Автор:
Максим Воронін Андрійович,
Студент 41 ІПЗ групи
Науковий керівник:
Бородкіна Ірина Лаврентіївна,
кандидат технічних наук, доцент
кафедри програмної та
комп'ютерної інженерії

ВЕБЗАСТОСУНОК ДЛЯ ПЕРЕГЛЯДУ ВІДЕОКОНТЕНТУ

Анотація. У дослідженні проведено проектування, створення, впровадження та тестування вебзастосунку для перегляду відеоконтенту відповідно до сучасних тенденцій та вимог стримінгових вебсервісів. Основну увагу приділено побудові ефективної архітектури із застосуванням актуальних вебтехнологій, що дає можливість реалізувати повноцінну систему перегляду відеоконтенту з підтримкою адаптивного потокового відтворення та рівня безпеки, що відповідає сучасним вимогам. У роботі використано HTML, JavaScript, CSS, React, Node.js, Firebase, FFmpeg і протокол HLS. Також розглянуто розгортання серверної інфраструктури на базі IIS у Windows Server 2022 та змодельовано процес опрацювання і конвертації відео в адаптивний формат стримінгу HLS.

Ключові слова: вебдодаток, вебтехнології, медіаконтент, потокове відтворення, адаптивне відео, HLS, FFmpeg, React, Node.js, Firebase, IIS.

Вступ. Сучасна епоха характерна своїм надшвидким технологічним прогресом і, як наслідок, стрімким зростанням обсягів мультимедійного контенту. Тому не дивно, що у таких умовах відеосервіси стали невід'ємною частиною у сферах розваг, освіти, маркетингу та дистанційного навчання. Оскільки кількість користувачів таких систем постійно зростає, підсилюються вимоги до їх якості, продуктивності та зручності, що викликає потребу в створенні ефективних, масштабованих і зручних рішень. З метою розробки такого рішення, у роботі реалізовано клієнт-серверну систему із використанням сучасного стеку технологій. Для оопрацювання відео застосовано FFmpeg, а відтворення здійснюється через плеєр із підтримкою HLS, що забезпечує автоматичне налаштування якості під швидкість з'єднання користувача [5]. Підготовлений контент розміщується на сервері застосунку, розгорнутому на базі IIS у Windows Server 2022, що сприяє стабільній роботі та високій продуктивності системи.

Постановка задачі. У процесі створення стримінгової платформи необхідно враховувати сучасні вимоги до потокового відтворення [5]. Зокрема, реалізацію адаптивного потокового відтворення на основі HLS, ефективне опрацювання медіафайлів за допомогою FFmpeg, а також розробку зручного користувацького інтерфейсу. Важливою складовою є інфраструктура застосунку, зокрема для зберігання та доставки контенту, а також Firebase для безпечного зберігання даних. Задачею даного дослідження є розробка та реалізація комплексного рішення, що поєднує сучасні підходи до веброзробки з ефективними засобами потокової передачі відео.

Мета. Метою цієї роботи є створення вебплатформи для відтворення медіаконтенту, яка забезпечує адаптивний стримінг, зручну взаємодію з користувачем та відповідає актуальним вимогам до продуктивності й надійності вебсистем. Для досягнення цієї мети були вирішені такі задачі:

- Аналіз сучасного стану та тенденцій розвитку вебтехнологій для трансляції медіаконтенту, з акцентом на методи адаптивного потокового відеомовлення.
- Виконання порівняльного аналізу існуючих сервісів відеостримінгу та протоколів передачі даних для визначення оптимального технологічного стеку.

