УДК 519.2

Г. Ю. СИДОРЕНКО, М. М. МАЛЬКО, М. А. ЛЯШЕНКО

ВИКОРИСТАННЯ SERVERLESS ПІДХОДУ ДЛЯ СТВОРЕННЯ ВЕБ-ДОДАТКУ МОНІТОРІНГУ ТОВАРІВ

Використовуються сучасні підходи до організації веб-застосунків. Аналіз проблемної області проведено на прикладі оцінки моніторингу зміни вартості товарів в інтернет-магазинах з використанням технології Amazon. В роботі проведено огляд та аналіз сучасних методів організації веб-застосувань. Також була досліджена тематична область електронної торгівлі, проведено порівняльний аналіз. Технології розробки веб-додатків розвиваються великими темпами, створюються нові способи організації, технології, мови програмування. Основою всіх сучасних додатків є його архітектура. Від вибору правильної архітектури залежить не тільки швидкість і зручність доступу та розробки, а й успіх продукту. Було зроблено припущення про доцільність створення системи, що могла б швидко та зручно проводити моніторинг та нотифікування користувача. В роботі для вирішення задачі моніторингу було запропоновано використовувати архітектуру serverless. Серверна частина системи базується на безсерверній архітектурі, що має низку переваг перед іншими підходами. У роботі був обраний даний підхід не тільки для організації веб-сервісу, прив'язаного до призначеного для користувача інтерфейсу, але і для створення непостійної архітектури збору даних з різних джерел. Основними причинами вибору цього напрямку є можливість створення серверної архітектури на обмежений час без постійного використання і можливість стабільної роботи незалежно від навантаження. В результаті була побудована гнучка і зручна система, що має змогу слідкувати за вартістю товару без участі користувача та надсилати сповіщення про вдалу покупку. Система зберігає мінімальну кількість даних про користувача, чим досягається ще більша безпека користування. Під час моніторингу, якщо правило відстеження буде виконано, система відправить повідомлення користувачу за допомогою електронного листа.

Ключові слова: інтернет-магазин, безсерверна архітектура, моніторинг, вартість, алгоритм, on-line режим.

А. Ю. СИДОРЕНКО, М. Н. МАЛЬКО, Н. А. ЛЯШЕНКО

ИСПОЛЬЗОВАНИЕ SERVERLESS ПОДХОДА ДЛЯ СОЗДАНИЯ ВЕБ-ПРИЛОЖЕНИЯ МОНИТОРИНГА ТОВАРОВ

Используются современные подходы к организации веб-приложений. Анализ проблемной области проведен на примере оценки мониторинга изменения стоимости товаров в интернет-магазинах с использованием технологии Amazon. В работе проведен обзор и анализ современных методов организации веб-приложений. Также была исследована тематическая область электронной торговли, проведен сравнительный анализ. Технологии разработки веб-приложений развиваются большими темпами, создаются новые способы организации, технологии, языки программирования. Основой всех современных приложений является его архитектура. От выбора правильной архитектуры зависит не только скорость и удобство доступа разработки, но и успех программного продукта. Было сделано предположение о целесообразности создания системы, которая могла бы быстро и удобно проводить мониторинг и нотифицирование пользователя. В работе для задачи мониторинга предложено использовать архитектуру serverless. Серверная часть системы базируется на бессерверной архитектуре, которая имеет ряд преимуществ перед другими подходами. В работе был выбран данный подход не только для организации веб-сервиса, привязанного к пользовательскому интерфейсу, но и для создания непостоянной архитектуры сбора данных из различных источников. Основными причинами выбора этого направления является возможность создания серверной архитектуры на ограниченное время без постоянного использования и возможность стабильной работы независимо от нагрузки. В результате была построена гибкая и удобная система, которая имеет возможность следить за стоимостью товара без участия пользователя и отправлять уведомление о достижении заданного критерия к стоимости товаров. Система сохраняет минимальное количество данных о пользователе, чем достигается большая безопасность использования. Во время мониторинга, если правило отслеживания будет выполнено, система отправит сообщение пользователю с помощью электронной почты.

Ключевые слова: интернет-магазин, бессерверная архитектура, мониторинг, стоимость, алгоритм, on-line режим.

