СЕКЦІЯ «ІНФОРМАТИКА»

УДК 004.5:004.92

DOI 10.31651/2076-5886-2024-1-57-67

PACS

ПЕДЧЕНКО Максим Анатолійович
студент спеціальності «Інформаційні системи та технології» Черкаського національного університету імені Богдана Хмельницького
e-mail: pedchenko.maksym@vu.cdu.edu.ua

СЕРДЮК Олександр Анатолійович
кандидат економічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені БогданаХмельницького
e-mail: serdyuk@ukr.net
ORCID 0000-0002-3919-4661

АРХІТЕКТУРА ТА РЕАЛІЗАЦІЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ІНТЕРАКТИВНОЇ ВІЗУАЛІЗАЦІЇ ПЕРСОНАЛЬНИХ ДАНИХ МУЗИЧНОГО СТРІМІНГУ

У статті описано архітектуру та практичну реалізацію веб-застосунку для аналізу й інтерактивної візуалізації персональних даних музичного стрімінгу платформи Spotify. Актуальність теми зумовлена стрімким зростанням обсягів даних, що генеруються користувачами стрімінгових сервісів, а також недостатністю вбудованих аналітичних засобів для їх глибокого дослідження. На підставі порівняльного аналізу існуючих рішень – сторонніх сервісів Last.fm та Stats.fm, а також вбудованих засобів Apple Music, YouTube Music і Spotify – визначено ключові обмеження наявних підходів та сформульовано вимоги до розроблюваної системи. В основу архітектурного рішення покладено принцип розподілу серверної та клієнтської частин: серверна частина реалізована на платформі Node.js з використанням фреймворку Express.js і документоорієнтованої СУБД MongoDB, клієнтська – на основі бібліотеки React з підтримкою суворої типізації через TypeScript і централізованого керування станом засобами Redux Toolkit. Для авторизації застосовано протокол OAuth 2.0 у варіанті потоку обміну кодом авторизації, що забезпечує безпечний серверний доступ до Spotify Web API без зберігання облікових даних користувача. Повнота ретроспективних даних про прослуховування досягнута завдяки імпорту розширеної історії стрімінгу відповідно до механізму запиту персональних даних за вимогами GDPR. Отримані дані нормалізуються, фільтруються та зберігаються локально для подальшої агрегованої обробки. Візуалізацію реалізовано за допомогою бібліотеки Recharts і охоплює щомісячну динаміку прослуховувань, розподіл активності протягом доби, рейтинги виконавців, треків та альбомів з детальними індивідуальними сторінками. Наукова новизна роботи полягає у розробці комплексного підходу до інтеграції Spotify Web API з персональною аналітичною системою, що поєднує безперервний збір поточних даних та повний ретроспективний імпорт й надає користувачеві інтерактивні засоби дослідження власних музичних уподобань, недоступні в жодному з розглянутих аналогів.

***Ключові слова**: Spotify API, OAuth 2.0, MongoDB, React, TypeScript, Node.js, Redux Toolkit, Recharts, візуалізація даних, музичний стрімінг, GDPR.*

Вступ

Цифровий музичний стрімінг докорінно змінив характер взаємодії людей з

музичним контентом. Такі платформи, як Spotify, Apple Music та YouTube Music, зробили мільярди треків доступними у будь-який момент, сформувавши нові патерни споживання аудіоконтенту. Щодня сотні мільйонів користувачів генерують колосальні масиви даних: кожне прослуховування, кожен пропуск чи повтор фіксуються платформою і у своїй сукупності здатні детально описати індивідуальні музичні уподобання.

Персоналізація є ключовим елементом сучасних стрімінгових сервісів. Не лише наявність широкої музичної бібліотеки, а й здатність платформи розуміти слухача та пропонувати відповідний контент – ось що формує лояльність аудиторії. Такий рівень персоналізації досягається завдяки складним алгоритмам аналізу даних, що прогнозують майбутні уподобання на основі попередньої активності.

Попри розвиненість алгоритмічної складової, самі платформи надають вкрай обмежені засоби для того, щоб користувач міг самостійно дослідити свої дані. Щорічний звіт Spotify Wrapped охоплює лише фіксований річний інтервал і не дозволяє варіювати метрики чи часовий діапазон. Apple Music Replay функціонує аналогічно, а YouTube Music і поготів обмежується загальними агрегованими показниками без деталізації.

Сторонні сервіси аналітики – Last.fm [1] та Stats.fm [2] – частково заповнюють цю прогалину, проте мають власні суттєві обмеження. Last.fm не охоплює даних, що передують реєстрації у сервісі, а Stats.fm обмежений доступом лише до 50 останніх записів через стандартний ендпоінт Spotify API. У підсумку жоден з існуючих інструментів не надає користувачеві повної довічної ретроспективи з одночасно гнучкими та інтерактивними засобами дослідження.

