DOI: https://doi.org/10.15276/ict.01.2024.22
УДК 004.414.38; 004.415.2; 004.91; 821

Проєктування архітектури програмної системи для читання та публікації аматорських літературних творів

Гурницька Вікторія Олександрівна[1)]
Бакалавр каф. Інженерії програмного забезпечення
E-mail: 9482098@stud.op.edu.ua
Сичков Віталій Сергійович[1)]
Асистент каф. Інженерії програмного забезпечення
ORCID: https://orcid.org/0009-0009-3709-2029; sychkov.v.s@op.edu.ua
[1)] Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна

АНОТАЦІЯ

У статті розглядаються функціональні можливості та ключові вимоги до вебзастосунку для читання та публікації аматорських літературних творів. На тлі сучасних змін у книжковій індустрії, зумовлених розвитком технологій та появою самвидаву, підкреслюється необхідність створення платформ, які забезпечують авторам можливість самостійної публікації контенту. Однією з головних проблем, з якими стикаються автори під час публікації, є втрата початкового форматування творів під час перенесення їх до текстових редакторів на вебплатформах. У роботі пропонується рішення цієї проблеми через розробку вебзастосунку з функцією імпорту вмісту текстових файлів зі збереженням їх первинного форматування. Стаття також містить порівняльний аналіз популярних вебплатформ для публікації творів, таких як Archive of Our Own, Wattpad та Аркуш, що допомогло виділити переваги та недоліки кожної з них. Зокрема, досліджено труднощі, пов'язані зі збереженням та обробкою неструктурованих даних, таких як зображення та текстові файли. Для розв'язання проблеми зберігання таких даних було проведено порівняльний аналіз трьох підходів: зберігання даних у реляційних базах даних, локальних файлових системах та хмарних сховищах. Враховуючи результати цього аналізу, для розроблюваного вебзастосунку було обрано інтеграцію з хмарним провайдером Amazon Web Services і використання хмарного сховища Amazon S3, яке забезпечує масштабованість, надійність та зручність в управлінні великими обсягами неструктурованих даних. Також вирішено застосувати CDN-сервіс Amazon CloudFront для покращення швидкості доступу до даних та підвищення загальної продуктивності системи. Окрім цього, в статті описано вибір технологічного стека для реалізації вебзастосунку, який включає Java, Spring Framework, MySQL, ORM Hibernate для серверної частини та HTML, CSS, React.js для клієнтської частини. Стаття висвітлює важливі аспекти проєктування архітектури вебзастосунку, що націлена на масштабованість і стійкість до великих обсягів даних та збільшення кількості активних користувачів, а також надає базу для подальшого вдосконалення системи.

Ключові слова: функціональні вимоги; архітектурна модель; хмарні сховища; локальні файлові системи; імпортування вмісту файлів; неструктуровані дані; реляційні бази даних; аматорські літературні твори; самвидав

Актуальність. Книжкова індустрія зазнала суттєвих трансформацій під впливом стрімкого розвитку технологій. Протягом багатьох років традиційне видавництво мало беззаперечний вплив, однак з появою Інтернету та розвитком електронних книг виникла альтернативна модель публікації творів – самвидав [1]. Це суттєво вплинуло на книжкову індустрію, маючи як позитивні, так і негативні наслідки. Розвиток самвидаву надав авторам більше свободи у виборі тем, дозволивши публікувати роботи, які не знайшли б підтримки у традиційних видавництвах. Однак, це також призвело до загострення конкуренції на книжковому ринку, через що маркетинг став найбільшою статтею витрат у видавничому процесі [2, 3]. З метою розв'язати цю проблему почали з'являтись вебзастосунки для читання та публікації літературних творів, які надають авторам-початківцям можливість знайти перших читачів.

Метою роботи є визначення функціональних можливостей та ключових вимог до вебзастосунку для читання та публікації аматорських літературних творів, аналіз й вибір оптимальних архітектурних рішень для забезпечення масштабованості та ефективної роботи програмної системи при збільшенні обсягу неструктурованих даних.