G. Y. SYDORENKO, M. M. MALKO, M. A. LYASHENKO

USING THE SERVERLESS APPROACH FOR THE CREATING WEB-APPLICATION OF THE MONITORING OF PRODUCTS

There have been using modern approaches to the organization of web applications. The analysis of the problem area was carried out using the example of monitoring the change in the value of goods in online stores using Amazon technology. The article has been reviewing and analyzing of modern methods of organizing web applications. The thematic area of the electronic commerce was also investigated, and a comparative analysis was carried out. Technologies for developing web-applications are developing at a rapid pace, new methods of organization, technologies, programming languages are being created. The basis of all modern applications is its architecture. Not only the speed and convenience of development access, but also the success of the software product, depends on choosing the right architecture. It was suggested that it would be advisable to create a system that could quickly and conveniently monitor and notify a user. In the work for the monitoring task, it was suggested to use the serverless architecture. The server part of the system is based on a serverless architecture, which has a number of advantages over other approaches. In this article, this approach was chosen not only for organizing a web service tied to the user interface, but also for creating a non-persistent architecture for collecting data from various sources. The main reasons for choosing this direction is the possibility of creating a server architecture for a limited time without constant use and the possibility of stable operation regardless of the load. As a result, a flexible and convenient system was built that has the ability to monitor the value of the product without user intervention and send a notification of the achievement of a given criterion to the value of the goods. The system saves a minimum amount of user data, thus achieving greater safety of use. During monitoring, if the tracking rule is met, the system will send a message to the user using e-mail.

Keywords: online store, serverless architecture, monitoring, cost, algorithm, on-line mode

Вступ. У сучасному світі вплив технологій на всі аспекти життя є дуже високим. Не зважаючи на велику кількість online-магазинів та достатній проміжок часу для їх розвитку, робота з таким магазином не проста. Однією із причин є дуже велика швидкість змін вартості та кількості товарів. Саме тому поліпшення швидкості і якості користування має неабияку роль [1].

Інтернет-магазин – це форма електронної комерції, яка дозволяє споживачам купувати товари або

послуги безпосередньо від продавця через Інтернет за допомогою веб-браузера [2]. Споживачі знаходять продукт, що представляє інтерес, відвідавши безпосередньо веб-сайт продавця або шукаючи серед альтернативних постачальників, використовуючи пошукову систему покупок, яка відображає наявність та вартість одного і того ж продукту у різних електронно-роздрібних продавців. Типовий інтернет-магазин дає змогу клієнту переглядати спектр продуктів та послуг фірми, переглядати фотографії або зображення продуктів, а також інформацію про технічні характеристики, особливості та ціни. Магазини, зазвичай, дозволяють покупцям використовувати функції пошуку конкретних моделей, брендів або предметів. Для фізичних продуктів (наприклад, книжок або одягу) продавець відправляє продукти клієнту, а для цифрових продуктів, таких як цифрові аудіофайли пісень або програмне забезпечення, продавець зазвичай надсилає файл клієнту через Інтернет.

Однією з проблем, які можна не побачити відразу, є проблема зміни вартості продукту. Ціни на товари формуються неоднорідно и не мають прямого чи єдиного чинника впливу. Спершу, можна сказати, що основою для вартості продукту є фактична ціна виробництва та додаткова вартість, що є прибутком продавця, але це не так. Покупцю важно почувати себе у виграшній ситуації, бо ціни постійно змінюються. У сучасному світі є велика кількість знижок, що можуть мати різні причини [3]:

1) акційні знижки на товари певного магазина;
2) знижки на сезоні товари;
3) специфічні знижки за даною категорією;
4) персональні знижки;
5) знижки на товари з пошкодженою упаковкою.

Тому дослідження методів моніторингу вартості продуктів та побудова сучасної зручної системи для поліпшення роботи користувачів має великий інтерес.

Не зважаючи на велику кількість різних знижок, покупець може знати про період дії знижки, проте знижка може бути і в деякий невеликий період. Також не зрозуміло, коли знижка почнеться і скільки годин вона буде діяти. До того ж покупці великих інтернет-магазинів можуть знаходитися у різних частинах світу. Все це є причиною того, що велика кількість знижок може пройти повз користувача. Саме тому важливим є процес моніторингу ціни на товари, які користувач хоче придбати. З іншого боку, знаходження користувача на сайті постійно, не є дуже зручним процесом. Саме тому, не менш важливим є надання нотифікацій користувачу з використанням електронних листів або інших видів нотифікацій.

