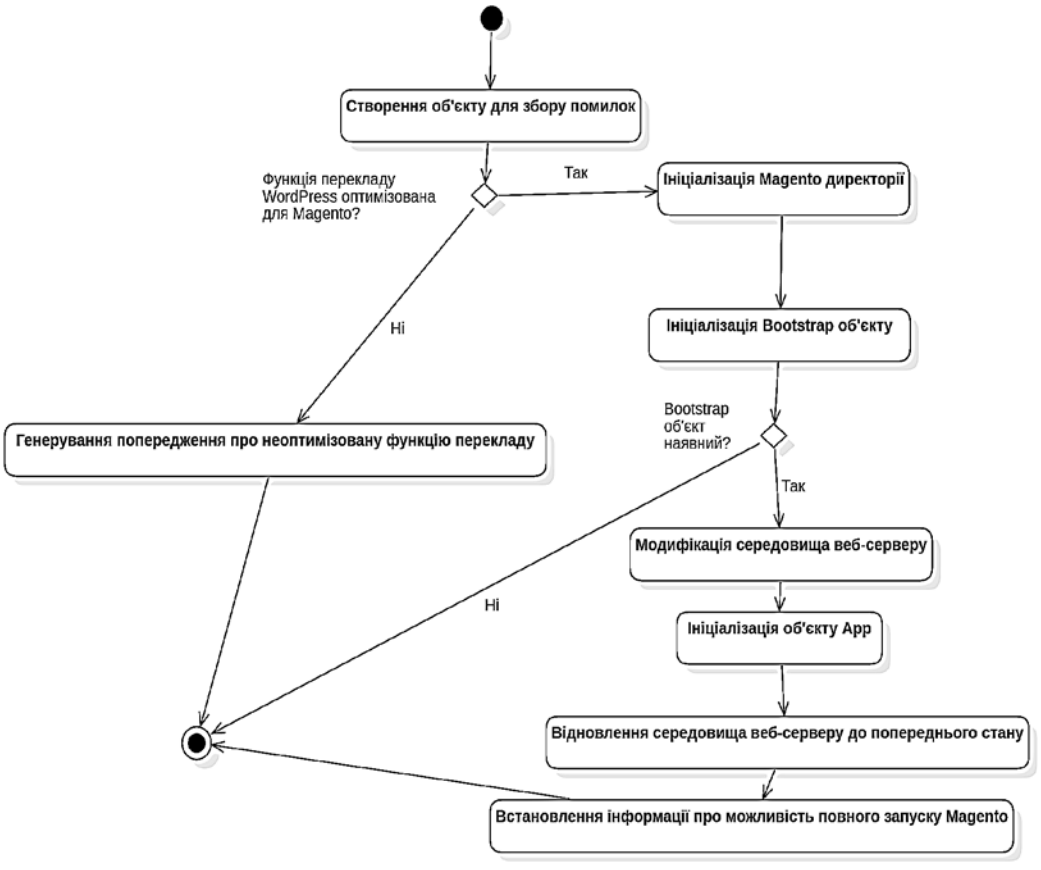


Рис. 1. Алгоритм ініціалізації Magento

також власні події, щоб забезпечити повноцінне функціонування інтеграції на всіх щаблях системи WordPress. Для кращого розуміння функціоналу подійно-орієнтовної системи плагіну розглянемо головну функцію підключення “m2i_plugin_init”, що додається в ролі дії на подію WordPress “setup_theme”. Додавання на подію “setup_theme” дозволяє забезпечити теми WordPress потрібним API для реалізації ручного функціоналу із інтеграції вмісту. Програмний код функції “m2i_plugin_init” наведено на рисунку 3.

На рисунках 4 та 5 наведено кінцевий результат використання розробленого плагіну для інтеграції Magento 2 у WordPress. Як видно із рисунків, блог на рушії WordPress має такий же вигляд, як і на Magento 2, тому кінцевий користувач не помітить різниці в користуванні. Те, що блог запущений на зовсім іншій системі, не завдасть незручностей для користувача, тобто відбудеться комфортний перехід сайту з однієї системи на іншу.