Функціонал вебплатформ для читання та публікації літературних творів орієнтований на просування контенту та залучення нової аудиторії, однак процес підготовки матеріалу до публікації, як-то форматування та редагування тексту, є не менш важливим етапом при створенні літературного контенту. Попри наявність у сучасних вебзастосунках вбудованих текстових редакторів, автори надають перевагу більш звичним програмним рішенням, як-от

Word, Google Docs чи Notion. Це обумовлено тим, що вони забезпечують вищий рівень надійності та стабільності, дозволяючи поступово працювати над твором, мінімізуючи ризики втрати тексту. Незважаючи на зручність використання таких програм, при публікації автори стикаються з проблемою: перенести текст твору у текстовий редактор вебзастосунку можливо лише вручну, що часто призводить до втрати початкового форматування. Внаслідок цього необхідно витрачати додатковий час на повторне оформлення тексту перед публікацією. Для вирішення цієї проблеми пропонується розробити вебзастосунок для читання та публікації літературних творів з функцією імпорту вмісту текстових файлів безпосередньо до вбудованого текстового редактора зі збереженням первинного форматування.

З метою визначення основних функцій та вимог до розроблюваної системи, проведемо порівняльний аналіз особливостей конкурентних аналогів, а саме Archive of Our Own [4], Wattpad [5] та Аркуш [6].

Archive of Our Own (AO3) – некомерційний міжнародний онлайн-архів, який слугує платформою для публікації фанатських творів. Наявність функціоналу для гнучкого сортування та фільтрації контенту значно спрощує пошук бажаних творів. Однак через складний, неінтуїтивний інтерфейс новим користувачам може бути важко орієнтуватися на сайті. Збільшення обсягу контенту й користувачів призводить до перевантаження серверів AO3. Маючи технічні та фінансові обмеження на масштабування системи, для оптимізації її роботи було введено обмеження на реєстрацію нових користувачів, яке передбачає очікування в черзі протягом 3-7 днів. Також згідно з вимогами платформи, використання зображень здійснюється через їх розміщення на сторонніх ресурсах (наприклад, Tumblr) та вбудовування в текст твору за допомогою гіперпосилань. Таким чином, файли зображень не зберігаються безпосередньо на серверах AO3.

Wattpad – міжнародна онлайн-спільнота для письменників та читачів, де користувачі можуть безоплатно читати та розміщувати свої твори. Головною перевагою платформи є низький поріг входу завдяки продуманому UX дизайну. На відміну від Archive of Our Own, Wattpad більше фокусується на просуванні творів, пропонуючи читачам твори на основі їх вподобань. При користуванні платформою було помічено, що якість зображень обкладинок на головній сторінці сайту значно нижча, ніж на інших сторінках. Ймовірно, це пов’язано із надмірним стисканням, яке використовується для пришвидшення завантаження головної сторінки, однак така оптимізація негативно впливає на візуальну привабливість сайту й зацікавленість користувачів у подальшому ознайомленні з матеріалами платформи.

Аркуш – мультимедійна вебплатформа для публікації літературних творів, віршів та аудіокниг, цільовою аудиторією якої є громадяни України. Вебзастосунок не надає широкі можливості пошуку та фільтрування, однак його головною метою є просування українських письменників шляхом проведення конкурсів, які дають авторам можливість отримати зворотний зв’язок і нових читачів. Оскільки Аркуш перебуває на етапі альфа-тестування, об’єктивно оцінити його технічні характеристики складно. Проте, останні оновлення продемонстрували значне покращення швидкодії платформи завдяки оптимізації зображень.

На підставі проведеного аналізу, результати якого наведено в Таблиці, було прийнято рішення інтегрувати переваги програмних систем-аналогів у розроблюваний вебзастосунок.

Таблиця. **Порівняльна характеристика продуктів-аналогів**

Характеристика	Застосунок			
	Wattpad	*Аркуш*	*AO3*	*FicScroll*
Інтуїтивно зрозумілий та зручний інтерфейс	+	+	-	+
Наявність вбудованого текстового редактора	+	+	+	+
Імпорт вмісту текстового файлу	-	-	-	+
Гнучкий пошук та сортування творів	-	-	+	+
Система відгуків та коментування	+	+	+	+

На Рис. 1 представлено основні функціональні можливості розроблюваної системи FicScroll, окрім процесів реєстрації та аутентифікації. Були визначені два типи акторів: гість (відвідувач вебсайту, що не ввійшов в обліковий запис або не зареєстрував його) та користувач (зареєстрований та аутентифікований відвідувач вебсайту). Головна відмінність між акторами полягає в функціональних можливостях: гість може лише пасивно споживати контент, тоді як зареєстрований користувач має можливість як споживати, так і публікувати твори на вебплатформі, а також коментувати, оцінювати та додавати роботи інших авторів до списку закладок.

