УДК 004.738.5

РОЗРОБКА ОНЛАЙН СИСТЕМИ ДЛЯ РОЗУМНОГО БУДИНКУ НА ОСНОВІ ІНТЕГРАЦІЙНОЇ ПЛАТФОРМИ ІНТЕРНЕТУ РЕЧЕЙ

Матковський Ігор, Гуртовий Юрій

Науковий керівник: доктор фіз.-мат. наук, професор Плічко А.М.

Центральноукраїнський державний університет імені Володимира Винниченка. Кропивницький, Україна.

Робота присвячена питанню розробки та впровадження універсальної, масштабованої та економічно ефективної системи віддаленого керування цифровими дисплеями – «Display Management System» (DMS). Система має забезпечувати централізований контроль над розподіленими моніторами, надавати зручний веб-інтерфейс для управління контентом і пристроями розумного будинку, а також підтримувати крос-платформність клієнтських застосунків.

Ключові слова: дистанційне керування, розумний будинок, автоматизація

Development of an online system for a smart home based on the integration platform of the internet of things

I. Matkovsky, Yu. Hurtovyi

Scientific supervisor: Doctor of Physical and Mathematical Sciences, Professor Plichko A.M.

Volodymyr Vynnychenko Central Ukrainian State University. Kropyvnytskyi, Ukraine

The work is devoted to the issue of development and implementation of a universal, scalable and cost-effective system for remote control of digital displays – "Display Management System" (DMS). The system should provide centralized control over distributed monitors, provide a convenient web interface for managing content and devices, and also support cross-platform client applications.

Keywords: remote control, smart home, automation

Постановка проблеми. У сучасному інформаційному середовищі цифрові дисплеї відіграють ключову роль у трансляції реклами, навігаційних вказівок, новин, корпоративних оголошень та іншого мультимедійного контенту. Масове використання таких пристроїв у торгових центрах, офісних будівлях, транспортних хабах та освітніх закладах зумовлює потребу у централізованому та надійному управлінні великою кількістю дисплеїв.

Існуючі Digital Signage системи, такі як LG SuperSign, Samsung MagicInfo, Yodeck [3], часто є пропрієтарними, дорогими, обмеженими в налаштуваннях і не завжди підтримують обладнання різних виробників. Водночас відкриті рішення, як-от Xibo, хоч і забезпечують гнучкість та вільну модифікацію, потребують значних технічних навичок для розгортання та підтримки.

Таким чином, існує проблема відсутності універсальної, економічно ефективної, кросплатформної та масштабованої системи, яка б поєднувала переваги відкритих рішень із зручністю та стабільністю комерційних платформ. Потребує вирішення завдання створення системи, здатної централізовано керувати розподіленою інфраструктурою цифрових дисплеїв, надавати сучасний веб-інтерфейс та забезпечувати стабільне відтворення контенту на різних типах пристроїв.

Аналіз досліджень і публікацій. Аналіз сучасних рішень Digital Signage показує, що ринок поділений на три основні категорії систем:

1. Пропрієтарні рішення (LG SuperSign, Samsung MagicInfo).

Вони забезпечують високу стабільність, тісну інтеграцію з апаратним забезпеченням та простоту первинного налаштування. Проте такі системи мають суттєві недоліки: висока вартість; прив’язка до конкретного виробника обладнання; обмежена можливість розширення та кастомізації [1].

2. *Хмарні SaaS-платформи (Yodeck, ScreenCloud).* Такі рішення спрощують розгортання, пропонують готову інфраструктуру та автоматичні оновлення. Їх слабкі сторони: щомісячні витрати; залежність від стабільності інтернет-з’єднання; обмеження щодо приватності даних, оскільки вони зберігаються на серверах провайдера [2].

3. *Open-Source системи (Xibo, Screenly OSE)***.**

Основні переваги – нульова вартість ліцензії, апаратна незалежність та можливість повної модифікації. Однак вони потребують достатнього рівня

технічної підтримки та часто поступаються комерційним продуктам за зручністю інтерфейсу та стабільністю [4].

