***Тишківський І. І., магістрант, гр. ЗІПЗм-19-1,***
***Кубрак Ю. О., канд. техн. наук, доц. кафедри ІПЗ,***
*Державний університет «Житомирська політехніка»*

# МЕРЕЖЕВЕ СХОВИЩЕ НА БАЗІ АРХІТЕКТУРИ МІКРО-СЕРВІСІВ

На даний момент існує безліч різноманітних інтернет сховищ, що дозволяють зберігати особисті дані. Мережеві середовища зберігання даних є досить новою технологією і підлягають їх ретельному вивченню. Найчастіше при наданні послуг хмарних обчислень, постачальники звертають увагу на переваги програмної складової, залишаючи поза увагою апаратний аспект і приватність інформації [1].

Прикладом подібних сервісів є файлове сховище для синхронізації даних GoogleDrive, Microsoft OneDrive, Dropbox. Проблеми при розгортанні такого сервісу переважно є реалізація безпеки приватних даних як іншими користувачами, так і самим сервісом, який може здійснювати збір різноманітних відомостей користувача, наприклад для побудови персоналізованої реклами. Пропонований в якості аналога веб-сервіс включає в себе такі функції як завантаження або видалення файлів (і навіть більше), прозорість перед користувачами – відсутність збору будь-яких приватних відомостей користувачів при використанні сервісу і в майбутньому можливість запровадження шифрування файлів а також аутентифікацію користувачів.

В структурі проекту в якості основних засобів були використані мови програмування JavaScript на базі платформи Node.js, а також такі відомі фреймворки як Express.js (backend) і Angular.js (frontend) а також їх різноманітні бібліотеки [2]. База даних ApacheCassandra для побудови серверних запитів що відповідають за різні операції. Була також використана система контролю версій git, середовище розробки Atom. В якості системи управління проектом було обрано GitHub.

Програмна реалізація цього проекту була заснована на архітектурі мікро-сервісів, що дозволило створити програмний інтерфейс (API) який можливо зв'язати з різноманітними frontend реалізаціями для побудові сервісів і взаємозв'язків між їх компонентами. В основу проектування сховища було обрано архітектурний шаблон MVC (англ. "Model-view-controller"). Це дало змогу більш детально та зручно структурувати сам веб-сервіс, узгодити всі його компоненти і полегшити їх розширення. Процес завантаження даних на сервер відносно складний, він заснований на отриманні даних у вигляді потоку даних в кодуванні multipart/form-data, та його конвертування в буфер (або масив буферів залежно від розміру файлу) і відповідно його розміщення в базі. [3].

База даних (або простір імен даних бази) складається з двох таблиць: перша містить інформацію про метадані файлів, друга містить фактичні дані завантажених файлів [4]. Така структура спрощує управління файлами і прибирає утворення дублікатів.

Процес скачування відбувається майже так само як і завантаження, за відмінністю в необхідності попереднього конвертування даних [5].

Отже, розробка мережевого сховища відкрила можливість впровадження приватного такого сховища для користувачів та більш ретельно дослідити можливість його оптимізації. Подальша перспектива розглянутого наукового дослідження полягає в створенні в майбутньому веб-орієнтованої системи, яку можливо буде розгорнути на різних бажаних сервісах хмарних обчислень, наприклад AWS. А також можливість впровадження шифрування приватних даних і аутентифікації користувачів для підвищення рівня захисту. Користувачі зможуть розміщувати свої файли, видаляти їх, перейменовувати і звісно скачувати їх. Підсумовуючи все сказане вище – майбутній веб-сервіс може змінити підхід збереження даних в мережі роблячи його більш лояльним до користувачів і перш за все більш безпечним в плані захисту приватної інформації.

**Список літератури**

1. How to storefilesinCassandrawithNodeJs? [Електронний ресурс]. Режим доступу: https://keytodatascience.com/store-file-cassandra-nodejs/.
2. UploadingandDownloadingFiles: Bufferingin Node.js [Електронний ресурс]. Режим доступу: https://dzone.com/articles/uploading-and-downloading-files-buffering-in-nodej.
3. Working with Buffers [Електронний ресурс]. Режим доступу:https://nodejs.org/dist/latest-v14.x/docs/api/buffer.html.
4. Data Stax Node.jsDriver 4.6 Documentation [Електронний ресурс]. Режим доступу: https://docs.datastax.com/en/developer/nodejs-driver/4.6/getting-started/.
5. Angular.js Documentation [Електронний ресурс]. Режим доступу: https://angular.io/start.