Проведений аналіз конкурентних аналогів виявив, що розглянуті вебплатформи мають труднощі при роботі з неструктурованими даними, зокрема зображеннями, що негативно впливає на їх загальну продуктивність. Оскільки в розроблюваному вебзастосунку користувачам надаватиметься можливість завантажувати зображення обкладинок та HTML-файли з текстом творів, необхідно знайти оптимальне рішення для їх зберігання та обробки.

Існує кілька способів зберігання неструктурованих даних: бази даних, локальні файлові системи та хмарні сховища, тож проведемо їх порівняльний аналіз.

Бази даних використовують тип даних BLOB для зберігання неструктурованих даних, таких як зображення чи документи. Згідно з [7], “Binary large object” (BLOB) – це загальний термін, який використовується для опису обробки та зберігання довгих рядків даних системами керування базами даних». Зберігання BLOB у реляційній базі даних забезпечує високий рівень цілісності та узгодженості даних, а також надає високий рівень безпеки через вбудовані механізми контролю доступу та шифрування. Однак зберігання великих обсягів неструктурованих даних призводить до швидкого заповнення бази даних, що вимагає постійного масштабування системи. Вертикальне масштабування може тимчасово вирішити проблему браку дискового простору, однак його застосування обмежене апаратними характеристиками. Горизонтальне масштабування дозволяє розподілити навантаження між декількома серверами, але його впровадження в реляційних базах даних є складним [8], оскільки супроводжується проблемами з підтриманням узгодженості даних та збільшенням

