Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка інформаційної системи для моніторингу музичних вподобань користувачів стрімінгових сервісів

Виконав: студент IV курсу, групи СТ-41

спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)

(підпис) Кузьо О.О. (прізвище та ініціали)

Керівник (підпис) Никитюк В.В. (прізвище та ініціали)

Нормоконтроль (підпис) Шимчук Г.В. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Михалик Д.М. (прізвище та ініціали)

Тернопіль
2021

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 126 Інформаційні системи та технології
(шифр і назва спеціальності)
Студенту Кузьо Олегу Олеговичу
(прізвище, ім'я, по батькові)
1. Тема роботи Розробка інформаційної системи для моніторингу музичних вподобань користувачів стрімінгових сервісів

Керівник роботи Никитюк Вячеслав Вячеславович, к.т.н., доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «01» лютого 2021 року № 4/7-63
2. Термін подання студентом завершеної роботи 25 червня 2021р.
3. Вихідні дані до роботи Літературні та інтернет джерела щодо системи для моніторингу музичних вподобань користувачів стрімінгових сервісів

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Предметне досліження та виділення основних аспектів роботи.дослідження. 1.1 Опис предмету дослідження. 1.2 Порівняльний аналіз існуючих інструментів для моніторингу стрімінгових сервісів. 1.3. Визначення оптимальних рішень для розробки кінцевого продукту. 1.4 Концептуальна модель. 1.5. Висновки до розділу. 2. Проєктування та створення кінцевого продукту. 2.1 Проєктування та візуалізація алгоритмів роботи компонентів в системі. 2.2 Створення програмно-апаратного середовища. 2.3 Експериментальна частина. 2.4 Функціональне та структурне тестування. 2.5 Висновки до розділу. 3. Безпека життєдіяльності, основи хорони праці. 3.1 Значення адаптації в трудовому процесі. 3.2 Загальні вимоги безпеки з охорони праці для користувачів ПК. Висновки. Список використаних джерел.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Титульна сторінка. 2 Тема, мета, задачі дослідження. 3 Опис предмету дослідження. 4 Концептуальна модель. 5 Блок-схема роботи інформаційної системи. 6 Діаграма класів. 7 Структура бази даних. 8 Діаграма компонентів системи. 9 Головна сторінка. 10 Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи хорони праці	Гурик О.Я., доцент кафедри МТ		

7. Дата видачі завдання 25 січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	25.01.2021	*Виконано*
2.	Пошук джерел інформації та складання плану роботи	26.01.2021-31.01.2021	*Виконано*
3.	Переклад та опрацювання інформаційних джерел	01.02.2021-07.02.2021	*Виконано*
4.	Розроблення програмного продукту	08.02.2021-12.02.2021	*Виконано*
5.	Оформлення розділу «Предметне досліження та виділення основних аспектів роботи»	13.02.2021-17.02.2021	*Виконано*
6.	Оформлення розділу «Проєктування та створення кінцевого продукту»	18.02.2021-22.02.2021	*Виконано*
7.	Оформлення «Безпека життєдіяльності, основи охорони праці»	23.02.2021	*Виконано*
8.	Оформлення кваліфікаційної роботи	08.06.2021	*Виконано*
9.	Нормоконтроль	10.06.2021	*Виконано*
10.	Перевірка на плагіат	11.06.2021	*Виконано*
11.	Попередній захист кваліфікаційної роботи	22.06.2021	*Виконано*
12.	Захист кваліфікаційної роботи	25.06.2021	

Студент _____ Кузьо О.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Никитюк В.В.
(підпис)

АНОТАЦІЯ

Розробка інформаційної системи для моніторингу музичних вподобань користувачів стрімінгових сервісів // Кваліфікаційна робота освітнього рівня «Бакалавр» // Кузьо Олега Олеговича // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СТ-41 // Тернопіль, 2021 // С. – 65 , рис. – 22, табл. – 0, кресл. – 0, додат. – 0, бібліогр. – 43.

Ключові слова: стрімінг, аналіз, дані, розробка, база даних, система, користувач, плейлист.

Кваліфікаційна робота присвячена вирішенню проблеми користувача стрімінгового сервісу по аналізу його музичних вподобань. Мета роботи: створити інформаційну систему для аналізу музичних вподобань користувачів стрімінгових сервісів.

В першому розділі кваліфікаційної роботи описано предмет дослідження, визначені основні складові для оцінки музичного смаку користувачів, проведено визначення необхідного функціоналу кінцевого продукту на основі існуючих аналогів, та обрано оптимальні рішення для розробки кінцевого продукту.

В другому розділі кваліфікаційної роботи детально описані етапи проектування кінцевого продукту. Розглянуто принцип роботи бази даних системи та побудови веб-інтерфейсу для користувача. Також було створено інструкцію користувачу та виведено мінімальні системні вимоги для якісної роботи кінцевого продукту та проведено функціональне та структурне тестування.

У третьому розділі висвітлено значення адаптації в трудовому процесі та описано загальні вимоги безпеки з охорони праці для користувачів ПК.

ANNOTATION

Information system development for musical preferences monitoring of streaming services users// Qualification work of the educational level "Bachelor" // Kuzo Oleg // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Sciences, group ST-41 // Ternopil, 2021 // P. – 65, fig. – 22, tables – 0, chair. – 0, annexes – 0, ref. – 43.

Keywords: streaming, analysis, data, development, database, system, user, playlist.

Qualification work is devoted to solving the problem of the user of the streaming service for the analysis of his musical preferences. Purpose: to create an information system for analyzing the musical preferences of users of streaming services.

The first section of the qualification work describes the subject of research, identifies the main components for assessing the musical taste of users, determines the required functionality of the final product based on existing analogues, and selects the best solutions for the final product.

The second section of the qualification work describes in detail the stages of designing the final product. The principle of operation of the system database and construction of the web interface for the user is considered. The user manual was also created and the minimum system requirements for the quality of the final product were derived, and functional and structural testing was performed.

The third section highlights the importance of adaptation in the labor process and describes the general safety requirements for occupational safety for PC users.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) - Прикладний програмний інтерфейс.

HTML (англ. Hyper Text Markup Language) - Мова тегів, яка використовується для розмітки веб-сторінок.

SQL (англ. Structured Query Language) - Декларативна мова програмування для взаємодії користувача з базами даних.

