УДК 004.773

А. В. Миргородський[1]
М. І. Третяк[1]

ВИКОРИСТАННЯ ХМАРНИХ ТЕХНОЛОГІЙ ДЛЯ АВТОМАТИЗАЦІЇ ПОТОКІВ ДАНИХ ТА РЕАЛІЗАЦІЇ ПЕРСОНАЛЬНИХ ОПОВІЩЕНЬ

[1]Вінницький національний технічний університет

Анотація

Досліджено варіанти реалізації системи для автоматизації потоків даних та створення на їх основі персональних оповіщень. Описано переваги використання хмарних технологій для даної задачі.

Ключові слова: IT, хмарні сервіси, хмарні платформи, безсерверні розрахунки, API, webhook.

Abstract

The options for implementing a system to automate data streams and create personal alerts based on them were investigated. The advantages of using cloud technologies for this task are described.

Keywords: IT, cloud services, cloud platforms, serverless computing, API, webhook.

Вступ

Щоденний технічний прогрес людства все швидше збільшує кількість доступної інформації, її складність та можливі точки доступу до неї. Це ускладнює не тільки процеси управління та збереження даних, а й погіршує якість та швидкість їх засвоєння іншими особами.

Сучасна людина підтримує швидкий темп життя та не може собі дозволити витрачати занадто багато часу на щоденний пошук та збір інформації з десятків месенджерів, соціальних мереж, інформаційних сайтів та бізнес-додатків. Даний процес потребує автоматизації.

Метою дослідження є дослідження варіантів реалізації системи для автоматизації потоків даних та створення на їх основі персональних оповіщень за допомогою сервісів безсерверних обчислень на хмарних платформах.

Об'єктом дослідження є процес автоматизації потоків даних та створення на їх основі персональних оповіщень.

Предмет дослідження – це система для автоматизації потоків даних та створення на їх основі персональних оповіщень.

Основна частина

Велика кількість інтернет-ресурсів надає користувачам доступ до інформації не тільки за допомогою сайтів та власних додатків, а й з використанням API. Прикладний програмний інтерфейс (API) дозволяє спростити отримання даних та використання деякого функціоналу ресурсу задля власних потреб [1]. Типовими є наступні використання:

- маніпулювання документами (Google Drive API, Dropbox API та інші);
- відображення контенту з інших ресурсів на власному сайті (Twitter API, YouTube API);
- авторизація, керування рекламою, обробка платежів (Facebook APIs);
- надсилання повідомлень, розширення функціоналу за допомогою додатків або ботів (Telegram APIs);
- оповіщення при нових подіях (Gmail API).

API для інтернет-сервісів зазвичай побудовані на основі клієнт-серверної архітектури з використанням можливостей протоколу HTTP, а найбільш популярним форматом для передачі даних є JSON, рідше – XML. Це дозволяє легко використовувати сторонні Web API при розробці додатків будь-якої складності.

Webhook – це ще один спосіб розширення функціоналу інтернет-ресурсу, який може доповнювати

можливості звичайного API. Якщо при використання API ми формуємо запит до серверу та потім обробляємо його відповідь, то попередньо налаштований webhook дозволяє без додаткового запиту отримати повідомлення про виникнення певної події. Ця різниця в особливостях роботи показана на рисунку 1.

Рисунок 1 – Порівняння використання API та Webhook при очікуванні певної події на сервері

Цей набір інструментів дозволяє автоматизувати процес отримання та обробки даних. Zapier є одним з найбільш популярних інструментів для цього. Цей продукт дозволяє отримувати дані або події з інших ресурсів, після чого фільтрувати їх, надсилати або використовувати при викликах інших API. При цьому, сам процес налаштування таких послідовностей має вигляд конструктора.

Використання сучасних хмарних технологій дозволяє просто й швидко реалізувати власний варіант даного інструменту, який буде мати ряд переваг: нижча вартість, простота підтримки, широкі можливості налаштування для будь-яких задач. Крім того, найбільш популярні хмарні платформи мають власні сервіси для безсерверних розрахунків: AWS Lambda, GCP Cloud Functions, GCP App Engine, Azure Functions та інші.

Безсерверні розрахунки (serverless computing) дозволяють динамічно контролювати використання ресурсів хмарної платформи. Очевидними перевагами є відсутність необхідності обслуговувати сервери, легка масштабованість, проста й вигідна модель оплати [2]. Легка масштабованість та дешевизна досягається за допомогою можливості розділити код додатку на невеликі окремі частини з чітко визначеними можливими входами та виходами. Це дозволяє досягти високого степеня багаторазового використання, а атомарність частин коду допомагає легко відслідковувати вузькі місця, що лише збільшує швидкість виконання та зменшує витрати.