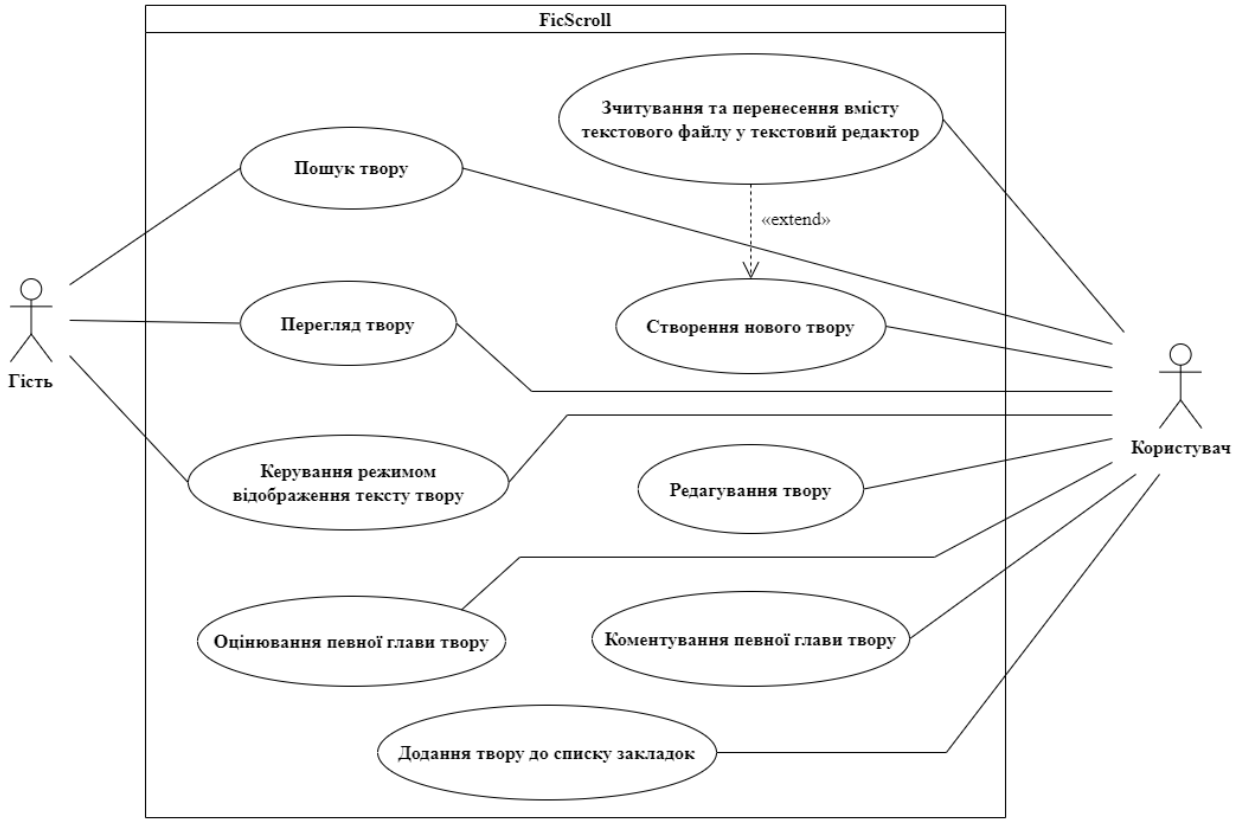

Рис. 1. **Діаграма варіантів використання розроблюваного вебзастосунку**

складності запитів. У дослідженні Microsoft [9] було встановлено, що бази даних працюють ефективніше з BLOB розміром до 256 кілобайтів. Однак, коли розмір BLOB перевищує один мегабайт, файлові системи мають явну перевагу. Оскільки розмір файлів у розроблюваній системі може значно варіюватися, реляційні бази даних не є оптимальним рішенням для зберігання неструктурованих даних через потенційні проблеми з продуктивністю.

Локальні файлові системи краще оптимізовані для зберігання та швидкого доступу до великих файлів і легше масштабуються горизонтально. Однак їхнє використання потребує безпосереднього налаштування та управління апаратним забезпеченням, значних витрат на забезпечення безпеки даних, їх резервне копіювання та адміністрування.

На противагу, хмарні сховища пропонують зручний і масштабований спосіб зберігання даних, звільняючи від необхідності прямого керування апаратним забезпеченням. Крім того, популярність цієї послуги зросла, оскільки вона надає дешеве рішення для резервного копіювання та аварійного відновлення, й потребує оплати лише за фактичне використання ресурсів [10]. Проте зберігання даних на сторонніх серверах несе в собі ризики, пов’язані з порушенням конфіденційності та безпеки. Незважаючи на те, що провайдери пропонують різноманітні засоби захисту, як-от шифрування чи двофакторна автентифікація, існує ймовірність того, що дані можуть бути викрадені або переглянуті неавторизованими користувачами [11]. Також відстань до серверів і час затримки при передачі даних через мережу можуть впливати на швидкість доступу до даних.

З огляду на результати порівняльного аналізу, представлених в Таблиці 2, було вирішено використовувати хмарне сховище для зберігання неструктурованих даних.

На ринку хмарних сервісів існує велика кількість провайдерів, проте лідерами галузі є Amazon Web Services (AWS), Google Cloud Platform (GCP) та Microsoft Azure. Кожен з них надає широкий спектр послуг, таких як обчислювальні ресурси, зберігання даних, мережеві сервіси, аналітика й обробка даних тощо, тому вибір оптимального провайдера більшою мірою залежить від конкретних потреб проєкту [12, 13]. Зважаючи на те, що розроблюваний вебзастосунок не передбачає використання екосистеми Microsoft, а також не потребує розширених можливостей аналізу даних, було обрано хмарне сховище Amazon S3. Також буде застосовано сервіс Amazon CloudFront, оскільки в більшості випадків використання CDN (Content Delivery Network) дозволяє знизити затримку розповсюдження контенту кінцевим користувачам та підвищити швидкість передачі даних завдяки кешуванню та розподіленій мережі серверів [14].

Загальна архітектурна модель вебзастосунку наведена на Рис. 2.