Метою даної роботи є дослідження методів моніторингу вартості продукції та побудова сучасної зручної системи моніторингу на базі безсерверної архітектури.

Сучасні способи організації архітектури. Технології розробки веб-додатків розвиваються високими темпами, створюються нові способи організації, технології, мови програмування [4]. Однак, основою всіх сучасних додатків є архітектура. Від вибору правильної архітектури залежить не тільки швидкість і зручність доступу та розробки, а й успіх продукту.

Проведемо ознайомлення з сучасними способами організації архітектури веб-додатків, їх порівняння і аргументований вибір архітектури для розробки програмного продукту. Спочатку, найбільш використовуваним підходом була монолітна архітектура, що дозволяє будувати величезні додатки, які складаються з безліч функціональних частин в рамках однієї великої структури. Одним з основних переваг цього підходу є незалежність від будь-яких сторонніх ресурсів на користь організації всього продукту в рамках однієї програми [5].

На зміну монолітної архітектури прийшов мікросервісний підхід. У ньому використовується абсолютно інша стратегія організації. В даній архітектурі широко застосовуються різні сторонні сервіси, програма містить кілька невеликих компонент, кожен з яких є окремим веб-додатком. Ці компоненти взаємопов'язані між собою. Кожному компоненту даної архітектури відповідає певна роль у функціонуванні програми. Основною перевагою даного підходу є зручність і гнучкість додатків, що дозволяє вдаватися до масштабування окремих компонент [4, 5]. На даний момент даних підхід є найбільш популярним.

Serverless (безсерверна архітектура) є підходом до організації архітектури веб-додатків, що активно набирає популярність (рис. 1). Взявши всі переваги мікросервісного підходу, безсерверний підхід має відсутність будь-якої апаратної частини в організації веб-додатків [6].

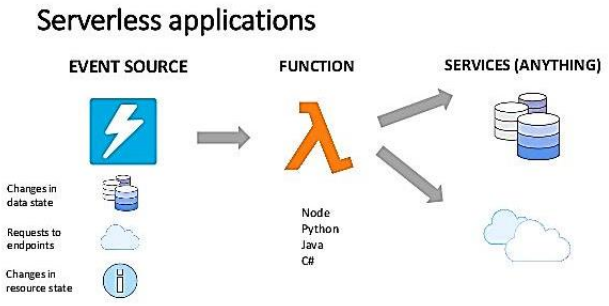

Рис. 1. Схема безсерверної архітектури

Serverless має як ряд переваг, так і деякі недоліки. Перевагами даного підходу є:

1) відсутність апаратної частини – серверів;
2) відсутність прямого контакту і адміністрування серверної частини;
3) практично безмежне горизонтальне масштабування додатку.

До недоліків даного підходу можна віднести:

1) відсутність чіткого контролю за виконанням програми;
2) відсутність цілісності додатка;
3) холодний старт може займати кілька секунд, що негативно сприймається користувачем [7].

В цілому даний підхід є досить придатним для вирішення завдань певного роду в рамках великого

додатка або для початкових етапів розробки. Особливий акцент в даному випадку ставиться на незалежність від серверної інфраструктури і безмежний приріст продуктивності в автоматичному режимі.

Одночасно з безсерверною архітектурою вводиться поняття FaaS (Function-as-a-Service, функція як сервіс), що характеризується підходом до розбиття сервісів на найменші функціональні частини спільно з відокремленням цих частин в окремі функціональні одиниці. Наприклад, веб-сервіс розбивається на окремі шляхи і для кожного такого шляху створюється окрема функція, яка викликається при кожному зверненні до нього. Також, безсерверні системи набагато простіше підтримувати, більш того, будь-яка оптимізація продуктивності FaaS-додатку автоматично знижує операційні витрати [8].

У роботі був обраний даний підхід не тільки для організації веб-сервера прив'язаного до призначеного для користувача інтерфейсу, але й для створення непостійної архітектури збору даних з різних джерел. Основними причинами вибору цього напрямку є можливість створення серверної архітектури на обмежений час без постійного використання і можливість стабільної роботи незалежно від навантаження.