Дослідження у суміжній галузі – Music Information Retrieval (MIR) – підтверджують, що аналіз музичних уподобань є ефективним інструментом для розуміння поведінки користувачів і побудови персоналізованих систем рекомендацій [5]. Проте академічні роботи здебільшого зосереджені на серверних алгоритмах, а не на розробці клієнтської аналітики, доступної кінцевому користувачеві в інтерактивному режимі.

Зазначене визначає актуальність розробки спеціалізованого веб-застосунку, що поєднує: повний імпорт ретроспективних даних прослуховування, їх збереження на власному сервері та детальну інтерактивну візуалізацію з гнучким вибором часових діапазонів.

Мета статті – описати методи проєктування та реалізації веб-застосунку для збору, зберігання й інтерактивної візуалізації персональних даних стрімінгу музики у сервісі Spotify, а також обґрунтувати обрані архітектурні рішення і стек технологій з позиції їх ефективності для вирішення поставленої задачі.

Виклад основного матеріалу

1. Огляд існуючих рішень та постановка задачі

Ринок аналітичних сервісів для стрімінгових платформ представлений двома категоріями: вбудованими засобами самих платформ і сторонніми застосунками-надбудовами, що використовують відкриті API.

Вбудовані засоби платформ. Apple Music надає функцію «Replay», що формує підсумковий перелік найпрослуховуваніших треків за поточний рік. Функціонал залишається обмеженим і не передбачає деталізованої аналітики чи користувацького налаштування параметрів. YouTube Music пропонує базову статистику уподобань і популярних треків, проте інтерактивне дослідження даних не підтримується. Spotify формує детальний щорічний звіт «Spotify Wrapped» і підтримує персоналізовані

плейлисти «Discover Weekly», однак у режимі реального часу не надає засобів для самостійного дослідження власної статистики.

Сторонні рішення. Last.fm [1] є піонерським сервісом трекінгу та аналітики прослуховувань, що фіксує активність на різних платформах через механізм «скроблінгу» і зводить дані у єдиний профіль. Основний недолік – відсутність ретроспективних даних до початку використання сервісу та залежність повноти статистики від безперебійної роботи «скроблінгу». Stats.fm [2] спрямований безпосередньо на користувачів Spotify і забезпечує докладну аналітику та візуалізацію: найпрослуховуваніші треки, виконавці, жанрові уподобання. Суттєве обмеження платформи Spotify API – доступ лише до 50 останніх записів через ендпоінт recently-played – звужує повноту даних у безкоштовному варіанті використання.

Порівняльний аналіз виявив такі системні вади розглянутих інструментів:

- неповнота ретроспективних даних – жодне рішення не охоплює повної довічної історії прослуховувань без додаткових умов;
- статичність представлення – відсутність засобів для інтерактивного дослідження та гнучкого налаштування параметрів відображення;
- обмеженість часових діапазонів – здебільшого доступні лише фіксовані інтервали (тиждень, місяць, рік) без можливості вибору довільного проміжку;
- брак аналізу внутрішньодобових патернів – жоден з розглянутих сервісів не надає деталізованого аналізу активності у розрізі годин доби.

Таким чином, задача полягає у розробці застосунку, що усуває зазначені обмеження: забезпечує повний імпорт ретроспективних даних, їх збереження та обробку на власному сервері, а також надає інтерактивні та гнучко налаштовувані засоби візуалізації.

2. Архітектура системи та стек технологій

2.1 Загальна архітектура

Розроблений застосунок побудовано за принципом клієнт-серверної архітектури. Серверна частина реалізує RESTful API [7], взаємодіє зі Spotify Web API [4] та відповідає за збереження, нормалізацію й агрегацію даних прослуховування. Клієнтська частина є SPA (Single Page Application), що відображає інтерактивні дашборди з персональною статистикою. Розподіл обов'язків між рівнями забезпечує незалежне масштабування кожного компонента та дозволяє реалізовувати складні агрегаційні запити безпосередньо на серверному рівні, а не перекладати цей обчислювальний тягар на браузер клієнта.

2.2 Серверна частина: Node.js, Express.js і MongoDB

Node.js і Express.js. Серверна частина реалізована на платформі Node.js [6] – середовищі виконання JavaScript на стороні сервера з подієво-орієнтованою неблокуючою архітектурою введення-виведення. Це рішення є обґрунтованим для застосунків з інтенсивним обміном даними, де конкурентна обробка численних запитів є критичною вимогою. Фреймворк Express.js надає зручне визначення маршрутів і ланцюжок middleware-функцій, що забезпечують обробку HTTP-запитів на різних рівнях: валідацію вхідних даних (бібліотека Zod), обробку завантажуваних файлів (Multer), аутентифікацію (перевірка JWT-токенів), логування запитів (Morgan). Модульна структура маршрутизатора (`/spotify`, `/artist`, `/album`, `/track`, `/oauth`) полегшує підтримку та розширення кодової бази.