Висновки. Таким чином, було розглянуто алгоритми ініціалізації та повного запуску інтеграції Magento 2 у WordPress та основні класи системи. У роботі розроблено плагін, який використовується для забезпечення електронних магазинів на Magento 2 сучасним та багатофункціональним блогом на системі керування вмістом “WordPress” без додаткових витрат на команду розробників і дизайнерів із боку клієнта, окрім того, цей продукт розділений на дві версії повну (комерційна) та неповну (безкоштовна, “open source”). Протягом тривалого часу комерційна версія успішно

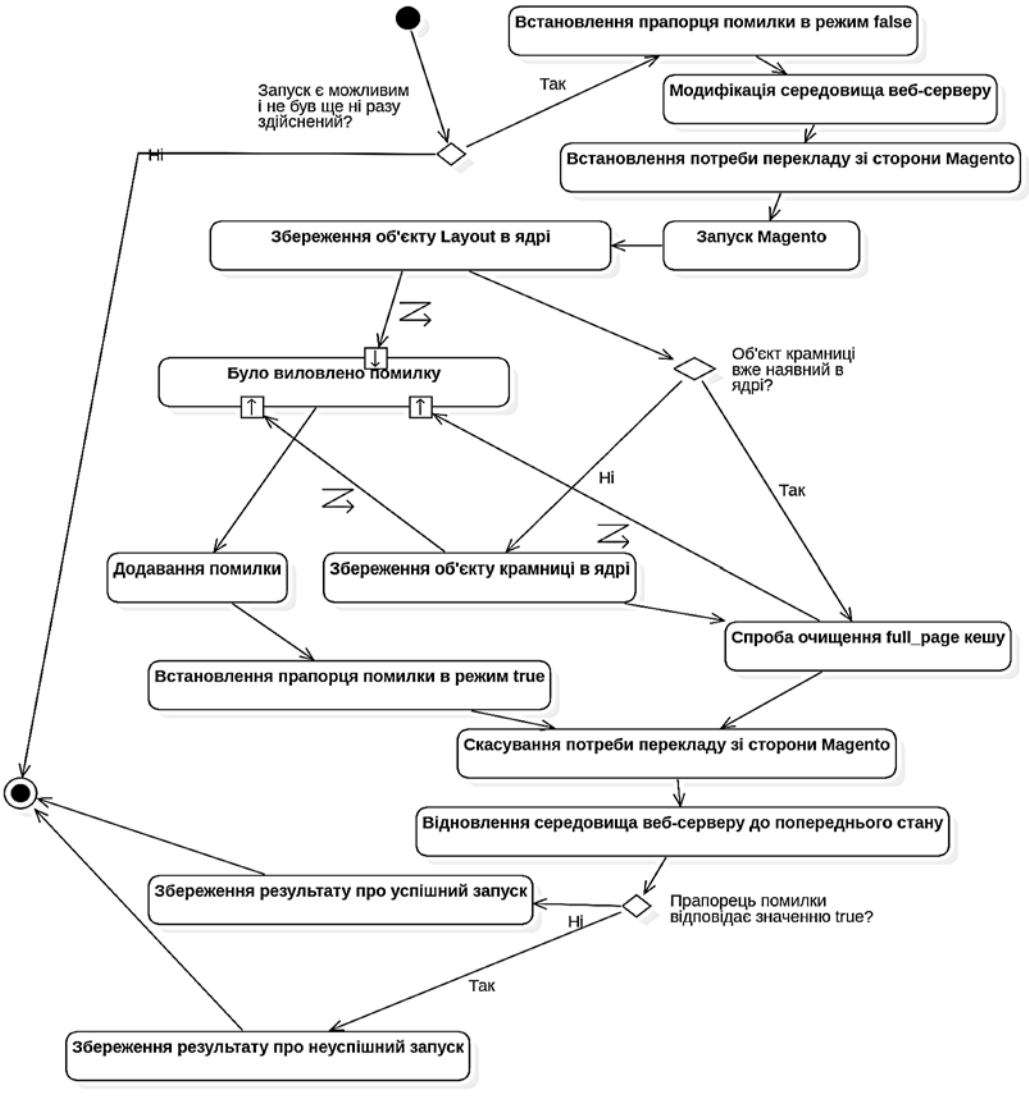
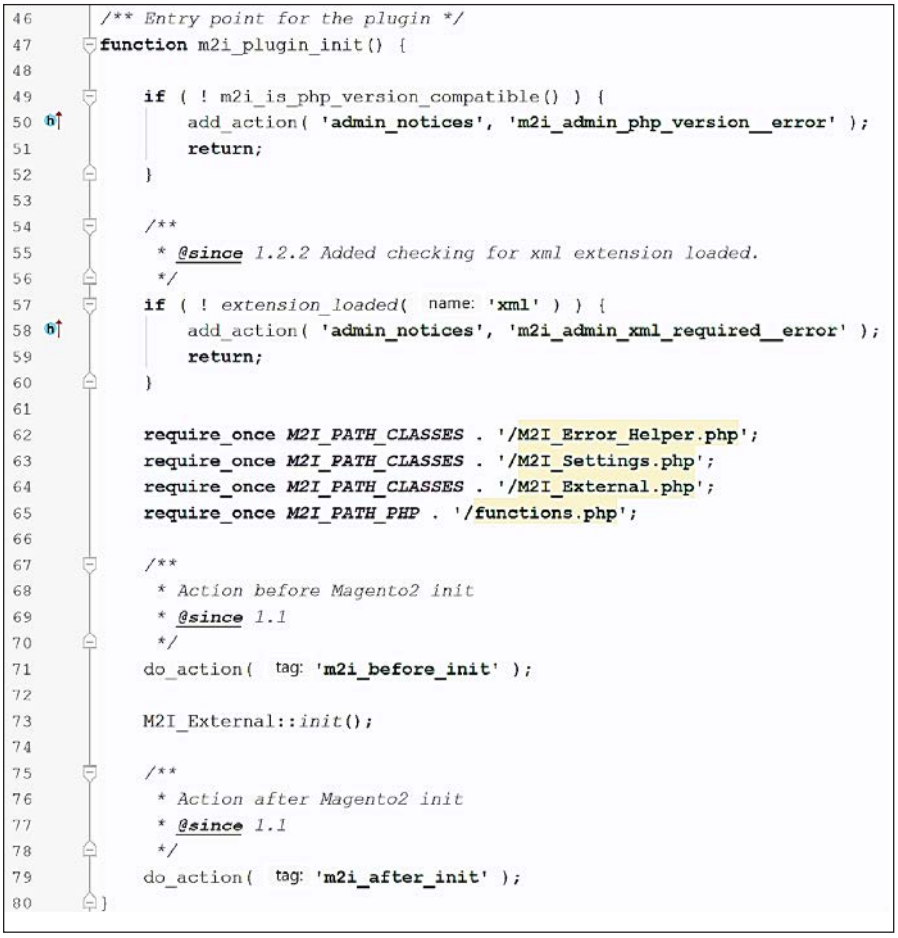