Огляд моделі веб-застосунку. Беручи до уваги усі вищезазначені дослідження та аналіз сучасних підходів, було вирішено створити програмну модель, яка б вирішувала проблеми моніторингу та мала б за основу безсерверну архітектуру. На рис. 1 зображена повна модель серверної частини веб-застосунку.

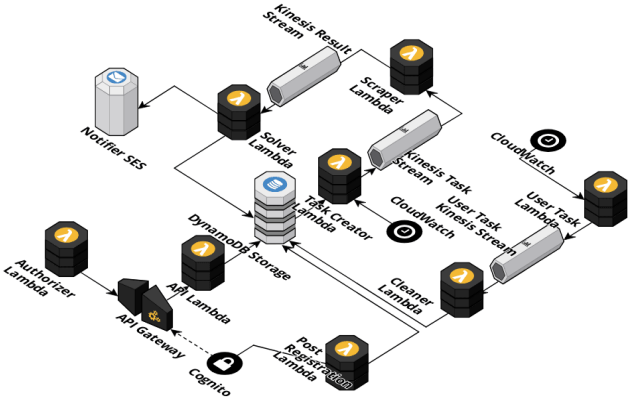

Рис. 1. Діаграма серверної частини веб-застосунку

На рис. 1 можна побачити, що веб-застосунок можна уявно поділити на три окремі частини: частину веб-сервісу, частину слідкування за ціною продукції, допоміжну частину для моніторингу за використанням користувачем доцільної кількості ресурсів в залежності від його типу аккаунта.

На рис. 2 представлено частину веб-сервісу. Веб-сервіс відіграє дуже важливу роль у застосунку. Він використовується для зв'язку між користувачем та системою моніторингу. Вся робота користувача з системою проводиться через веб-сервіс. На рис. 2 можна спостерігати, що ця частина також має декілька складових: API Gateway, Authorizer Lambda, API Lambda, Cognito, Post Registration Lambda та DynamoDB Storage [9, 10].

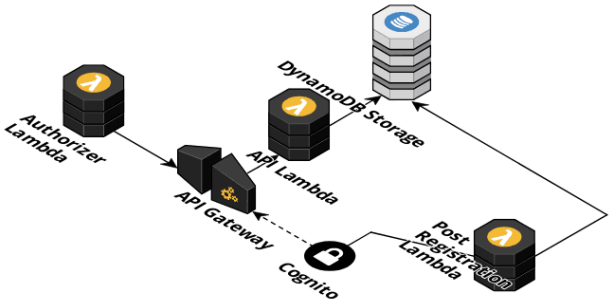

Рис.2. Діаграма частини веб-сервісу

Другою частиною веб-застосунку є частина моніторингу вартості продукції. Схема цієї частини представлена на рис. 3. Частина моніторингу є найбільш важливою у роботі застосунку. Вона виконує автоматичну роботу без участі користувача. Таким чином досягається можливість позбавлення користувача необхідності постійно слідкувати за вартістю продукції та, навіть, проводити час за комп'ютером. Це досягається за допомогою об'єднання багатьох сервісів Amazon [11, 12]. Дивлячись на діаграму можна побачити, що частина моніторингу також має декілька складових: DynamoDB Storage, CloudWatch, Task Creator Lambda, Kinesis Task Stream, Scraper Lambda, Kinesis Result Stream, Solver Lambda та Notifier SES.

Іншою складовою є CloudWatch. Amazon CloudWatch – це сервіс моніторингу хмарних ресурсів AWS і додатків, які запускаються за їх допомогою. Amazon CloudWatch можна використовувати для збору і відстеження метрик, накопичення та аналізу файлів журналів, створення попереджень, а також автоматичного реагування на зміни ресурсів AWS [13]. Також цей сервіс можна застосовувати для отримання зведеної інформації про систему, що включає в себе інформацію про ресурси, які використовуються, а також продуктивність додатків і загальний стан системи. Ці дані застосовуються для оперативного реагування та забезпечення стабільної роботи додатків.