CSS (англ. Cascading Style Sheets) - Спеціальна мова стилю сторінок.

IDE (англ. Integrated Development Environment) - Комплексне програмне рішення для розробки програмного забезпечення.

WSGI (англ. Web Server Gateway Interface) - Стандарт взаємодії між Python-програмою.

CI (англ. Continuous Integration) - Практика розробки програмного забезпечення.

ORM (англ. Object-relational mapping) - Технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування.

BSD (англ. Berkeley Software Distribution) - Операційні системи сімейства UNIX.

БД - База даних.

ОС – Операційна система.

ООП – Об’єктно-орієнтоване програмування.

ЗМІСТ

ВСТУП

Актуальність теми. Музичний потоковий контент стає настільки популярним, що забирає все більше слухачів у радіо-станцій. Такі слова як "стрім" міцно увійшли в словниковий запас сучасної людини.

Музичні стрімінгові сервіси зазвичай дають користувачеві доступ до виконавців і їхніх музичних пісень. Але стрімінговий сервіс не дає користувачеві можливість аналізу та виведення статистики прослуханого.

Разом з цим, у багатьох користувачів, що проводять час за прослуховуванням музики з’явились нові потреби. Тепер стрімінгові сервіси збирають інформацію про користувача і поширюють її за допомогою своїх прикладних програмних інтерфейсів. Це дає можливість для сторонніх розробників розширювати функції потокових сервісів.

Мета і задачі дослідження. Метою є створення інформаційної системи та побудови інструментальної системи розробки та відлагодження готового програмного забезпечення. Були сформовані основні задачі проекту:

- провести системний аналіз конкурентів та імплементувати функції, які дозволять системі бути конкурентною на ринку;
- оглянути літературу, яка допоможе написати програмний код ефективно;
- розробити та відобразити алгоритми роботи компонентів системи;
- розробити користувацький інтерфейс, який буде максимально гнучким та інтуїтивно зрозумілим;
- розробити програмне рішення інформаційної системи.

Практичне значення одержаних результатів, полягає в тому, що розроблений продукт буде мати кінцеву цінність для клієнтів, які хочуть отримувати персоніфікований аналіз своїх музичних вподобань та отримувати креативний контент, який генерує програмний код.

1 ПРЕДМЕТНЕ ДОСЛІЖЕННЯ ТА ВИДІЛЕННЯ ОСНОВНИХ АСПЕКТІВ РОБОТИ

1.1 Опис предмету дослідження

За формуванням музичного смаку людей стоїть безліч психологічних факторів. По всій планеті щодня прослуховується багато музики різних жанрів, відповідно впливаючи по різному на психологію слухачів. Цей вплив може бути різним - від розвитку когнітивних функцій до регулювання емоційного стану. Також багато для кого музика виступає можливістю самовираження та перетворення емоцій в творчість.

Інформаційна система для моніторингу стрімінгових сервісів досить затребувана, тому що часто слухач бажає краще пізнати себе та свій музичний смак, завдяки даним які ця система збере та продемонструє користувачу в зручному форматі [1].

Стрімінгові сервіси зазвичай дають користувачу доступ до музичного матеріалу виконавців. Але переважно ці сервіси не дають користувачеві можливість аналізувати прослухане по певних відрізках часу, по тому, який стиль вона чи він слухають, наприклад в вечірню пору або на ранковій пробіжці, або наприклад дізнатися який артист є твоїм улюбленим.

Використовуючи прикладний програмний інтерфейс стрімінгового сервісу можна зібрати дані, які допоможуть розробити інформаційну систему:

- «зафоловлені» артисти;
- додані альбоми;
- додані пісні;
- дані про час та жанр музики яку слухав користувач;
- останні прослухані;
- плейлисти;
- місцезнаходження слухача;

Це опис тільки дещиці можливостей, що надаються користувачу музичним сервісом.

Звичайно, аналіз і збір інформації за всіма критеріями, які мав би бажання зібрати користувач не завжди можливо. Як приклад, дані про музику яку слухали батьки, адже ця інформація ніяк не оцифрована, крім того в ті часи навіть згадок про музичні сервіси не було.

Отже, інформаційна система для аналізу музичних смаків поділяється на такі складові:

- складова самовираження;
- креативна складова;
- соціальна складова;
- хронологічна складова.

Складова самовираження повинна стати найзатребуванішим розділом системи, оскільки в ньому користувач зможе побачити свого топ артиста, свій найулюбленіший жанр та найбільш прослуховувану пісню. Також кожного разу при вході в БД зберігатиметься «зліпок» з даних, які «прилітають» в час виходу. Таким способом, коли в клієнта буде потреба знову побачити аналіз, з'явиться можливість порівняння в розрізі минулих відвідувань.

Креативна складова створює можливість формувати плейлисти базуючись на клієнтській інформаційній системі. Для прикладу, завдяки інформації про кількість бітів в пісні, можливо створити плейлист для занять спортом беручи за основу збережені пісні слухача. Також, можна створювати плейлисти рекомендаційного характеру за топом найпопулярнішого.

Соціальна складова відповідає за запис таких даних авторизованого користувача:

- улюблений жанр користувача;
- «настрій» треків, які переважно програються;
- популярність композицій;
- середній час прослуховування;
- місцезнаходження.

Що є основою соціальності? Те, що будь-який користувач має можливість дізнатися свою статистику, порівняти та поділитися нею з статистикою інших. Також таким чином формується інформація «середньої температури» по регіонах і з’являється можливість все це порівняти.

Функція хронологічної складової досить очевидна. Людина повинна мати змогу виставляти часові рамки, в яких можна подивитися прослухане за цей період. Використовуючи цю функцію просто визначити, що було прослухано в процесі тих чи інакших подій.

Звичайно, все вищезгадане може поєднуватися багатьма різними способами. Як приклад, можливо дізнатися, улюблених артистів клієнта, та порівняти результат з користувачами з його країни, або всього світу.

1.2 Порівняльний аналіз існуючих інструментів для моніторингу стрімінгових сервісів

Плейлист сервіс Sort Your Music

Це сервіс для менеджменту плейлистів за популярністю, танцювальністю, датою релізу, ритмом та іншими ознаками.

Список жанрових атрибутів, для сортування своїх треків, досить об’ємний, і тільки розширюється.