MongoDB і Mongoose. Для зберігання даних обрано документоорієнтовану СУБД MongoDB [3], що надає гнучку схему документів і ефективно опрацьовує різнорідні дані великого обсягу. Схема бази даних включає шість основних колекцій (рис. 1):

`users` (профілі користувачів та токени авторизації), `infos` (записи про прослуховування – часова мітка, ідентифікатор треку, статус фільтрації), `tracks`, `artists`, `albums` (метадані музичного контенту, отримані від Spotify), `importstates` (журнал імпортів розширеної історії). Для роботи з MongoDB у Node.js застосовано бібліотеку Mongoose [8] – ODM-рішення (Object Data Modelling), що реалізує схемо-орієнтоване моделювання даних з вбудованою валідацією та механізмом віртуальних полів (virtuals) для реалізації зв'язків між колекціями без зберігання надлишкових посилань.

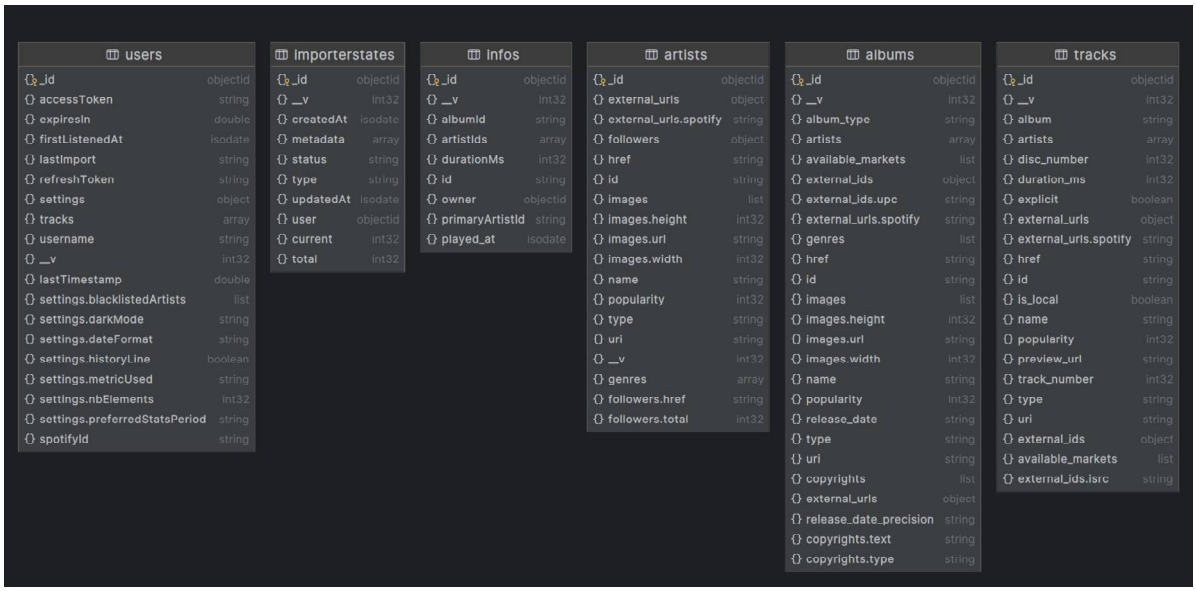

Рис. 1. Схема колекцій бази даних MongoDB

Особливу роль в архітектурі відіграє агрегаційний пайплайн MongoDB: операції `$lookup`, `$match`, `$unwind`, `$group` виконуються безпосередньо на сервері бази даних, що дозволяє ефективно обчислювати показники (наприклад, загальний час прослуховування виконавця за заданий період або розподіл активності по годинах доби) без перевантаження серверного процесу.

Для забезпечення ізоляції та відтворюваності середовища бази даних використано контейнеризацію засобами Docker із Docker Compose. Це гарантує узгодженість конфігурації на різних машинах та спрощує розгортання і міграцію.

2.3 Клієнтська частина: React, TypeScript, Redux Toolkit і Recharts

React і TypeScript. Клієнтська частина реалізована за допомогою бібліотеки React [8] у поєднанні з TypeScript [9] – суворо типізованим суперсетом JavaScript. Така комбінація підвищує надійність коду, зменшує кількість помилок часу виконання та поліпшує підтримуваність великої кодової бази. Компонентна архітектура React дозволяє будувати інтерфейс як ієрархію перевикористовуваних самодостатніх елементів зі своєю логікою та станом. JSX-розмітка компілюється у TypeScript (файли з розширенням .tsx), що дозволяє перевіряти типи пропсів компонентів на етапі компіляції.

Redux Toolkit. Для керування глобальним станом застосунку використано Redux Toolkit [9] – офіційно рекомендований спосіб написання Redux-логіки, що суттєво скорочує шаблонний код. Централізований сховок (store) охоплює зрізи (slices) для даних профілю користувача, системних повідомлень і стану імпорту. Утиліта createAsyncThunk забезпечує обробку асинхронних дій у трьох станах: pending, fulfilled,

rejected – що дозволяє відображати стан завантаження та помилки безпосередньо у компонентах без дублювання логіки.

Material-UI. Компонентна бібліотека Material-UI (MUI) [10] застосовується для побудови естетичного і функціонального інтерфейсу: таблиць, кнопок, діалогових вікон, навігаційних елементів, компонентів вибору дат. Система тем MUI забезпечує узгодженість візуального оформлення у всьому застосунку.