Рис. 2. Алгоритм повного запуску Magento



```php
/** Entry point for the plugin */
function m2i_plugin_init() {

    if ( ! m2i_is_php_version_compatible() ) {
        add_action( 'admin_notices', 'm2i_admin_php_version__error' );
        return;
    }

    /**
     * @since 1.2.2 Added checking for xml extension loaded.
     */
    if ( ! extension_loaded( name: 'xml' ) ) {
        add_action( 'admin_notices', 'm2i_admin_xml_required__error' );
        return;
    }

    require_once M2I_PATH_CLASSES . '/M2I_Error_Helper.php';
    require_once M2I_PATH_CLASSES . '/M2I_Settings.php';
    require_once M2I_PATH_CLASSES . '/M2I_External.php';
    require_once M2I_PATH_PHP . '/functions.php';

    /**
     * Action before Magento2 init
     * @since 1.1
     */
    do_action( tag: 'm2i_before_init' );

    M2I_External::init();

    /**
     * Action after Magento2 init
     * @since 1.1
     */
    do_action( tag: 'm2i_after_init' );
}
```

Рис. 3. Головна функція підключення “m2i_plugin_init”

Таблиця 2

Опис основних класів системи

Клас	Опис
M2I_Content	Відповідає за додавання дій на події WordPress, що знаходяться у верхній та нижній частині сайту, а також після підключення основного темплейту. Допомагає навісити функціонал по автоматичній інтеграції (перехоплення верхньої та нижньої частини WordPress) та інтеграції CSS/JS вмісту Magento. Повна публічність цього класу дозволятиме розробникам використовувати його у своїх темах та плагінах за потреби.
M2I_Mage_Autoloader	Успадковує клас \Magento\Framework\Code\Generator \Autoloader із системи Magento, що відповідає за автоматичне завантаження інших класів Magento. M2I_Mage_ Autoloader запобігає можливим конфліктам між автоматичними підвантажувачами PHP та логує усі фатальні помилки, що надходять від Magento, не дозволяючи їм зламати функціонування WordPress.
M2I_DOMDocument	Успадковує стандартний клас PHP, що відповідає за розбір XML/HTML вмісту, запобігає виникненню внутрішніх помилок, передає встановлене кодування вмісту відповідно до налаштування WordPress.
M2I_Editor_Button	Згідно з подійно-орієнтованою архітектурою збирає до купи усі пов’язані між собою дії по додаванню кнопки для швидкого вибору потрібного шорткоду. Містить у собі генерацію модального вікна для даних шорткодів, встановлює зв’язок із налаштуваннями віджетів для економії коду, тобто віджети використовуються для генерації потрібного шорткоду для подальшого вставлення до вмісту публікації цього шорткоду.
M2I_Widgets	Збирає до купи усі дії для подій у WordPress, що використовуються для віджетів. Запускає весь процес інтеграції із Magento для використання її у віджетах системи. В автоматичному режимі підвантажує та реєструє усі віджети з директорії widgets у системі інтеграції, оминаючи інтерфейси чи абстрактні класи для включення на реєстрацію.