Ресурс надає можливість формувати плейлисти за такими характеристиками:

- ритм пісні;
- енергійність;
- тривалість;
- настрій композицій;
- випадкове число – для перемішування треків;
- акустичність;

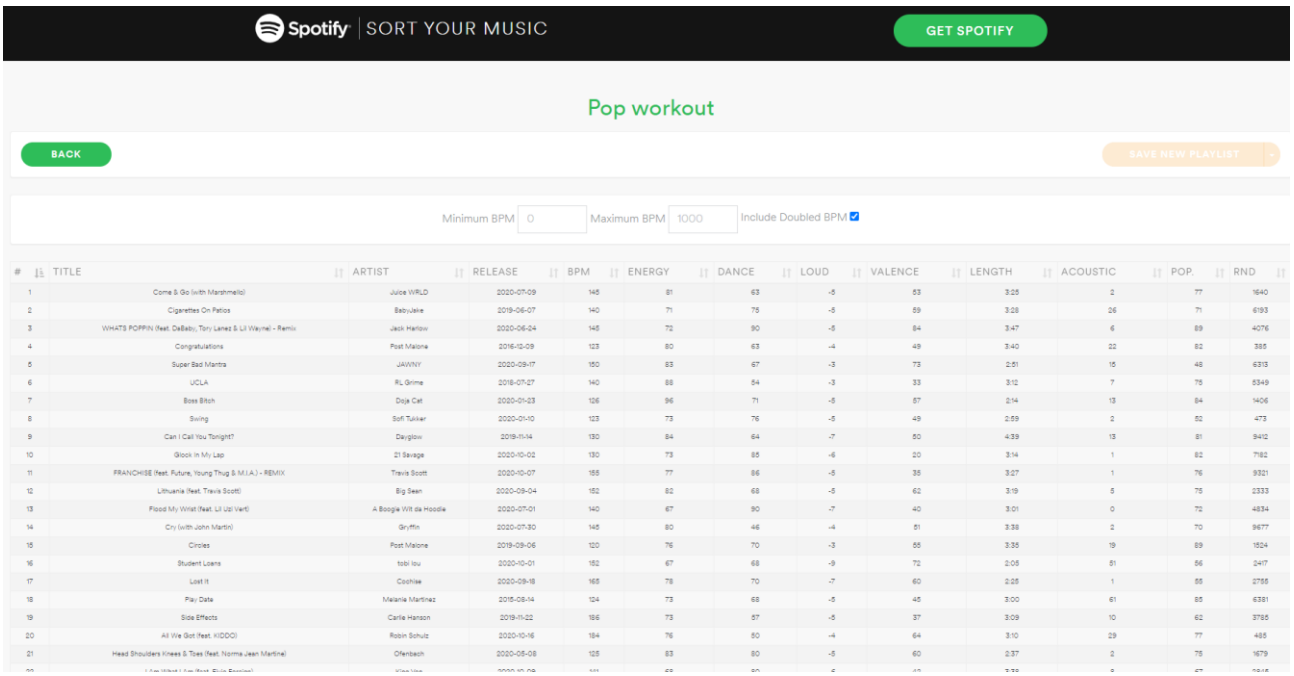


Рисунок 1.1 Скріншот сформованого в сервісі Sort Your Music плейлиста

Переваги та недоліки Sort Your Music:

- можливість випадкового відтворення плейлистів;
- широкий спектр фільтрів для сортування пісень;
- функціонал, обмежений лише списками відтворень;
- не адаптований під мобільні пристрої;
- доступний однією мовою.

Аналітична система Obscurify

Ця аналітична система є різнобічним інструментом.

Особливостями є світові рейтинги популярності авторів у композиціях користувача, що отримують бал Obscurity за певним алгоритмом, який можна порівняти з іншими користувачами сервісу у регіоні слухача, щоб визначити, наскільки особливим є його музичний смак.

Obscurify має функціонал створення списків відтворення, з можливістю зразу додавати їх до вашої бібліотеки у Spotify, включно даними історії прослуховувань , які зберігаються коли ви відвідуєте Obscurify цомісячно.

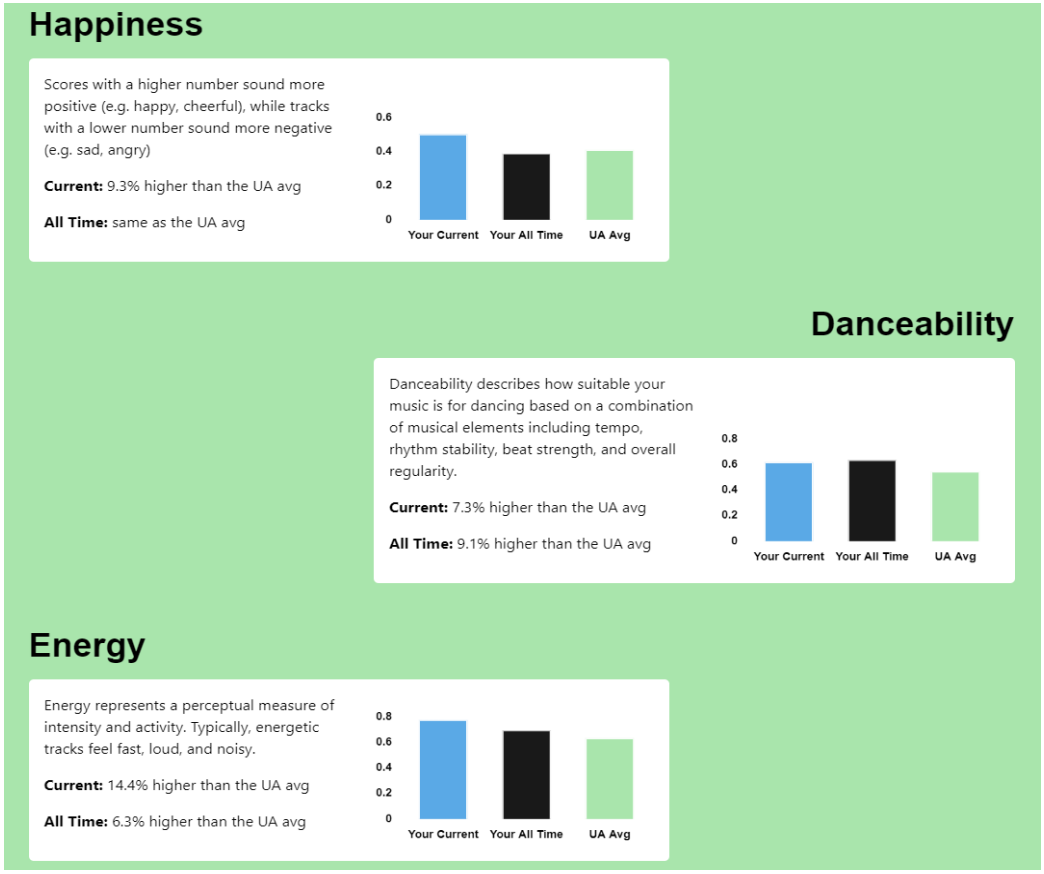


Рисунок 1.2 Частина аналітичних даних, які надає Obscurify

Переваги та недоліки Obscurify:

- інтуїтивний інтерфейс (див. рисунок 1.2);
- порівняння в межах регіону(країни);
- функціонал створення плейлистів;
- не локалізований;
- підтримка лише сервісу Spotify.

Онлайн-ресурс Spotify.me

Дана система доступна на офіційному веб-сайті збору даних Spotify. Вона проаналізовує звичні для користувача години прослуховування.

Проте, Spotify.me обмежена лише аналізом даних щодо часу прослуховування, але не враховує виконавців, жанри, настрій тощо.

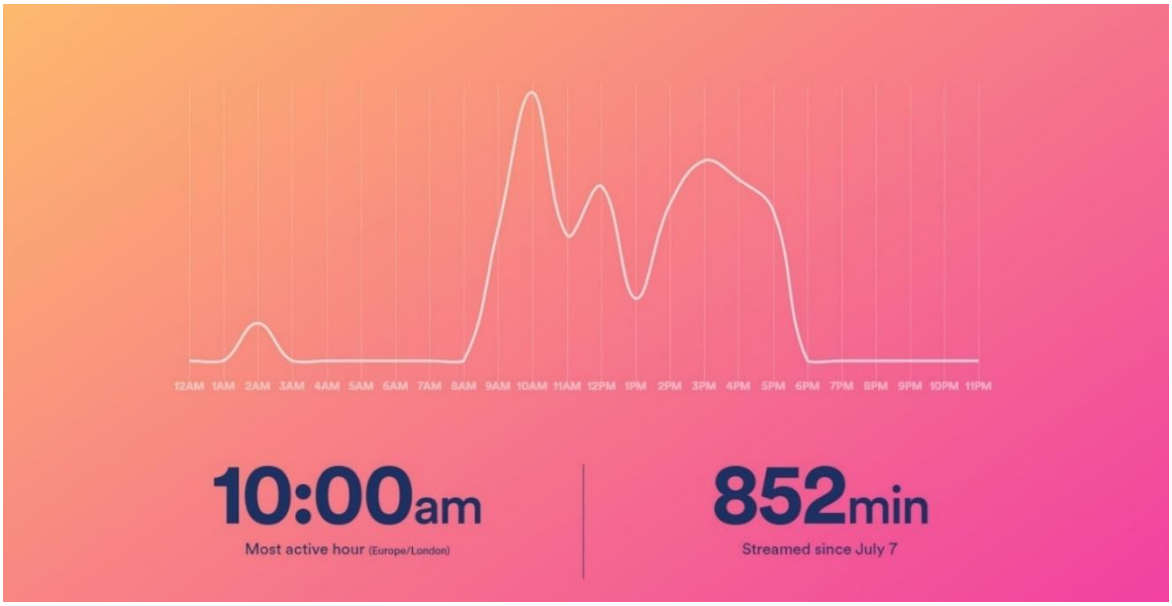


Рисунок 1.3 Графік добової активності на Spotify.me

Переваги та недоліки Spotify.me :

- інформація надається лише про час активності (див. рисунок 1.3);
- розробниками системи є автори сервісу;
- ресурс доступний декількома мовами;
- нереалізована соціальна складова;
- підримка лише одного сервісу.

Web-додаток Replayify

Цей помічник розкриває безліч характеристик і має, мабуть, найширший функціонал з-поміж конкурентів на ринку.

Тут є вкладки з даними про улюблених виконавців, рейтинг пісень і списком нещодавно відтворених пісень.

Буквально в декілька кліків можна створити плейлист завдяки різним алгоритмам підбору композицій. Наприклад, система виділяє п’ять найпопулярніших треків з вибірки виконавців, що стосуються двадцяти найпрослуховуваніших (див. рисунок 1.4).

Вкладка "Нещодавнє" дозволяє бачити історію програних композицій за найближчий час.