Recharts. Бібліотека Recharts [11] слугує основою для всіх інтерактивних візуалізацій. Вона нативно інтегрована з React та надає широкий набір готових компонентів: `LineChart`, `BarChart`, `AreaChart` (потоковий графік). Для забезпечення єдиного стилю й уникнення дублювання коду над стандартними компонентами Recharts побудовано три кастомні абстракції – лінійний графік, стовпчаста діаграма та складена стовпчаста діаграма, – які централізовано задають CSS-стилі, кастомні підказки (tooltips) та налаштування осей.

2.4 Авторизація через OAuth 2.0

Інтеграція зі Spotify Web API потребує аутентифікації користувача. Стандарт OAuth 2.0 [5] визначає кілька варіантів авторизаційних потоків: код авторизації, код авторизації з PKCE-розширенням, облікові дані клієнта та неявний потік. Для розробленого застосунку обрано Authorization Code Flow (рис. 2): він виконується на стороні сервера та повертає поряд із токеном доступу також токен оновлення (`refresh token`), що дає змогу підтримувати сеанс без повторного введення облікових даних.

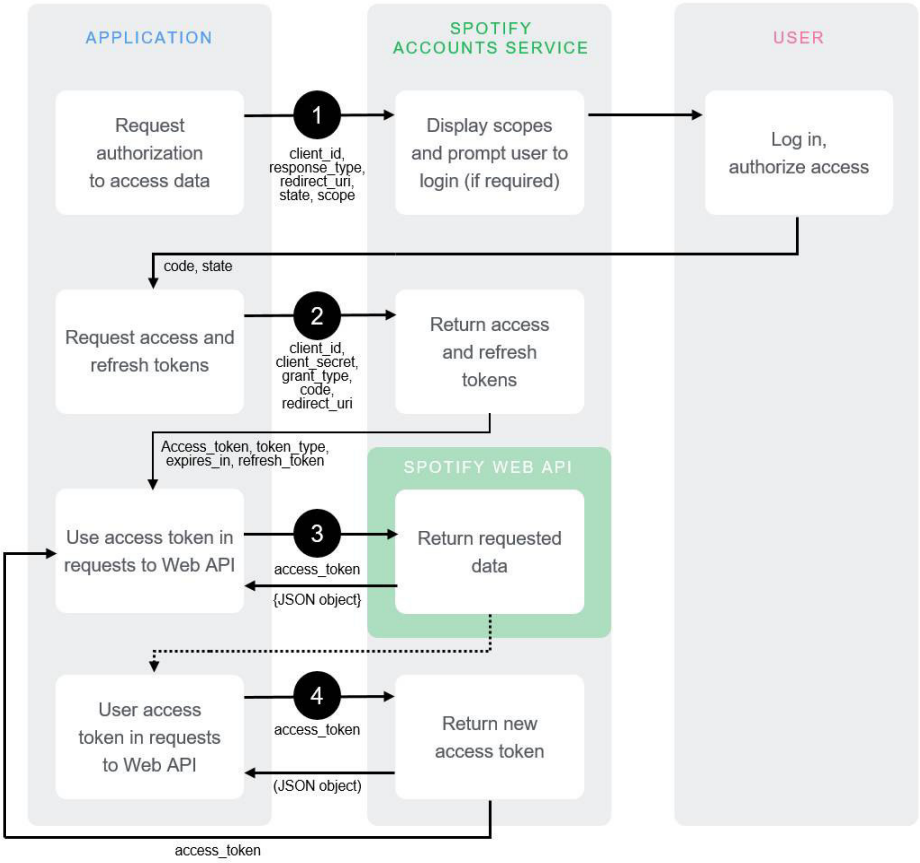

Рис. 2. Діаграма потоку авторизації OAuth 2.0

Етапи авторизації виконуються у наступній послідовності:

1. Застосунок перенаправляє користувача до сервісу Spotify Accounts з параметрами запиту дозволу.
2. Після підтвердження переліку запитуваних прав Spotify повертає одноразовий код авторизації на вказаний callback URL.
3. Сервер обмінює цей код на пару токенів (access token / refresh token), надсилаючи POST-запит безпосередньо зі свого боку – без участі клієнтського браузера.
4. Токен доступу використовується для підписаних запитів до Spotify Web API; після закінчення терміну дії він оновлюється автоматично без участі користувача.

Для захисту сеансу в межах власного застосунку отриманий токен доступу Spotify конвертується на стороні сервера у власний JWT-токен. Це мінімізує ризики витоку: навіть якщо JWT-токен буде перехоплено, зловмисник отримає доступ лише до функцій поточного застосунку, але не до облікового запису Spotify.

3. Методи отримання та обробки даних

3.1 Поточна історія прослуховувань

Для безперервного відстеження активності сервер регулярно – кожні 5 хвилин – звертається до ендпоінту Spotify Web API `api.spotify.com/v1/me/player/recently-played`. Ендпоінт підтримує параметри `after` та `before` (Unix timestamp у мілісекундах), що дозволяє щоразу отримувати лише нові записи, уникаючи дублювання. Стандартне обмеження ендпоінту – дані лише за останні 24 години – компенсується частим опитуванням.