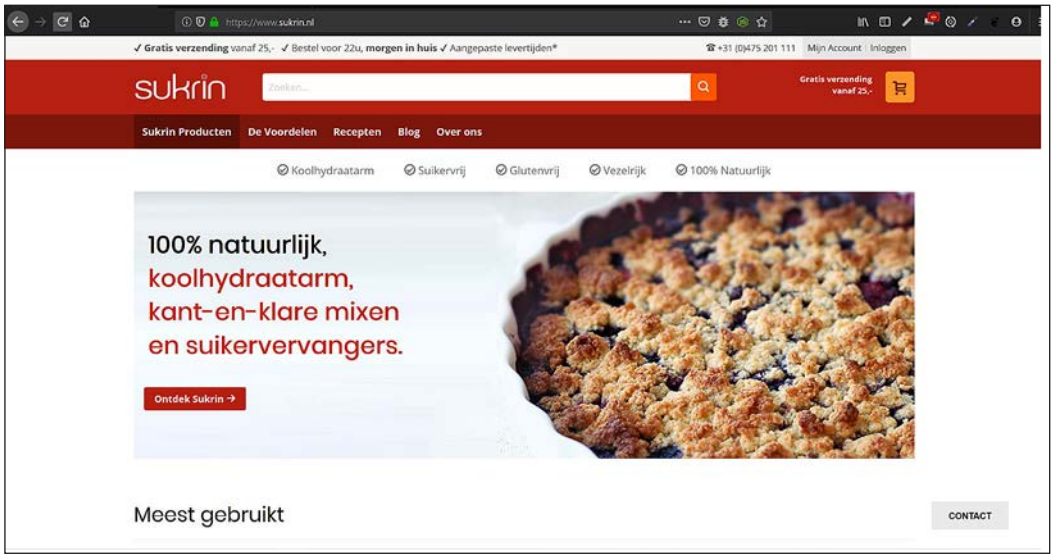


Рис. 4. Вигляд сайту в системі Magento 2

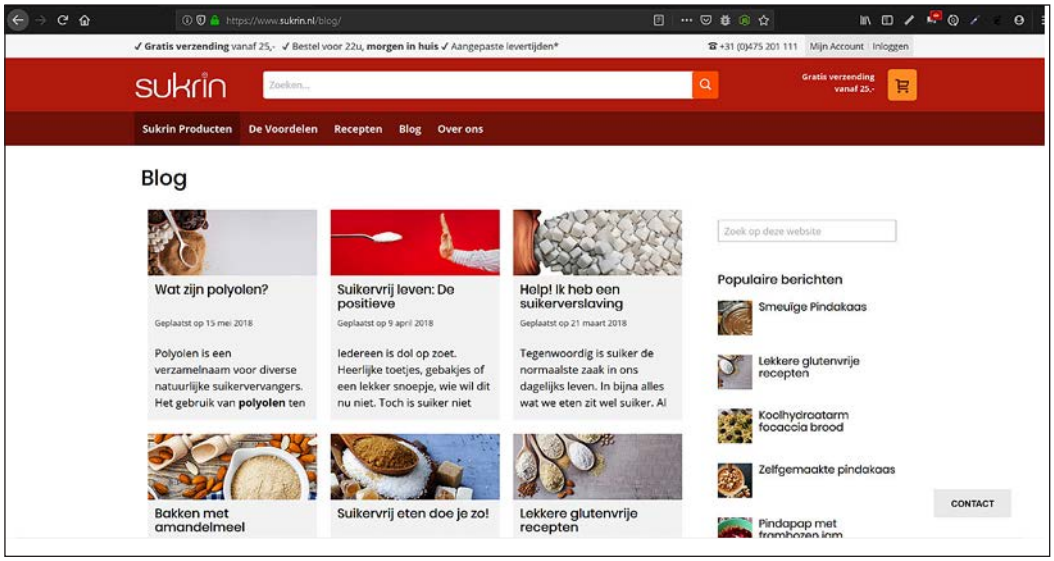


Рис. 5. Вигляд сайту в системі WordPress

продається на ринку WordPress плагінів та має більше 300 активних користувачів, що позначає неабиякий успіх продукту [11].