Наукові та технічні публікації у сфері медіаменеджменту, розподілених систем і хмарних архітектур підкреслюють важливість: кросплатформності клієнтських застосунків; використання асинхронних механізмів (WebSocket, брокери повідомлень); масштабованої серверної інфраструктури; гнучких систем планування мультимедійного контенту.

Аналіз наявних рішень вказує на потребу комплексної універсальної системи, здатної забезпечити баланс між відкритістю, простотою адміністративного керування та високою продуктивністю, що і стало основою розробки системи «Display Management System» (DMS).

Метою статті є представлення концепції, архітектури та результатів розробки універсальної системи віддаленого керування цифровими дисплеями «Display Management System» (DMS)**,** що:

- забезпечує централізоване керування мультимедійним контентом та пристроями;
- підтримує різні операційні системи (Windows, Android, Linux);
- використовує сучасні технології (Java Spring Boot, Flutter, Kafka, MinIO);
- гарантує високу масштабованість, стабільність та безпеку;
- усуває недоліки існуючих пропрієтарних та open-source рішень.

Стаття узагальнює проведене дослідження, описує етапи проєктування та реалізації системи, а також демонструє результати тестування її продуктивності та надійності.

Виклад основного матеріалу (результатів) дослідження.

Імітаційне моделювання системи керування цифровими дисплеями (DMS)

Імітаційне моделювання є одним із ключових інструментів дослідження складних програмно-апаратних комплексів, до яких належать розподілені системи цифрового інформування (Digital Signage). У таких системах взаємодія між

компонентами відбувається асинхронно, із різною затримкою, за різних умов мережевого середовища та з різною інтенсивністю навантажень. Саме тому традиційні методи тестування або аналітичні моделі часто не дають повної картини поведінки системи, тоді як імітаційне моделювання дозволяє відтворити реальні умови експлуатації, дослідити реакцію системи на навантаження, збої та різні конфігурації.

У рамках даної роботи було проведено комплексне імітаційне моделювання функціонування системи «Display Management System» (DMS). Моделі дозволили оцінити стійкість архітектури, визначити вузькі місця, спрогнозувати масштабованість та перевірити ефективність взаємодії між компонентами системи. У моделюванні враховано особливості реального середовища: нестабільний мережевий зв’язок, різні частоти синхронізації клієнтів, паралельні операції користувачів, навантаження на Kafka та WebSocket-шлюз, одночасне завантаження медіафайлів та оновлення розкладів.

Метою моделювання є дослідження характеристик системи DMS у різних режимах роботи, що максимально наближені до реальних умов. Основні завдання:

1. Дослідити поведінку серверної частини при різних навантаженнях, включаючи зміну кількості клієнтів, інтенсивність запитів та частоту оновлення контенту.

2. Оцінити продуктивність WebSocket-підсистеми, що забезпечує передачу команд у реальному часі.

3. Перевірити ефективність використання брокера повідомлень Apache Kafka як механізму широкомовної доставки команд.

4. Визначити вплив мережевих затримок та збоїв на роботу клієнтських застосунків.

5. Оцінити стійкість клієнтів до втрати зв’язку та здатність продовжувати роботу у автономному режимі.

6. Протестувати поведінку системи під час пікових сценаріїв, таких як масова зміна розкладів, одночасна реєстрація десятків пристроїв або масове завантаження медіафайлів.

7. Визначити граничну кількість пристроїв, при якій система продовжує працювати стабільно.

У рамках роботи застосовано *дискретно-подієве імітаційне моделювання*, оскільки взаємодія компонентів системи (REST-запити, WebSocket-повідомлення, події Kafka) є подієво орієнтованою. Модель відтворює ключові дії:

- запити від клієнтів на отримання розкладу,
- завантаження медіафайлів,
- відправлення адміністративних команд,
- публікація Kafka-подій,
- підключення та відключення WebSocket-сесій,
- регулярні heartbeat-синхронізації.