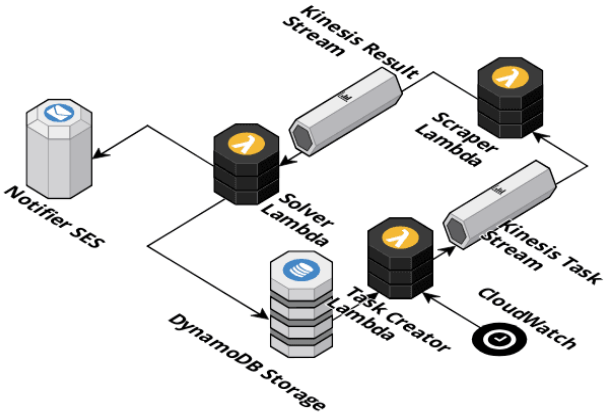

Рис.3. Діаграма частини моніторингу

Використана система сервісів дозволяє кожної години проводити моніторинг будь-якої кількості товарів та проводити нотифікування користувачів про зміни товарів.

Останньою частиною є допоміжна частина для слідкування за використанням користувачем доцільної кількості ресурсів в залежності від його типу аккаунту. У сервісі присутні 5 видів аккаунтів, що дозволяють користувачу виконувати моніторинг різної кількості продукції у місяць.

На рис. 4 представлена діаграма допоміжної частини. З огляду на те, що кількість товарів для моніторингу обмежена, кожний місяць допоміжна частина користувачів виконує роль очищення бази даних від записів про кількість товарів, що була використана. Ця частина має такі складові: DynamoDB Storage, Cleaner Lambda, User Task Kinesis Stream, User Task Lambda та CloudWatch. Кожного місяця CloudWatch створює подію, яка визначає момент, коли треба провести очищення бази.

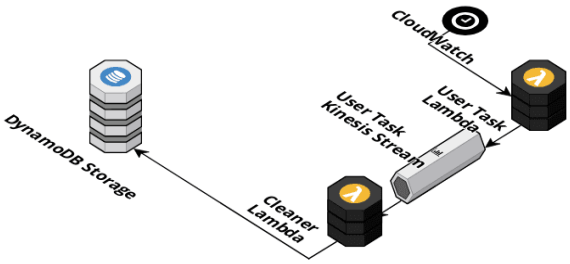

Рис.4. Діаграма допоміжної частини веб-сервісу

User Task Lambda створює низку завдань для виконання та вкладає їх у User Task Kinesis Lambda. Cleaner Lambda реагує на наявність завдань та проводить зазначені вище дії [8, 9]. Візуальна частина виконана за допомогою фреймворка React та системи станів Redux [12]. Як хостинг та для зменшення витрат на основі декількох сервісів Amazon була створена система, схему якої можна побачити на рис. 5.

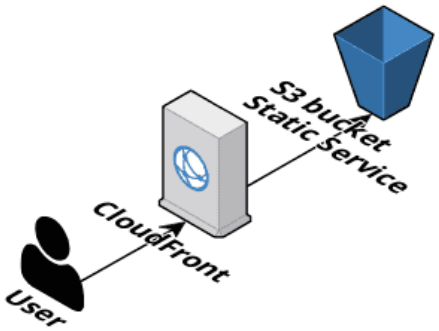

Рис.5. Діаграма візуального сервісу

З рис. 5 можна зробити висновок, що візуальна частина складається з двох складових: CloudFront та S3 Bucket Static Service. В роботі S3 використовується саме як сервіс статичних даних разом з CloudFront, проте також використовуються усі вищезазначені переваги.

Основні результати роботи. Спираючись на принципах та архітектурі, описаних вище, було створено веб-застосунок для моніторингу вартості продукції на баз архітектури serverless. На рис. 6 зображена загальна структура веб-застосунку.