Реалізований плагін готовий до використання. Система може бути використана для забезпечення електронного магазину повнофункціональним блогом із мінімальними витратами на розробку або навпаки, забезпечення блогу електронним магазином. Надалі ця система може бути розширена шляхом впровадження нового функціоналу та розширення наявного API. Завдяки обраній архітектурі масштабування системи відбуватиметься значно швидше.

Список літератури.

1. News – One-third of the web! – WordPress.org. URL: https://wordpress.org/news/2019/03/one-third-of-the-web.
2. Magento WordPress Integration. URL: https://uk.wordpress.org/plugins/magento-wordpress-integration/.
3. Magento 2 compatibility. URL: https://wordpress.org/support/topic/magento-2-compatibility/.
4. Степура І. С. Особливості використання системи wordpress при реалізації сайту видання «грінченко-інформ». Електронне наукове фахове видання «Відкрите освітнє е-середовище сучасного університету». 2016. С. 240–243. URL: https://openedu.kubg.edu.ua/journal/index.php/openedu/article/view/50.
5. Мельник М.О. Організація захисту сайту, створеного на платформі WordPress за допомогою плагіна iThemes Security / М.О. Мельник, Р.В. Дудко, А.Д. Поліщук. Системи обробки інформації. 2017. № 2(148). С. 118–121. URL: https://doi.org/10.30748/soi.2017.148.22.

6. Язвинський Є. Шляхи підвищення конверсії в галузі e-commerce для вітчизняних підприємств. 2014. С. 224–231. URL: http://irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis.
7. Event Processing: Designing IT Systems for Agile Companies / W. Roy Schulte, K. Mani Chandy. McGraw-Hill Education; 1 edition, 2009. 257 c.
8. $_SERVER URL: https://www.php.net/manual/en/reserved.variables.server.php.
9. Magento 2 WordPress Integration. URL: https://uk.wordpress.org/plugins/m2wp/#installation.
10. Creational pattern. URL: https://en.wikipedia.org/wiki/Creational_pattern.
11. ZERO to ONE: NOTES ON STARTUPS, OR HOW TO BUILD THE FUTURE / Peter Thiel, Blake Masters – CROWN BUSINESS, New York, 2014 – 210 c.

Yachmenov Ya.O., Levkivskyi V.L., Kravchenko S.M., Grishkun E.O. **AUTOMATIZATION OF THE INTEGRATION PROCESS MAGENTO 2 TO WORDPRESS WITH THE HELP OF A PLUGIN**

In this work has been reviewed development of a new programming software as the plugin based on CMS "WordPress" (PHP/MySQL) and with the usage of Magento 2 API (PHP/MySQL) for making an automatic integration of Magento 2 content to WordPress and for resolving all possible conflicts between both systems. The system of integration Magento 2 to WordPress is building with the goal of maximum saving of WordPress possibilities, integration of Magento 2 design (blocks, containers, JS/CSS) to the blog engine, Magento 2 content (products, categories, etc.), reducing cost of additional resources for development and supporting blog for the requested e-commerce shop.

The done analysis of subject area has helped to determine the basic aspects of Magento 2 integration to WordPress. The review of existent alternatives and identifying of contradiction between available possibilities and needs has shown, that the list should include the next functions of the future system: full saving of WordPress functionality, an integration from Magento versions 2.0 – 2.3 in design and content, availability of handy GUI, availability of API for the further expanding and using with 3rd parties systems. The action oriented architecture has been selected for saving scalability of integration system. We've considered algorithms of the main system processes and analyzed features of integration engine functioning. We've given the detailed explanation of some realization aspects.

Developed plugin will be used for providing the modern and multifunctional blog based on the content management system "WordPress" to e-commerce shop based on Magento 2 without any additional costs for development team and designers from the client side, besides this, this product could be splitted up into two versions: the full one (paid) and limited one (free, public), which will make a good benefit too. This system might be used for providing the fully functional blog to the e-commerce store with minimal costs for development or vice-versa: providing e-commerce store to the blog.

Key words: *Magento, Magento 2, integration, WordPress, plugin, engine, PHP.*