Отримані записи про прослуховування збагачуються метаданими (виконавець, альбом, жанр, тривалість, показник популярності) через пакетний ендпоінт `api.spotify.com/v1/tracks`, що дозволяє запитувати відомості про декілька треків за один виклик API. Це суттєво знижує загальну кількість запитів та ризик перевищення ліміту.

Spotify обмежує частоту запитів на основі ковзного вікна у 30 секунд. У разі отримання статусу відповіді 429 Too Many Requests застосунок автоматично очікує 60 секунд перед повторною спробою, що забезпечує стале безперебійне опитування.

3.2 Розширена ретроспективна історія (GDPR-імпорт)

Регламент GDPR, зокрема стаття 15 [12], надає суб'єктам персональних даних право на доступ до інформації, яку контролер зберігає про них. Spotify, дотримуючись цих вимог, надає у налаштуваннях приватності можливість запитати розширену (довічну) історію стрімінгу – докладний журнал всієї активності акаунту з моменту його реєстрації.

Після обробки запиту (до 30 днів) користувач отримує ZIP-архів з JSON-файлами. Кожен об'єкт у файлах містить численні поля, але для цілей застосунку використовуються лише три: `ts` (часова мітка UTC), `ms_played` (тривалість відтворення у мілісекундах) та `spotify_track_uri` (ідентифікатор треку). Решта полів відкидається. Записи, в яких трек відтворювався менше 30 секунд, відфільтровуються як несуттєві – адже вони відповідають пропуску треку, а не його прослуховуванню.

Після нормалізації та фільтрації дані імпортуються до колекції `infos`, а стан імпорту (прогрес, кількість записів) фіксується у колекції `importstates`. Цей підхід дозволяє отримати повну ретроспективну картину музичних уподобань без залежності від моменту реєстрації у сторонніх сервісах.

4. Результати: функціональні можливості застосунку

Реалізований застосунок надає користувачеві розгалужену систему аналітичних

дашбордів, структурованих за тематичними розділами. Усі розділи підтримують гнучкий вибір часового вікна аналізу: фіксовані варіанти (останній день, тиждень, місяць, рік, весь час) і довільний діапазон через компонент вибору дат. Усі часові мітки зберігаються у форматі UTC, що забезпечує коректність відображення незалежно від часового поясу користувача.

4.1 Головна сторінка

Головна сторінка (рис. 3) формує загальний огляд музичної активності за обраний період. Зведені картки відображають кількість прослуханих треків, сумарний час прослуховування у хвилинах і кількість унікальних виконавців з порівнянням із відповідним попереднім періодом. Лінійний графік відображає часову динаміку прослуховувань, стовпчаста діаграма – розподіл активності по 24 годинах доби. Окремо виділено «найкращого виконавця» та «найкращий трек» з відповідними показниками.