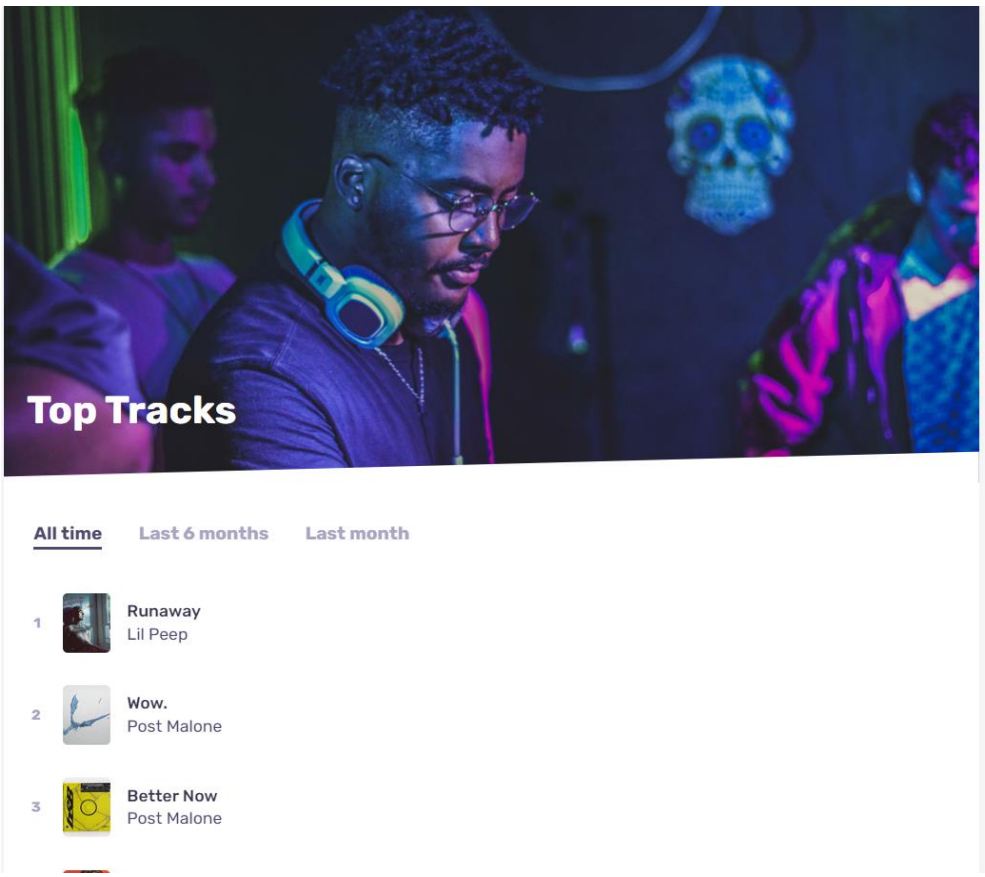


Рисунок 1.4 Домашня сторінка Replayify

Також, формується топ прослуховування за останній місяць, півроку, або за весь час.

Переваги та недоліки Replayify:

- дуже широкий функціонал;
- красивий дизайн інтерфейсу;
- реалізовано креативну складову;
- не реалізовано порівняння з рештою користувачів;
- немає підтримки різних платформ.

Інформаційна система Skiley

Skiley - це система, яка збирає дані про звички користувача, налаштовує плейлист за декількома параметрами, підбирає потенційно цікаві для слухача пісні, а також надає доступ до такої інформації як тексти та переклади пісень. Вона себе позиціонує, як багатофункціональна платформа для задовільнення

більшості потреб пересічного користувача. Система поширення безкоштовна, проте є можливість благодійної подяки, як одноразової так і регулярної.

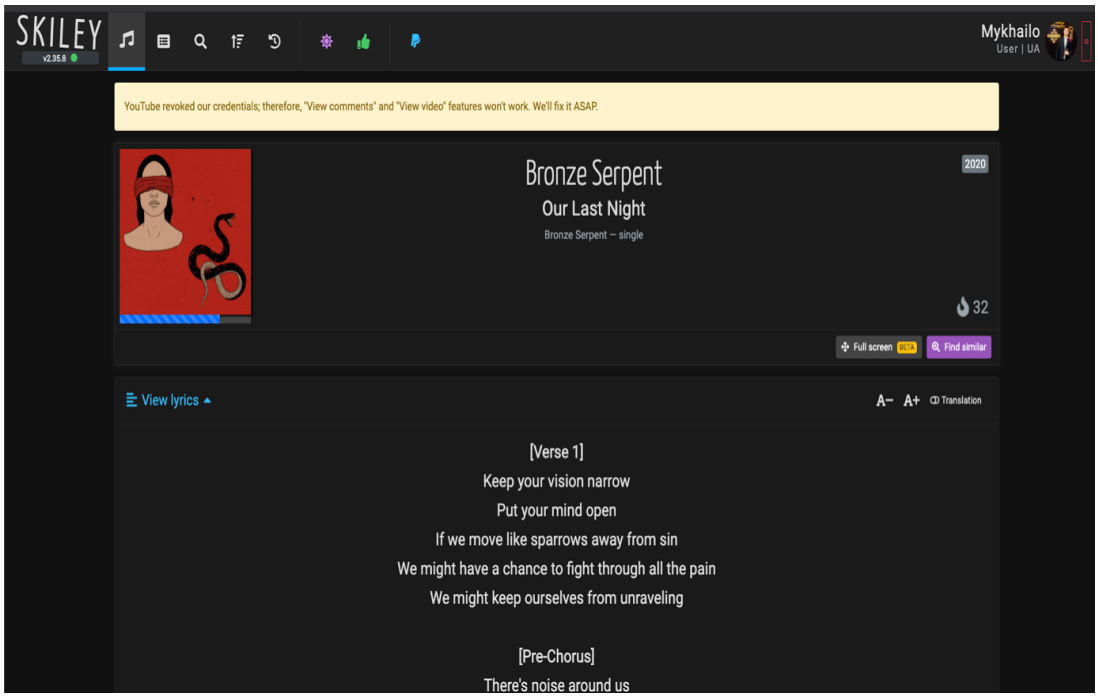


Рисунок 1.5 Перегляд тексту пісні на Skiley

Переваги та недоліки Skiley:

- перегляд тексту пісні (див. рисунок 1.5);
- створення плейлиста на основі вже існуючого;
- підтримка темної теми інтерфейсу;
- немає сортування за жанровим забарвленням композицій;
- відсутня можливість переходу на відеокліп за посиланням, або в самому інтерфейсі.

Web-додаток Run BPM

Додаток створений для фільтрування композицій за музичними характеристиками. Виділяється мінімалістичним, проте фунцкціональним інтерфейсом.

Ця програма, використовує внутрішні можливості аналізування треків у Spotify щоб створювати власні плейлисти в певному ритмі. Сервіс особливо популярний серед любителів заняття спортом, оскільки можна підібрати список

відтворення під ритм биття серця. Крім того, є можливість фільтрації за емоційним забарвленням композицій.

Рисунок 1.6 Дизайн додатку Run BPM

Переваги та недоліки Run BPM:

- інтуїтивний інтерфейс (див. рисунок 1.6);
- обмежений лише маніпуляціями з списками відтворень;
- доступний лише однією мовою;
- немає можливості порівняння з іншими користувачами;
- відсутня можливість самовираження.

1.3 Визначення оптимальних рішень для розробки кінцевого продукту

Основним завданням цієї роботи є розробка інформаційної системи для моніторингу стрімінгових сервісів користувачів.

Для реалізації цієї задачі потрібно обрати найоптимальнішу мову програмування. Вибір було зроблено на користь Python – завдяки універсальності цієї мови можна досить просто реалізувати всі задумки. Переваги є нескладний для освоєння синтаксис, неявна типізація та інтерпретованість. Зважаючи на масову доступність у браузерах також було обрано JavaScript для реалізації користувацького інтерфейсу.