Для автоматизації потоків даних скористаємося AWS Lambda. AWS Lambda – це сервіс безсерверних розрахунків від Amazon. На 2021 рік хмарні сервіси AWS залишаються найбільш популярними в світі [3]. Розглянемо наступний сценарій: викладач створив на Google Drive теку з доступом по посиланню та хоче при завантаженні студентом нового файлу отримувати повідомлення в Telegram. Приклад архітектури з використанням AWS наведено на рисунку 2.

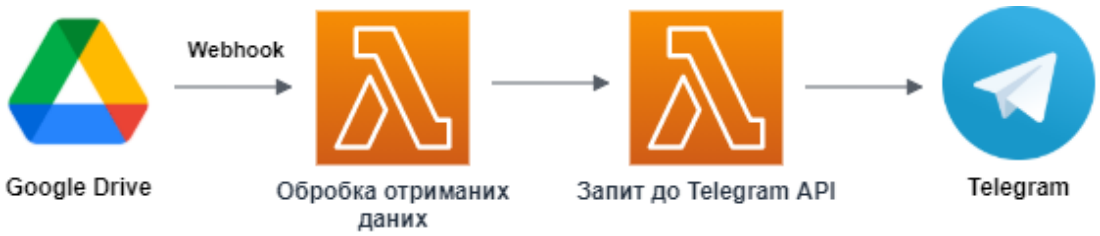

Рисунок 2 – Приклад реалізації оповіщення при додаванні нового файлу

Для реалізації даної послідовності необхідно створити webhook на обрану теку в Google Drive, який буде надсилати дані про зміну стану файлів до функції, що створена в AWS Lambda. Логічно буде

розділити цю частину обробки на дві складові: окрему функцію для читання даних, їх фільтрації та інших маніпуляцій, після чого скористатися іншою функцією, яка буде налаштована для гнучкої роботи з Telegram API (наприклад, буде надсилати повідомлення від імені попередньо створеного бота).

Використання інших хмарних сервісів дозволяє ще більше розширити можливості кінцевого користувача. Розглянемо наступний варіант: існує популярний сайт з великою кількістю технічних статей, але людина не хоче витрачати час на довгу фільтрацію їх списку. При виході нових статей, що потенційно можуть бути цікавими для неї, необхідно автоматично надсилати оповіщення через Telegram-бота, а також надати можливість оцінки статті через бота, щоб покращити фільтрацію в майбутньому. Таким чином, приклад архітектури для даної задачі наведено на рисунку 3.

Рисунок 3 – Приклад реалізації з використанням машинного навчання

Дана архітектура передбачає використання AWS SageMaker – сервісу для машинного навчання. При отриманні нових даних з сайту необхідно не відразу надіслати їх кінцевому користувачу, а підготувати й передати в даний сервіс, який буде вирішувати – підходить конкретна стаття користувачу чи ні. В якості сховища даних додатково використовується AWS S3. Також додається додаткова функція для обробки оцінки користувача, адже в Telegram API є можливість налаштувати webhook на нові дії користувача з ботом, тому необхідно отримати ці дані, підготувати їх і також надіслати в AWS SageMaker для оновлення моделі.

Висновок

Таким чином, було розглянуто декілька прикладів використання хмарних сервісів для реалізації системи автоматизації потоків даних та створення на їх основі персональних оповіщень. Було описано архітектуру рішень на прикладі сервісів AWS, Telegram API та Google Drive API. Коротко описано використані технології та їх можливості. Було розглянуто переваги використання безсерверних розрахунків для досягнення мети дослідження.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Introduction to web APIs [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction#what_are_apis.

2. Why use serverless computing? | Pros and cons of serverless [Електронний ресурс] – Режим доступу до ресурсу: https://www.cloudflare.com/learning/serverless/why-use-serverless/.
3. AWS vs Azure vs Google Cloud Market Share 2021: What the Latest Data Shows [Електронний ресурс] – Режим доступу до ресурсу: https://www.parkmycloud.com/blog/aws-vs-azure-vs-google-cloud-market-share/.

Миргородський Андрій Вікторович **–** студент групи 3ПІ-18б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця.

Третяк Мирослава Ігорівна – студентка групи 3ПІ-18б, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця.

Науковий керівник: ***Бабюк Наталя Петрівна–*** канд. техн. наук, доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця.

Myrhorodskyi Andrii V. **–** student of group 3PI-18b, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia.

Tretiak Myroslava I. – student of group 3PI-18b, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia.

Scientific supervisor: ***Babiuk Natalia P.*** – PhD in Engineering sciences, Associate Professor of Software department, Vinnytsia National Technical University, Vinnitsa.