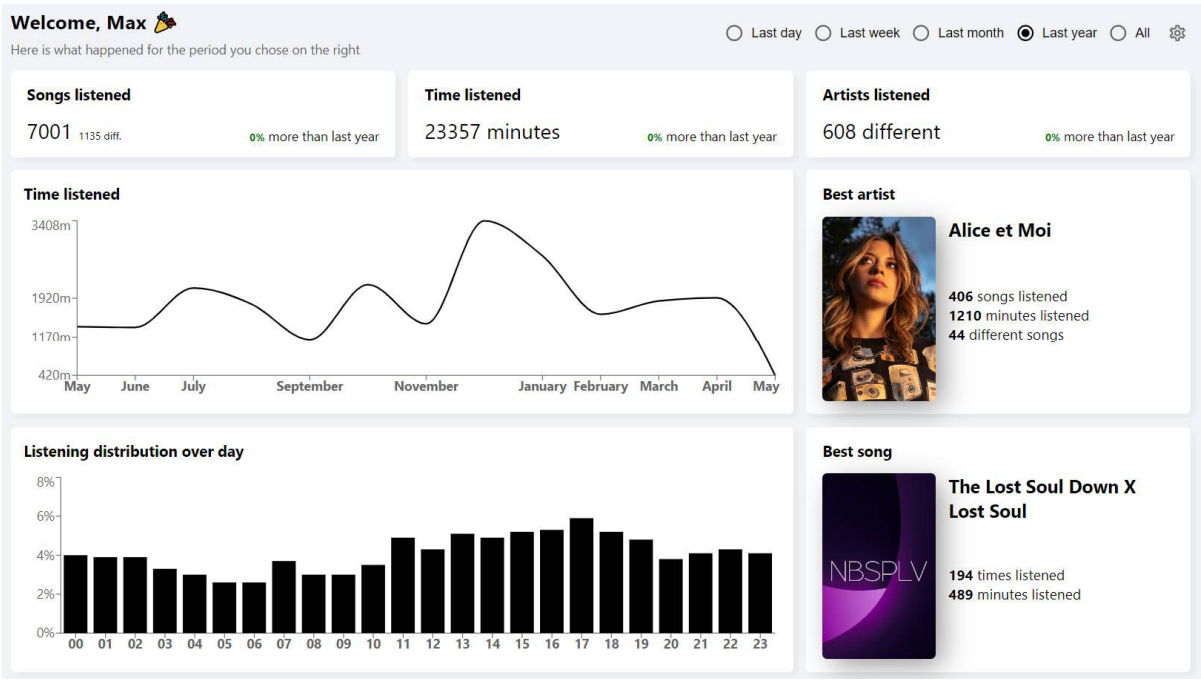

Рис. 3. Головна сторінка застосунку

4.2 Розширена статистика

Розділ «Всі статистики» включає взаємопов’язані візуалізації. Стовпчаста діаграма «Найкращі виконавці» відображає топ-виконавців з їхніми фотографіями під стовпцями для зручної ідентифікації. Потоковий графік «Розподіл прослуховування за виконавцями» ілюструє зміну часток виконавців у загальному часі прослуховування протягом року, виявляючи зсуви у музичних уподобаннях. Стовпчаста діаграма «Розподіл по годинах доби» показує відсотковий розподіл активності для виявлення пікових годин слухання. Складена стовпчаста діаграма «Найкращі виконавці для кожної години» деталізує топ-виконавців (треків або альбомів – перемикається за вибором користувача) для кожної з 24 годин.

Три додаткові лінійні графіки у щомісячному розрізі розкривають характеристики прослуховуваного контенту: середня дата виходу альбому (відображає схильність до нових релізів чи ретро), середня кількість фічерингів на трек (інтерес до колаборацій), середня популярність треків (нішові чи масові вподобання).

4.3 Рейтинги та детальна статистика

Розділ «Топи» (рис. 4) містить таблиці найпрослуховуваніших треків, виконавців і альбомів із детальними метриками: кількість відтворень (абсолютне значення та відсоток від загального), сумарний час прослуховування, жанрова приналежність (для виконавців). При переході на сторінку конкретного елемента відображається поглиблена статистика: перша та остання дата прослуховування, два місяці з найбільшою активністю, список останніх відтворень, розподіл прослуховувань по годинах доби.

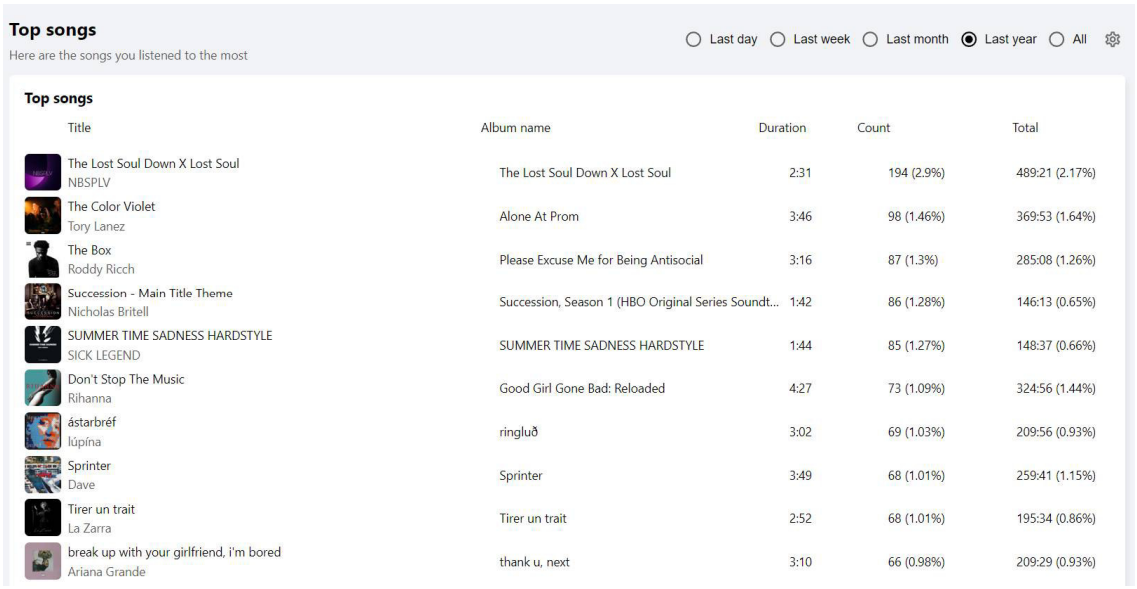

Рис. 4. Сторінка рейтингу треків

5. Аналіз результатів та наукова новизна

5.1 Порівняльний аналіз

Розроблений застосунок суттєво розширює функціональність у порівнянні з аналогами. У таблиці 1 наведено зіставлення ключових характеристик.

Найбільш принциповою відмінністю є підтримка повного GDPR-імпорту та аналізу розподілу активності за годинами доби – функцій, відсутніх у будь-якому з розглянутих аналогів.