- Формування переліку функціональних та нефункціональних вимог до системи, з урахуванням аспектів продуктивності, безпеки та зручності інтерфейсу.
- Проектування архітектури вебзастосунку на основі клієнт-серверної моделі з інтеграцією хмарних сервісів (Firebase) та серверної логіки (Node.js).
- Розробка підсистеми підготовки відеоконтенту з використанням інструментарію FFmpeg для конвертації медіафайлів у формат адаптивного стримінгу HLS.
- Реалізація клієнтської частини застосунку на базі бібліотеки React, із забезпеченням підтримки програвача з автоматичним підлаштуванням якості відео.
- Налаштування серверної інфраструктури на базі IIS у середовищі Windows Server 2022 для ефективного хостингу та дистрибуції медіафайлів.
- Проведення комплексного тестування (зокрема модульного та end-to-end за допомогою Cypress), оцінка результатів та визначення шляхів подальшої модернізації системи.

Основна частина. Ключовим етапом розробки стало визначення оптимальної архітектури [3]. У роботі застосовано клієнт-серверний підхід із розподілом функціональності між окремими рівнями: інтерфейс користувача реалізовано з використанням React [8], серверну логіку – на Node.js, а зберігання та синхронізацію даних забезпечує Firebase.

Протягом усього життєвого циклу розробки здійснювався контроль якості. Було проведено модульне тестування компонентів інтерфейсу, а також end-to-end тестування із використанням Cypress для перевірки коректності відтворення відео. Окрім цього, виконано інтеграційне тестування взаємодії між клієнтською та серверною частинами системи.

Перетворення відеоматеріалів у формат адаптивного стримінгу (плейлисти .m3u8 та сегменти .ts) здійснюється за допомогою FFmpeg [6], що забезпечує ефективну багатопотокову конвертацію. Підготовлений контент зберігається та доставляється через сервер на базі IIS у Windows Server 2022 [2] з використанням механізмів кешування, стиснення та захищеного з’єднання. Застосування Node.js і Firebase дозволило реалізувати надійну автентифікацію користувачів і синхронізацію метаданих відео[7, 1].

Висновки. Використання сучасних технологій роботи з відеоконтентом дало змогу створити ефективну, масштабовану та безпечну платформу для потокового відтворення відео. React своєю модульною архітектурою забезпечує чудову гнучкість інтерфейсу, тоді як Node.js ефективно обробляє запити та виконує серверну логіку. FFmpeg забезпечує легку конвертацію відео у формат HLS, а IIS виступає надійним середовищем для розгортання застосунку, забезпечуючи високу швидкість передачі відеоконтенту, безпеку з’єднання та зручність адміністрування [2]. Інтеграція з Firebase покриває питання безпечного зберігання даних та миттєвої синхронізації станів, що робить систему сприятливою для масштабування та готовою для майбутнього розширення функціоналу. Перспективним напрямком є автоматизація процесів обробки відео та вдосконалення серверної інфраструктури, зокрема перехід до більш гнучких рішень на основі реляційних баз даних або власних серверних реалізацій та використанням CDN у майбутньому.

Список використаних джерел

1. Firebase Documentation: developer documentation. URL: https://firebase.google.com/docs (дата звернення: 30.04.2026).
2. IIS documentation. Microsoft Learn. URL: https://learn.microsoft.com/en-us/iis/ (дата звернення: 01.05.2026).
3. Бородкіна І. Л., Бородкін Г. О. Інженерія програмного забезпечення: посібник. Київ: НУБіП України, 2018. 251 с.

4. Галицький О.В., Микитенко П.В. Особливості формування професійної компетентності фахівців комп’ютерних наук. Педагогічна Академія: наукові записки. 2025. № 14. URL: https://pedagogicalacademy.com/index.php/journal/article/download/607/494
5. Документація Apple (HLS): HTTP Live Streaming. Apple Developer Documentation. URL: https://developer.apple.com/documentation/http-live-streaming (дата звернення: 30.04.2026).
6. Документація FFmpeg: ffmpeg documentation. FFmpeg. URL: https://ffmpeg.org/ffmpeg.html (дата звернення: 30.04.2026).
7. Навчальний посібник Node.js: Introduction to Node.js. Node.js. URL: https://nodejs.org/learn (дата звернення: 30.04.2026).
8. Навчальний посібник React: Quick Start. React. URL: https://react.dev/learn (дата звернення: 30.04.2026).