Після цього прийшов час вибору середовищв розробки. Враховуючи обрані мови вибір впав на операційну систему Linux. Основними факторами вибору стали безкоштовне розповсюдження, UNIX залежність Python, зручність розробки та підтримки на цій системі. Ці фактори та довершеність архітектури роблять Linux лідером на ринку. Крім того на Linux доступний дуже обширний вибір різних інструментів для розробки.

Обрані мови програмування – Python, JavaScript

Python – мова, яка за три десятки років свого існування стала дуже широко застосованою. Python поширюється за замовчування з більшістю популярними операційними системами, крупні веб-сайти використовують його для виконання своїх задач, широко використовуєть вченими у різних галузях. Зважаючи на його популярність над його розробкою та вдосконаленням працює багато людей. Завдяки цьому робота з використанням Python проходить швидко та гнучко.

Більшість розробників Python не мають можливості слідкувати за всіма особливостями та нововведеннями, адже робота ведеться різними частинами ком’юніті. Як наслідок не всі ці особливості стають досить відомими і не стають широко використовуваними, хоч і заслуговують на це.

Python з open source кодом зручний навіть для комерційного використання, тому що розроблений під OSI ліцензії.

Загалом перевагами Python можна назвати:

- інтегративність компонентів;
- продуктивність розробки;
- підтримка великої кількості бібліотек;
- висока якість програмного забезпечення;
- портативність програмного забезпечення.

Якість вихідного продукту, узгодженість, увага до читабельності вирізняє Python серед решти мов програмування. Код у Python був спроектований для читабельності, тож це робить його зручним для подальшої підтримки та неодноразового використання в порівнянні з іншими мовами. Його однорідність спрощує розуміння коду написаного не вами. Також Python підтримує просунуті механізми для повторних використань програмного забезпечення, такого, як наприклад : програмування функцій та об'єктно-орієнтоване програмування.

Python суттєво продуктивніший в порівнянні з статичними, або ж компільованими мовами, такими як Java, C та C++. Він не вимагає такого налагоджування та облуговування під час розробки, та й об’єм набраного тексту значно менший. Програми на Python запускаються зразу, без довгих кроків зв’язування та компіляції, які необхідні для інакших інструментів, що ще більше пришвидшує процес написання.

Програми написані на Python не потребують змін для роботи на основних платформах. Для прикладу, при перенесенні коду між Windows та Linux, в більшості випадків потрібно лише скопіювати код сценарію. Крім цього, Python надає можливість написання переносних графічних користувацьких інтерфейсів, веб-систем та програм для доступу до баз даних тощо. Також портативність інтерфейсів операційної системи, обробки каталогів, запуску програм на дуже високому рівні.

Python поширюється з об’ємною колекцією стандартних бібліотек . Вони підтримують на рівні програми масив завдань, таких як відповідність мережевих сценаріїв до шаблону тексту . Також, він має можливість розширення власними бібліотеками, або ж вже існуючим програмним забезпеченням[2]. Python

пропонує інструменти для створення сайтів, розробки ігор доступу до послідовних портів, чисельного програмування тощо. Для прикладу візьмемо NumPy, розширення яке було розроблене як безкоштовний і функціональніший аналог Matlab [3] .

JavaScript - мова програмування для Web. Більша частина веб-сайтів створенні з використанням JavaScript, всі актуальні браузери - для персональних комп'ютерів, приставок, смартфонів - включають інтерпретатор JavaScript. Це робить його найширокозастосованішою мовою програмування за весь час. JavaScript в першій трійці технологій , в яких має бути обізнаний кожен веб-розробник, разом з мовою стилів CSS та мовою розмітки HTML.

На даний момент JavaScript можна назвати «безпечною» мовою програмування широкого призначення. Він не працює низькорівнево з пам’яттю, адже зразу при розробці орієнтація була на використання в браузерах, де в цьому немає потреби [4].

Решта можливостей залежить від середовища, в якому працюють з JavaScript. При стандартному сценарії використання в браузері JavaScript має функціонал для проведення маніпуляцій з сторінкою та взаємодії з сервером:

- операції з HTML-тегами (створення, видалення, заміна існуючих, зміна стилів, приховування та демонстрація елементів і т.п);
- реакції на дії користувача, обробка кліків миші, натисків на клавіатуру, пересування курсора тощо;
- надсилати запити на сервер і завантажувати дані без оновлення сторінки (технологія «AJAX»);
- встановлення отриманих cookie, виведення повідомлень, надсилання запитів на дані [5].

Використана операційна система – Linux

Linux - це UNIX-подібна операційна система, складається з програмного забезпечення, яке управляє вашим комп'ютером і дозволяє запускати програми по ньому. Основними функціями Linux, як і будь-якої комп’ютерної операційної системи такі:

- можливість мати користувацький інтерфейс;
- інструментарій для програмування;
- забезпечення користувацьким доступом та автентифікацією;
- керування процесами та пам’ятю;
- контроль файловою системою;
- виявлення та підготовка обладнання;
- забезпечення адміністративними інструментами.

Сучасні системи на базі Linux виходять за рамки можливостей перших систем UNIX (які є базою для Linux). Додатковий функціонал Linux, який в більшості використовується наступний:

- кластеризація;
- віртуалізація;
- обчислення в хмарі;
- система реального часу;
- спеціалізовані сховища.

На Linux можна налаштувати роботу в кластерах, таким чином, що декілька систем виглядатимуть як єдина зовні. Служби можна налаштувати на передачу туди-назад між вузлами кластера, паралельно відображаючись тим, хто користується службою, яка запускається, без затримок.

Для ефективнішого керування обчислювальними ресурсами можливий режим роботи Linux, як хоста віртуалізації. Це дасть можливість запуску інших систем Linux, Windows, BSD або ж інакших операційних систем, як віртуальними гостями. Для створення таких хостів в Linux задіюються такі технології як Xen та KVM.

Для керування об’ємними середовищами віртуалізації використовуються різні обчислювальні платформи. Прикладами є oVirt, OpenStack, які можуть водночас керувати декількома хостами віртуалізації, мережевим сховищем, аутентифікацією користувачів та системи, віртуальними мережами, віртуальними гостями. Для управління контейнерними програмами, можуть використовувати проекти по типу Kubernetes [6].

Застосування Docker контейнерів

Docker - це інструментарій який застосовується для інкапсуляції програм та середовищ в контейнери. Коли на одній машині запущено декілька версій таких контейнерів, вони ніби працюють на окремих машинах.