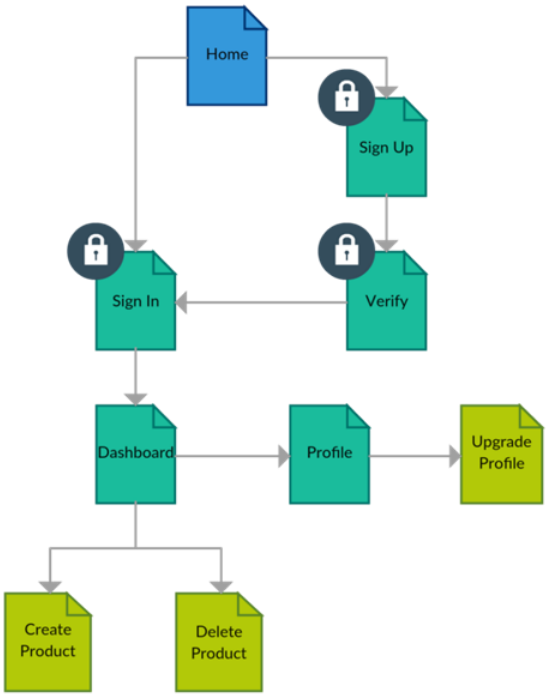

Рис. 6. Загальна структура веб-застосунку

Після авторизації користувача виконується запит до Amazon Cognito, який збереже введені дані до бази, якщо вони відповідають вимогам безпеки. Після збереження даних Cognito створює системний лист до користувача з кодом верифікації поштової адреси та наміру реєстрації у той час, коли веб-застосунок перенесе користувача на сторінку верифікації та запропонує ввести код. Після введення коду виконується спеціальний запит до Amazon Cognito з метою верифікації та активації користувача за наявності вірного коду. Після успішної верифікації веб-застосунок перенесе користувача на сторінку автентифікації.

У випадку правильно введених даних користувача система перенесе його на основну сторінку зі списком товарів, що вже є зареєстрованими для моніторингу (приклад, рис. 7).

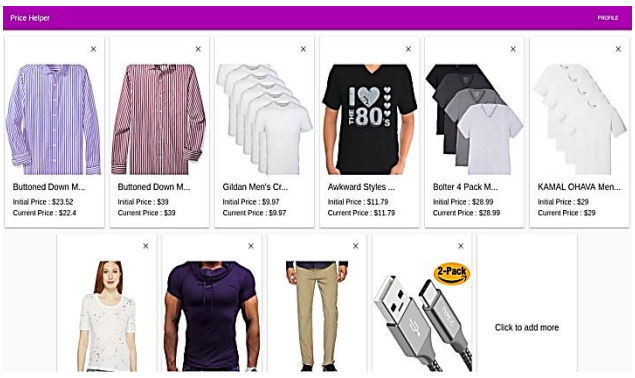

Рис. 7. Приклад основної сторінки користувача

Товари розподіляються рівномірно по сторінці. У веб-застосунку присутня можливість видалити непотрібний товар або додати новий товар. Додавання нового товару виконується у два кроки. На першому кроці користувач має ввести посилання на товар, що він бажає додати. На другому кроці система виконає запит за товаром та запропонує підтвердити чи треба додати товар до системи. Також на другому кроці система пропонує користувачу додати параметри моніторингу, який він бажає. Система дає можливість обрати тип зміни ціни: підвищення або зниження, та величину у відсотках, на яку повинна змінитися ціна товару. Під час моніторингу, якщо правило відстеження буде виконано, система відправить повідомлення до користувача за допомогою електронного листа.

Висновки. У даній роботі було проведено огляд та аналіз сучасних методів організації веб-застосувань. Також було досліджено тематичну область електронної торгівлі і проведено порівняльний аналіз. Було зроблено припущення про доцільність створення системи, що могла б швидко та зручно проводити моніторинг та нотифікування користувача.

Як результат, була змодельована система, що дає можливість користувачу передати мінімум інформації про продукт та водночас отримувати детальну інформацію про моніторинг вартості обраної продукції.

Список літератури

1. Düüna K. *Secure your Node.js web application*. The Pragmatic Programmers, LLC. 2016. 222 p.
2. OWASP Top 10. *Application Security Risk*. URL: https://www.owasp.org/index.php/Top_10-2017_Top_10 (дата звернення 25.03.2018).
3. Fowler M. *Serverless architectures* URL: https://martinfowler.com/articles/serverless.html (дата звернення 05.04.2018).
4. Притула И., Агапов Д. *OWASP Top 10: практический взгляд на безопасность веб приложений*. URL: https://habrahabr.ru/company/simplepay/blog/28499 (дата звернення 25.05.2018).
5. Kimhi T. *The state of the serverless ecosystem.* URL: https://medium.com/@talkimhi/the-state-of-the-serverless-ecosystem (дата звернення 25.05.2018).
6. Jamieson F. *Losing the server? Everybody is talking about serverless architecture*. URL: https://bcs.org/content/conWebDoc/58491 (дата звернення 13.04.2018).
7. *AWS documentation*. URL: https://aws.amazon.com/documentation/ (дата звернення 03.02.2018).
8. Miller Ron. *Amazon launches lambda, an event-driven compute service*. URL: https://techcrunch.com/2014/11/13/amazon-launches-lambda-an-event-driven-compute-service/ (дата звернення 17.03.2018).
9. Miller Ron. *AWS Lambda makes serverless applications a reality* URL: https://techcrunch.com/2015/11/24/aws-lambda-makes-serverless-applications-a-reality/ (дата звернення 07.05.2018).
10. Haviv Yaron. *CNCF Serverless whitepaper*. URL: https://github.com/cncf/wg-serverless/tree/master/whitepaper/ (дата звернення 07.05.2018).
11. Sbarski P. *Serverles architectures on AWS*. Manning Publications Co. 2017. 500 p.
12. *Redux, API docs.* URL: https://redux.js.org (дата звернення 11.05.2018).
13. *Claudia.JS, API docs*. URL: https://claudiajs.com/ (дата звернення 06.04.2018).