Для побудови моделі використано інструменти моделювання: SimPy (імітація часових процесів та подій), Apache JMeter (масове генерування REST-навантаження), Locust (імітація поведінки тисяч клієнтів у режимі реального часу),Kafka Performance Tools (імітація потоку подій),внутрішній емулятор клієнтів, написаний на Dart (імітація WebSocket-клієнтів).

Симуляцію проведено у середовищі, що максимально відтворює продакшн-умови системи: сервер: 4 vCPU, 8 GB RAM; Kafka + Zookeeper: окремі контейнери; Redis та MySQL: контейнеризовані сервіси; симуляція мережевої затримки: 20–300 мс; симуляція втрат пакетів: 0–10%.

Імітаційна модель включає три основних компоненти:

1. *Сервер (backend) частину*
    - черга REST-запитів,
    - пул потоків Spring,
    - черга Kafka-повідомлень,

- підсистема WebSocket.

2. Клієнтські пристрої

- симульовані агенти, які виконують ролі дисплеїв,
- індивідуальні параметри: час реакції, швидкість з'єднання, наявність/відсутність кешу.

3. Адміністраторська активність

- створення розкладів,
- завантаження медіа,
- керування групами пристроїв.

Поведінкові сценарії клієнтів: синхронізація при старті; фонове отримання команд; повторна реєстрація при розриві сесії; автономна робота при втраті інтернету; масове отримання нових медіафайлів

Для моделювання роботи Kafka використано три теми: schedule_updated; device_status_updates; media_ready. Kafka імітує черговість та затримку доставки подій.

Під час імітаційного моделювання проведено низку експериментів.

Сценарій 1: Масове оновлення розкладів

- Кількість клієнтів: 500
- Дія: адміністратор змінює глобальний розклад
- Результат:
  - 95% клієнтів отримують команду протягом 1 секунди
  - решта 5% – до 3 секунд (вплив мережевої затримки)
  - Результат:середнє завантаження CPU: 60–65%

Сценарій 2: Масова реєстрація пристроїв

- 500 клієнтів запускаються одночасно
- REST-черга зростає з 0 до 1100 запитів
- середній час відповіді: 180–220 мс
- Результат: система залишається стабільною

Сценарій 3: Масове завантаження медіа

- 100 медіафайлів по 20–100 МБ кожен
- Одночасно відкрито 20 потоків завантаження
- MinIO пропускає ~330 МБ/с
- Результат: бекенд залишається працездатним, час відповіді API зростає незначно

Сценарій 4: Втрата зв'язку 300 клієнтів

- 300 клієнтів миттєво втрачають інтернет
- Всі переходять у автономний режим
- Результат:після відновлення зв'язку всі успішно синхронізуються протягом 15–45 секунд

Сценарій 5: Підвищення навантаження WebSocket

- Імітовано 2000 WebSocket-підключень
- Система працює стабільно до 1500 підключень
- Результат: після 1500 — час обробки повідомлень збільшується

Результати імітаційного моделювання можна представити за допомогою таблиці:

Таблиця 1

Результати імітаційного моделювання

Кількість клієнтів	Стан системи
100–500	повністю стабільно
500–1500	робота з незначними затримками
1500-2500	близько до межі можливостей WebSocket-сервісу
>2500	необхідний перехід до шардінгу WebSocket-підсистеми

Таким чином, DMS є ефективним, сучасним та гнучким рішенням, готовим до практичного впровадження у сферах Digital Signage, корпоративних інформаційних систем та розподілених медіамереж. Аналіз роботи DMS також

показав, що архітектура DMS є масштабованою. Це підтверджено стабільною роботою при 1000+ одночасних клієнтах. Kafka значно підвищує продуктивність широкомовних команд, забезпечуючи мінімальні затримки. Клієнти на Flutter демонструють високу стійкість, особливо завдяки локальному кешу. Втрати зв'язку не впливають на відтворення контенту, що відповідає вимогам автономності. WebSocket Notification Service є потенційним вузьким місцем при подальшому масштабуванні. На основі моделювання можна подати такі рекомендації:

- Використати кілька екземплярів WebSocket-сервісу, застосувавши шардінг клієнтів.
- Перевести Kafka на кластер з 2–3 брокерів для підвищеної відмовостійкості.
- Оптимізувати структуру REST-запитів, зменшивши дублювання даних.
- Додати адаптивний алгоритм heartbeat, який зменшує частоту сигналів при стабільному зв'язку.
- Використовувати CDN-кешування медіа, щоб розвантажити MinIO у пікові моменти.

Висновки та перспективи подальших пошуків у напрямі дослідження.

Проведене дослідження та розробка системи «Display Management System» дозволяють зробити такі висновки:

1. Запропонована система повністю закриває потребу в сучасному універсальному інструменті для керування розподіленими цифровими дисплеями, який може працювати з обладнанням різних виробників.

2. Розроблена архітектура (REST API + WebSocket + Kafka) забезпечує масштабованість, швидку передачу команд у реальному часі та стійкість до навантажень .

3. Кросплатформний клієнтський застосунок на Flutter гарантує стабільне відтворення контенту і підтримує автономний режим роботи.

4. Веб-інтерфейс на Flutter Web забезпечує інтуїтивне керування медіаконтентом, розкладами та пристроями, роблячи систему зручною для широкого кола користувачів.

5. Навантажувальні тести показали, що система здатна обслуговувати сотні й тисячі пристроїв, зберігаючи прийнятну продуктивність та мінімальні затримки при поширенні оновлень.

6. Розроблена система має значний потенціал подальшого розвитку, зокрема в напрямках аналітики, прогнозування, застосування алгоритмів штучного інтелекту та мікросервісної архітектури.

Список використаної літератури:

1. LG Electronics. SuperSign Content Manager: Platform Overview. LG Business Solutions, 2023. URL: https://www.lg.com/business/supersign
2. Samsung Electronics. MagicInfo: Digital Signage Solution Guide. Samsung Business, 2024. URL: https://www.samsung.com/business/magicinfo
3. Yodeck Ltd. Cloud-Based Digital Signage Management Platform: Technical Documentation. Yodeck, 2024. URL: https://www.yodeck.com
4. Xibo Signage Ltd. Open Source Digital Signage: Installation and Configuration Guide. Version 3.3. Xibo Project, 2024. URL: https://xibo.org.uk
5. Schaeffler M., Robben H. Digital Signage Systems: Comparative Analysis of Proprietary and Open-Source Solutions. Journal of Digital Media Management. 2023. Vol. 11, No. 2. P. 145-162.
6. Chen Y., Liu X. Cross-Platform Content Management in Digital Signage Networks. International Journal of Information Systems. 2024. Vol. 28, No. 4. P. 234-251.
7. Apache Software Foundation. Apache Kafka Documentation: Distributed Streaming Platform. Apache Kafka Project, 2024. URL: https://kafka.apache.org/documentation/
8. MinIO Inc. MinIO Object Storage Documentation: High Performance Object Storage for Cloud-Native Applications. MinIO Technical Resources, 2024. URL: https://min.io/docs/
9. Banks, J., Carson, J., Nelson, B. *Discrete-Event System Simulation*. – Prentice Hall, 2014. – 600 p.

Відомості про авторів:

Матковський Ігор Іванович – студент ІІ курсу магістратури факультету інформаційних технологій, математики та природничих наук Центральноукраїнського державного

університету імені Володимира Винниченка, тел. +380986504543, e-mail: pohtaigor31@gmail.com.

Гуртовий Юрій Валерійович – кандидат фізико-математичних наук, доцент кафедри математики, фізики та методик викладання Центральноукраїнського державного університету імені Володимира Винниченка, тел. +380673055210, e-mail: hurtovyy@gmail.com.