5.2 Наукова новизна та внесок

Наукова новизна роботи включає такі складники.

По-перше, запропоновано та реалізовано методологію повного імпорту ретроспективних даних стрімінгу через механізм GDPR-запиту з подальшою нормалізацією (зниження регістру, уніфікація формату часових міток), фільтрацією (відсів відтворень коротше 30 секунд) і збереженням у власній базі даних. Такий підхід принципово відрізняється від стандартного використання Spotify API, обмеженого останніми 50 записами.

По-друге, реалізовано гібридну стратегію збору даних: регулярне опитування поточного ендпоінту (кожні 5 хвилин) забезпечує актуальність нових даних, тоді як GDPR-імпорт заповнює ретроспективний контекст від початку існування акаунту. Ця комбінація дозволяє підтримувати безперервну та повну часову шкалу прослуховувань.

По-третє, розроблено архітектурне рішення персоналізованої аналітики, у якому вся обробка даних виконується на стороні власного сервера. Це унезалежнює

аналітичні можливості від поточних обмежень Spotify API та дозволяє реалізовувати агрегаційні запити довільної складності.

По-четверте, впроваджено аналіз часових патернів у розрізі годин доби – що розкриває нові виміри музичної поведінки (пікові години слухання, прив'язаність певних виконавців до певного часу доби), недоступні в стандартних аналітичних сервісах.

Таблиця 1

Порівняння функціональних можливостей систем аналізу даних Spotify

Характеристика	Last.fm	Stats.fm	Spotify Wrapped	Розроблений застосунок
Повна ретроспективна історія	Ні	Частково	Лише за рік	Так
Інтерактивна візуалізація	Часткова	Так	Ні	Так
Довільний часовий діапазон	Ні	Частково	Ні	Так
Аналіз за годинами доби	Ні	Ні	Ні	Так
Детальна сторінка виконавця	Часткова	Так	Ні	Так
Аналіз характеристик контенту	Ні	Ні	Ні	Так
Підтримка GDPR-імпорту	Ні	Ні	–	Так

5.3 Обмеження та перспективи розвитку

Серед поточних обмежень застосунку слід відзначити: ліміт у 25 користувачів без розширення квоти розробника Spotify; відсутність підтримки декількох стрімінгових платформ одночасно; аналітика доступна лише для музики, прослуханої через Spotify.

Перспективними напрямами подальшого розвитку є: (а) застосування методів машинного навчання для автоматичного виявлення прихованих закономірностей у музичних уподобаннях та побудова прогностичних моделей активності; (б) реалізація порівняльного аналізу між двома профілями користувачів; (в) інтеграція з Last.fm API для збагачення даних про жанри та суміжних виконавців; (г) розширення квоти розробника для підтримки ширшої аудиторії.

Висновки

У статті описано процес проєктування та реалізації веб-застосунку для аналізу й

інтерактивної візуалізації персональних даних музичного стрімінгу Spotify. Проведений порівняльний аналіз існуючих рішень показав, що жодне з них не надає одночасно повної ретроспективної історії, гнучкого вибору часового діапазону та глибокої інтерактивної аналітики.

Для усунення виявлених обмежень розроблено застосунок на основі клієнт-серверної архітектури: серверна частина на Node.js/Express.js і MongoDB забезпечує збереження та ефективну агрегацію даних, клієнтська частина на React/TypeScript/Redux Toolkit і Recharts надає інтерактивні дашборди з багатовимірною аналітикою. Авторизацію реалізовано через OAuth 2.0 Authorization Code Flow, що забезпечує безпечний серверний доступ без зберігання облікових даних у клієнтській частині. Унікальною функціональною особливістю є підтримка повного імпорту ретроспективних даних через GDPR-запит у поєднанні з регулярним опитуванням поточного ендпоінту.

Реалізований застосунок надає користувачам принципово нові можливості для дослідження музичних уподобань: розподіл активності за годинами доби та місяцями, динаміка зміни улюблених виконавців протягом року, характеристики прослуховуваного контенту (середня популярність, вік релізів, кількість фічерингів), детальна статистика окремих виконавців, треків і альбомів. Отримані результати демонструють ефективність обраної архітектури та відкривають перспективи для подальшого розширення аналітичних можливостей – зокрема в напрямку застосування методів машинного навчання для виявлення прихованих закономірностей у музичній поведінці користувачів.