Таблиця 2. **Порівняльна характеристика методів зберігання BLOB-даних**

Характеристика	Метод зберігання		
	База даних	*Файлова система*	*Хмарне сховище*
Продуктивність	При збільшенні обсягу даних продуктивність знижується	Залежить від конфігурації сервера й типу файлової системи	Висока продуктивність із можливістю масштабування
Масштабованість	Обмежена вертикальна та складна горизонтальна масштабованість	Обмежена горизонтальна та вертикальна масштабованість	Автоматичне масштабування
Резервне копіювання	Може бути складним та ресурсомістким	Може бути складним при великому обсязі даних	Автоматичне резервне копіювання
Безпека	Високий рівень безпеки на рівні бази даних	Залежить від конфігурації системи та налаштувань прав доступу	Залежить від заходів безпеки, які вживають хмарні провайдери
Доступ до даних	Може бути повільним при великих обсягах даних і частих запитах	Швидкий доступ, але залежить від конфігурації сервера	Швидкий доступ, але залежить від швидкості мережі

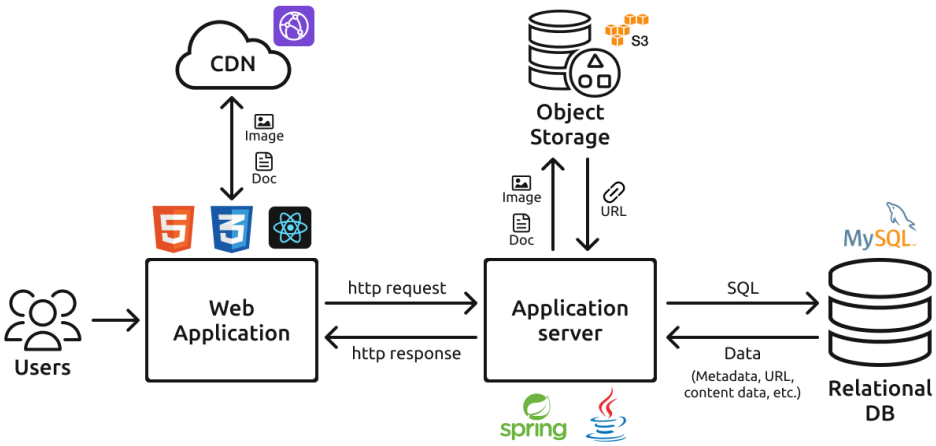

Рис. 2. **Архітектурна модель розроблюваного вебзастосунку**

Для розробки серверної частини програми було обрано мову Java та фреймворк Spring. Для зберігання метаданих буде використана база даних MySQL, а для спрощення роботи з нею – ORM Hibernate. Клієнтська частина буде розроблена за допомогою HTML, CSS та React.js.

Висновки. Швидке зростання обсягів даних та збільшення активних користувачів можуть потребувати впровадження додаткових рішень для оптимізації роботи вебзастосунку. На даному етапі архітектура системи є динамічною та підлягає подальшому розвитку для забезпечення стабільної роботи в умовах великих навантажень. Це може включати перехід на мікросервісну архітектуру, що дозволить легше адаптувати окремі модулі до мінливих вимог користувачів, а також забезпечити високу доступність і відмовостійкість системи.

СПИСОК ЛІТЕРАТУРИ

1. Mark L. “The fine print of self-publishing: a primer on contracts, printing costs, royalties, distribution, ebooks, and marketing”. *Minneapolis: North Loop Books*. 2016. p. 127–134.

2. Eckstut A., Sterry D. H. “The essential guide to getting your book published: How to Write It, Sell It, and Market It… Successfully”. *New York: Workman Publishing Company*. 2010. p. 330–336.

3. Penn J. “Successful Self-Publishing: How to self-publish and market your book in ebook and print”. *Oxford: Curl Up Press*. 2017. p. 71–86.

4. “Archive of Our Own”. *Organization for Transformative Works.* – Available from: https://archiveofourown.org. – [Accessed: 29.07.2024].

5. “Wattpad”. *Naver Corporation.* – Available from: https://www.wattpad.com. – [Accessed: 29.07.2024].

6. «Аркуш». – Available from: https://arkush.net. – [Accessed: 29.07.2024].

7. “Information Technology (IT) Glossary – Essential Information Technology (IT) Terms & Definitions”. *Gartner*. – Available from: https://www.gartner.com/en/information-technology/glossary/blob-binary-large-object. – [Accessed: 12.08.2024].