Docker поширюється як відкрите програмне забезпечення, яке оптимізоване для роботи Linux у контейнерах, та інакших доступних open-source компонентів, використовуючи які можлива побудова складної системи. Це наслідок прогресу технологій хостингу протягом останнього десятиліття. Вимогами для цієї прогресії стали безпека, надійність та масштабованість.

Інструментарій Docker дозволяє раціональне використання підтримки віртуалізації на рівні операційної системи , для можливості виконання декількох контейнерів одночасно. За замовчування контейнери ізольовані від машини-хоста, і по суті є екземпляром образу контейнера [7].

Контейнери розділяють ядро з Linux і хостом, тому немає потреби встановлення цілих операційних систем всередині контейнера, як у випадку з віртуальними машинами. Керує ними Docker демон, який опрацьовує управління ресурсами та контейнерами які використовуються, а також образами, мережею, томами.

Ще однією різницею між контейнерами та віртуальними машинами є спільне використання ресурсів з хостом, адже другим потрібні дублікати ресурсів. Для прикладу, ідентичні контейнери використовують оперативну пам'ять хоста, тоді як віртуальні машина використовує блок, налаштований перед запуском. У випадку потреби є можливість обмеження ресурсів контейнера (пам’ять, підкачку, процесор), за замовчанням ніяких обмежень немає.

Обрані фреймворки – Flask, Vue.js

Фреймворк - це програмний каркас, який облегшує розробку, забезпечуючи загальний функціонал. При потребі користувач може збільшувати, або змінювати функціональність написанням додаткового коду.

Для написання back-end частини на Python вибір зупинився на Flask.

Flask – можна назвати «мікрофреймворком», адже для його використання не потрібно особливих засобів чи бібліотек. Проте він має широку підтримку різних розширень, а невеликий об'єм його вихідного коду дозволяє швидко розібратися в його роботі.

Особливістю, яка його виділяє є якраз ця простота, адже при потребі є можливість надбудувати потрібні функції розширеннями, але це не спричиняє так званої «боротьби з фреймворком», ситуації при якій обране вами рішення не підтримується фреймворком.

Flask з самого початку розроблявся як фреймворк, який би забезпечував лише основні послуги, а забезпечення для розширень могли додати будь-які. Це забезпечує стабільну роботу стеків без просядів, адже виконується лише те що вам потрібно і нічого лишнього не задіюється.

У Flask відсутня нативна підтримка для роботи з базами даних, автентифікації, або перевірки форм. Але, знову ж таки завдяки побудові, всі ці недоліки виправляються додаванням потрібних розширень. користувачів чи інших завдань високого рівня [20].

Основою цього мікрофреймворку є дві базові залежності:

- WSGI, підсистема маршрутизації та налаштування шлюзу веб-сервера від Werkzeug;
- шаблони Jinja2.

Ці залежності були закладені розробником Flask, Арміном Ронакером [8].

Для написання front-end частини, було обрано фреймворк Vue.js.

Vue.js – фреймворк, який використовує шаблон архітектури MVVM (Model-View-ViewModel). Безсумнівними перевагами цього фреймворку є робота лише на рівні представлення, що дозволяє просто інтегровувати Vue з проектами та бібліотеками, та його функціональності досить для створення потужних веб-додатків.

Для прикладу, маючи готовий проект, який був написаний на інакшому фреймворці JS, можна просто інтегрувати Vue для створення інтерфейсу

користувача, адже обробляти він буле лише його вигляд, а обробкою даних і візуалізацією на стороні сервера може займатися інший фреймворк[9].

При простоті, доступності, порівняно невисокому порогу входження, Vue досить швидко набрав популярність, в особливості в невеликих проектах та веб-дотатаках, які складаються з одної HTML сторінки.

До того ж, завдяки компактності, швидкості роботи та невибагливості в порівнянні з іншими програмними каркасами, такими як наприклад Angular або React, для виконання нашої задачі Vue.js підходить більше [10].

Система керування базами данних – PostgreSQL.

PostgreSQL - це система керування базами даних SQL, яка розробляється глобальною групою розробників. Це виступає однією з головних переваг PostgreSQL, адже поширюється вона відкрито, відповідно її можливо безкоштовно встановити та використовувати. Крім цього, PostgreSQL база даних, яка майже не потребує постійної підтримки та розрахована на довготривале використання. В результаті PostgreSQL гарантує мінімальні витрати на володіння.

Враховуючи свою тридцятилітню історію, та величезну армію авторів та розробників, PostgreSQL прославилася солідним набором вдосконалених функцій.

Переваги та функції PostgreSQL:

- відповідає всім стандартам SQL;
- клієнт-серверна архітектура;
- має можливість конфігурації та додавання розширень для широкого спектру різних додатків;
- широкий функціонал налаштування та чудова масштабованість та продуктивність;
- підтримує різні типи моделей даних (реляційні, постреляційні, JSON ,XML та ключ / значення);

PostgreSQL може досягати понад мільйон читань в секунду на чотирисокетному сервері та орієнтована на значно більше ніж тридцять тисяч

транзакцій запису в секунду з повною довговічністю. Вдосконалюючи обладнання можливо досягнути навість вищих рівнів продуктивності [21].

PostgreSQL підтримує широкий діапазон типів індексів та даних загальнодоступних серверів баз даних.

Поєданання усіх цих можливостей PostgreSQL дозволяє виконувати складні аналітичні та пошукові завдання, а також підтримувати змішаний режим роботи на транзакційних систем. В нашому випадку це ключова перевага, адже для виконання запитів не буде потреби вивантажувати дані з виробничих ситем у сховища аналітичних даних[11].

В підсумку для розробки було використано такі рішення:

- Операційна система – Linux OS;
- Мови програмування – Python, JavaScript;
- Інтегровані середовища розробки– PyCharm, WebStorm;
- Фреймворки – Flask, Vue.js;
- Docker - контейнери;
- Середовища – PyCharm та WebStorm;
- Систему керування базами даних – PostgreSQl.

1.4 Концептуальна модель

Концептуальна модель - це абстрактне уявлення змодельованої системи, утвореної з елементів та причинно-наслідкових зв’язків, яка допомагає краще зрозуміти представлений предмет або область.

Концептуальне моделювання потрібне для попереднього збору даних про потенційні функції, які буде виконувати бажана система.