Список використаної літератури:

1. Last.fm | Play music, find songs, and discover artists [Електронний ресурс]. – Режим доступу: https://last.fm
2. Stats.fm (Formerly Spotistats for Spotify) [Електронний ресурс]. – Режим доступу: https://stats.fm
3. MongoDB Documentation [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/docs
4. Spotify Web API | Spotify for Developers [Електронний ресурс]. – Режим доступу: https://developer.spotify.com/documentation/web-api
5. Schedl M., Gómez E., Urbano J. Music Information Retrieval: Recent Developments and Applications // Foundations and Trends in Information Retrieval. – 2014. – Vol. 8, No. 2-3. – P. 127–261.
6. Node.js Documentation [Електронний ресурс]. – Режим доступу: https://nodejs.org/docs/latest/api
7. Lim G. Beginning Node.js, Express & MongoDB Development. – Independently Published, 2019.
8. React Documentation [Електронний ресурс]. – Режим доступу: https://react.dev/learn
9. Getting Started | Redux Toolkit [Електронний ресурс]. – Режим доступу: https://redux-toolkit.js.org/introduction/getting-started
10. Overview – Material UI [Електронний ресурс]. – Режим доступу: https://mui.com/material-ui/getting-started
11. Recharts. Redefined chart library built with React and D3 [Електронний ресурс]. – Режим доступу: https://recharts.org
12. Article 15 GDPR. Right of access by the data subject | GDPR-Text.com [Електронний ресурс]. – Режим доступу: https://gdpr-text.com/read/article-15

References:

1. Last.fm | Play music, find songs, and discover artists. [Electronic resource]. – Access: https://last.fm
2. Stats.fm (Formerly Spotistats for Spotify). [Electronic resource]. – Access: https://stats.fm
3. MongoDB Documentation. [Electronic resource]. – Access: https://www.mongodb.com/docs
4. Spotify Web API | Spotify for Developers. [Electronic resource]. – Access: https://developer.spotify.com/documentation/web-api
5. Schedl M., Gómez E., Urbano J. Music Information Retrieval: Recent Developments and Applications. Foundations and Trends in Information Retrieval. 2014. Vol. 8, No. 2-3. P. 127–261.
6. Node.js Documentation. [Electronic resource]. – Access: https://nodejs.org/docs/latest/api
7. Lim G. Beginning Node.js, Express & MongoDB Development. Independently Published. 2019.
8. React Documentation. [Electronic resource]. – Access: https://react.dev/learn

9. Getting Started | Redux Toolkit. [Electronic resource]. – Access: https://redux-toolkit.js.org/introduction/getting-started
10. Overview – Material UI. [Electronic resource]. – Access: https://mui.com/material-ui/getting-started
11. Recharts. Redefined chart library built with React and D3. [Electronic resource]. – Access: https://recharts.org
12. Article 15 GDPR. Right of access by the data subject. [Electronic resource]. – Access: https://gdpr-text.com/read/article-15

PEDCHENKO Maksym,
Student, Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
SERDIUK Oleksandr,
Candidate of Economic Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine

ARCHITECTURE AND IMPLEMENTATION OF A WEB APPLICATION FOR INTERACTIVE VISUALISATION OF PERSONAL MUSIC STREAMING DATA

Summary. Introduction. *The rapid growth of music streaming services has led to an accumulation of large volumes of personal data about users' listening habits. Despite the availability of basic analytics features within platforms such as Spotify, Apple Music, and YouTube Music, these tools are largely static and do not support interactive exploration or flexible time-range selection. Third-party services such as Last.fm and Stats.fm partially address this gap, yet they are limited by incomplete retrospective data coverage and constrained analytical capabilities.*

Purpose. *The purpose of this article is to describe the design principles and implementation details of a web application for collecting, storing, and interactively visualising personal Spotify music streaming data, and to justify the chosen technology stack and architectural decisions from the perspective of their effectiveness.*

Results. *A full-stack web application was developed using a client-server architecture. The backend is implemented on Node.js and Express.js with a MongoDB database managed via Mongoose, enabling scalable storage and complex server-side aggregation. User authentication is handled through the OAuth 2.0 Authorization Code Flow, which provides secure server-side access to the Spotify Web API along with refresh tokens for seamless session maintenance. A hybrid data collection strategy was implemented: regular polling of the recently-played endpoint every five minutes ensures up-to-date data, while complete retrospective history is obtained via GDPR data requests from Spotify – a combination unavailable in any existing analogue. The frontend is built with React and TypeScript, with Redux Toolkit for state management, Material-UI for interface components, and Recharts for interactive chart rendering. Three custom chart abstractions (line chart, bar chart, stacked bar chart) provide consistent styling across all dashboards. The application delivers multi-faceted analytical views: listening activity over time, hourly distribution charts, dynamic rankings of artists/tracks/albums with detailed individual statistics pages, stream graphs of artist distribution, and charts of content characteristics (average album release date, number of features per track, song popularity). Flexible date range selection – including fully custom intervals – is supported across all views.*

Conclusion. *The developed application addresses the key limitations of existing solutions by providing full retrospective listening history via GDPR import, flexible time-range analysis, and hourly-level activity breakdown. The proposed hybrid data collection methodology and the architectural approach of server-side aggregation represent contributions to the field of personalised music analytics. The results open perspectives for further development, including the integration of machine learning methods for automatic discovery of hidden patterns in users' musical behaviour.*

Keywords: *Spotify API, OAuth 2.0, MongoDB, React, TypeScript, Node.js, Redux Toolkit, Recharts, data visualisation, music streaming, GDPR.*

Одержано редакцією 09.09.2024 р.
Прийнято до публікації 30.10.2024 р.