8. Ali W., Shafique M. U., Majeed M. A., Raza A. “Comparison between SQL and NoSQL Databases and Their Relationship with Big Data Analytics”. *Asian Journal of Computer Science and Information Technology*. 2019. DOI: https://doi.org/10.9734/ajrcos/2019/v4i230108.

9. Sears R., van Ingen C., Gray J. “To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem?”. *Technical Report MSR-TR-2006-45*. 2006. DOI: https://doi.org/10.48550/arXiv.cs/0701168.

10. Obrutsky S. “Cloud storage: Advantages, Disadvantages and Enterprise Solutions for Business”. *EIT New Zealand*. 2016. – Available from: https://www.researchgate.net/publication/305508410_Cloud_Storage_Advantages_Disadvantages_and_Enterprise_Solutions_for_Business. – [Accessed: 15.08.2024].

11. Liubchenko V. V., Volkov D. V. “Cyber-aware threats and management strategies in cloud environments”. *Herald of Advanced Information Technology*. 2024; 7 (2): 158–170. DOI: https://doi.org/10.15276/hait.07.2024.11.

12. Choudhary A., Verma P., Rai P. “Comparative Study of Various Cloud Service Providers: A Review”. *2022 International Conference on Power, Energy, Control and Transmission Systems*. 2022. DOI: https://doi.org/10.1109/ICPECTS56089.2022.10047594.

13. Borra P. “Comparison and Analysis of Leading Cloud Service Providers (AWS, Azure and GCP)”. *International Journal of Advanced Research in Engineering and Technology*. 2024; 15 (3): 266–278. DOI: https://doi.org/10.17605/OSF.IO/T2DHW.

14. Persico V., Montieri A., Pescapè A. “On the Network Performance of Amazon S3 Cloud-Storage Service”. *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*. 2016. p. 113–118. DOI: https://doi.org/10.1109/CloudNet.2016.16.

DOI: https://doi.org/10.15276/ict.01.2024.22
UDC 004.414.38; 004.415.2; 004.91; 821

Design of software system architecture for reading and publishing amateur literary works

Viktoriia O. Hurnytska[1)]
Bachelor, Department of Software Engineering
E-mail: 9482098@stud.op.edu.ua
Vitalii S. Sychkov[1)]
Assistant, Department of Software Engineering
ORCID: https://orcid.org/0009-0009-3709-2029; sychkov.v.s@op.edu.ua
[1)] Odesa Polytechnic National University, 1, Shevchenko Ave. Odesa, 65044, Ukraine

ABSTRACT

The article discusses the functionality and key requirements for a web application for reading and publishing amateur literary works. Against the backdrop of current changes in the book industry caused by the development of technology and the emergence of self-publishing, the need to create platforms that provide authors with the opportunity to publish content independently is emphasized. One of the main problems that authors face when publishing is the loss of the original formatting of their works when transferring them to text editors on web platforms. This paper proposes a solution to this problem by developing a web application with the function of importing the content of text files while preserving their original formatting. The article also contains a comparative analysis of popular web-based platforms for publishing works, such as Archive of Our Own, Wattpad, and Arkush, which helped to highlight the advantages and disadvantages of each of them. In particular, the difficulties associated with storing and processing unstructured data, such as images and text files, were investigated. To solve the problem of storing such data, a comparative analysis of three approaches was conducted: storing data in relational databases, local file systems, and cloud storage. Taking into account the results of this analysis, the web application under development was chosen to integrate with the Amazon Web Services cloud provider and use Amazon S3 cloud storage, which provides scalability, reliability, and convenience in managing large amounts of unstructured data. It was also decided to use the Amazon CloudFront CDN service to improve the speed of data access and increase overall system performance. In addition, the article describes the choice of a technology stack for implementing the web application, which includes Java, Spring Framework, MySQL, ORM Hibernate for the server side and HTML, CSS, React.js for the client side. The article highlights important aspects of designing a web application architecture aimed at scalability and resilience to large amounts of data and an increase in the number of active users, and provides a basis for further system improvement.

Keywords: Functional requirements; architectural model; cloud storage; local file systems; importing file content; unstructured data; relational databases; amateur literary works; self-publishing