References (transliterated)

1. Düüna K. *Secure your Node.js web application*. The Pragmatic Programmers, LLC, 2016. 222 p.
2. OWASP Top 10. *Application Security Risk.* Available at: https://www.owasp.org/index.php/Top_10-2017_Top_10 (accessed 25.03.2018).
3. Fowler M. *Serverless architectures* Available at: https://martinfowler.com/articles/serverless.html (accessed 05.04.2018).
4. Pritula I., Agapov D. *OWASP Top 10: prakticheskiy vzglyad na besopasnost' web prilogeniy* [OWASP Top 10: a practical look at the security of web applications]. Available at: https://habrahabr.ru/company/simplepay/blog/28499 (accessed 25.05.2018).
5. Kimhi T. *The state of the serverless ecosystem.* Available at: https://medium.com/@talkimhi/the-state-of-the-serverless-ecosystem (accessed 25.05.2018).
6. Jamieson F. *Losing the server? Everybody is talking about serverless architecture*. Available at: https://bcs.org/content/conWebDoc/58491 (accessed 13.04.2018).
7. *AWS documentation.* Available at: https://aws.amazon.com/documentation/ (accessed 03.02.2018).
8. Miller Ron. *Amazon launches lambda, an event-driven compute service.* Available at: https://techcrunch.com/2014/11/13/amazon-launches-lambda-an-event-driven-compute-service/ (accessed 17.03.2018).
9. Miller Ron. *AWS Lambda makes serverless applications a reality* Available at: https://techcrunch.com/2015/11/24/aws-lambda-makes-serverless-applications-a-reality/ (accessed 07.05.2018).
10. Haviv Yaron. *CNCF Serverless whitepaper*. Available at: https://github.com/cncf/wg-serverless/tree/master/whitepaper/ (accessed 07.05.2018).
11. Sbarski P. *Serverles architectures on AWS*. Manning Publications Co, 2017. 500 p.
12. *Redux, API docs.* Available at: https://redux.js.org (accessed 11.05.2018).
13. *Claudia.JS, API docs*. Available at: https://claudiajs.com/ (accessed 06.04.2018).

Надійшла (received) 01.06.2018

Відомості про авторів / Сведения об авторах / About the Authors

Сидоренко Ганна Юріївна (***Сидоренко Анна Юрьевна***, ***Sydorenko Ganna Yuriivna***) – кандидат технічних наук, доцент, Національний технічний університет «Харківський політехнічний інститут», доцент кафедри системного аналізу та інформаційно-аналітичних технологій; м. Харків, Україна; ORCID: https://orcid.org/0000-0002-0761-2793; e-mail: annsydorenko01@gmail.com

Малько Максим Миколайович (***Малько Максим Николаевич***, ***Malko Maksym Mykolayovych***) – кандидат технічних наук, доцент, Національний технічний університет «Харківський політехнічний інститут», доцент кафедри системного аналізу та інформаційно-аналітичних технологій; м. Харків, Україна; ORCID: https://orcid.org/0000-0002-0125-2141; e-mail: maxim_malko@hotmail.com

Ляшенко Микита Андрійович (***Ляшенко Никита Андреевич***, ***Lyashenko Mykyta Andriyovych***) – Національний технічний університет «Харківський політехнічний інститут», студент; м. Харків, Україна; ORCID: https://orcid.org/0000-0002-1825-0088; e-mail: neiket@hotmail.com