Вивчаючи концептуальну модель можливо зауважити, що користувач взаємодіє з компонентами тільки ззовні за допомогою відповідного інтерфейсу. Першим відвідувача зустрічатиме інтерфейс для автентифікації, далі здійснюватиметься перехід до інтерфейсу користувача, який демонструватиме дані проаналізовані на бекенді.

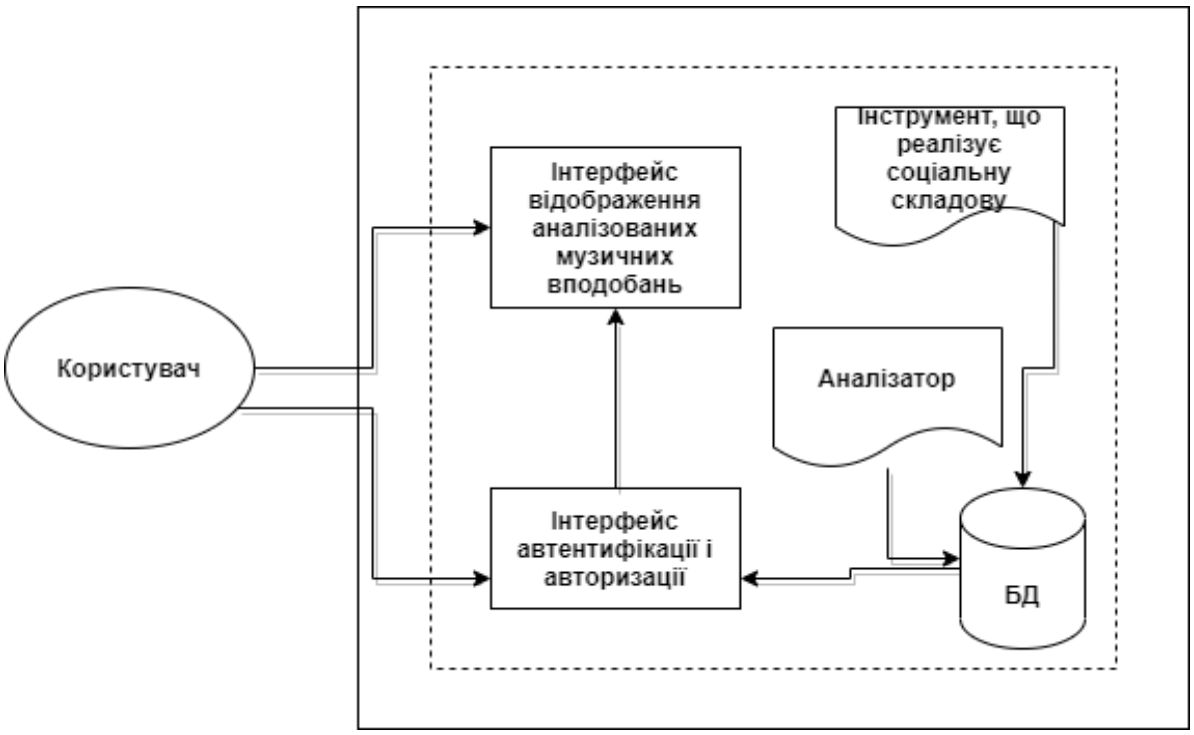


Рисунок 1.7 Концептуальна модель інформаційних потоків всередині системи

Решта компонентів є недоступними для прямого впливу користувачем. Після проходження автентифікації задіюються інструменти, які опрацьовують вхідні дані, виводять основні характеристики, та проводять запис логів, статистичних даних та інформації про користувача в базу даних для подальшої обробки. Крім цього в базі даних агрегуються дані щодо регіональних показників для підтримки масштабності.

Загалом ця модель демонструє основні функції цієї інформаційної системи:

- функцію запису (відображення змін, які впливають на систему);
- функцію інформування (демонстрація впливу змін в візуальному форматі);
- активну функцію (взаємодія з системою та вплив на неї).

1.5 Висновки до розділу

В даному розділі описаний предмет дослідження, визначені основні складові для оцінки музичного смаку користувачів, а саме:

- складова самовираження;
- креативна складова;

- соціальна складова;
- хронологічна складова.

Для визначення необхідного функціоналу кінцевого продукту проведено порівняльний аналіз існуючих інструментів для моніторингу стрімінгових сервісів, що показав на практиці всі переваги та недоліки існуючих аналогів. В результаті я отримав кінцевий список потрібних функцій та перейшов до етапу вибору програмних складових.

Мовою програмування, зважаючи на швидкість та гнучкість, була обрана Python. Серед її переваг:

- інтегративність компонентів;
- продуктивність розробки;
- підтримка великої кількості бібліотек;
- висока якість програмного забезпечення;
- портативність програмного забезпечення.

Для Web-представлення мого продукту було обрано JacaScript, через наявність потрібного для реалізації мети функціоналу для проведення маніпуляцій з сторінкою та взаємодії з сервером (операції з HTML-тегами; реакції на дії користувача; надсилання запитів на сервер і завантажування даних без оновлення сторінки; встановлення отриманих cookie, надсилання запитів на дані).

Операційною системою в ході розробки виступить Linux. Серед переваг останньої можна виділити користувацький інтерфейс, інструментарій для програмування, забезпечення користувацьким доступом та автентифікацією.

В підсумку для розробки було використано такі рішення:

- операційна система – Linux OS;
- мови програмування – Python, JavaScript;
- інтегровані середовища розробки– PyCharm, WebStorm;
- фреймворки – Flask, Vue.js;
- контейнери – Docker;
- середовища – PyCharm та WebStorm;

- систему керування базами даних – PostgreSQl.

Кінцевим етапом даного розділу виступило створення концептуальної моделі інформаційних потоків даної системи.

2 ПРОЄКТУВАННЯ ТА СТВОРЕННЯ КІНЦЕВОГО ПРОДУКТУ

2.1 Проєктування та візуалізація алгоритмів роботи компонентів в системі

Блок-схема – це вид діаграми, що візуалізує робочий процес. ЇЇ також можна описати як схематичне відображення алгоритму вирішення задачі.

Блок-схема демонструє етапи роботи у вигляді контейнерів різного типу в визначеному порядку, з'єднаних стрілками. Ця схема показує модель вирішення певного недоліку. Блок-схеми використовують при проектуванні, управлінні процесом, аналізу під час та після завершення розробки.

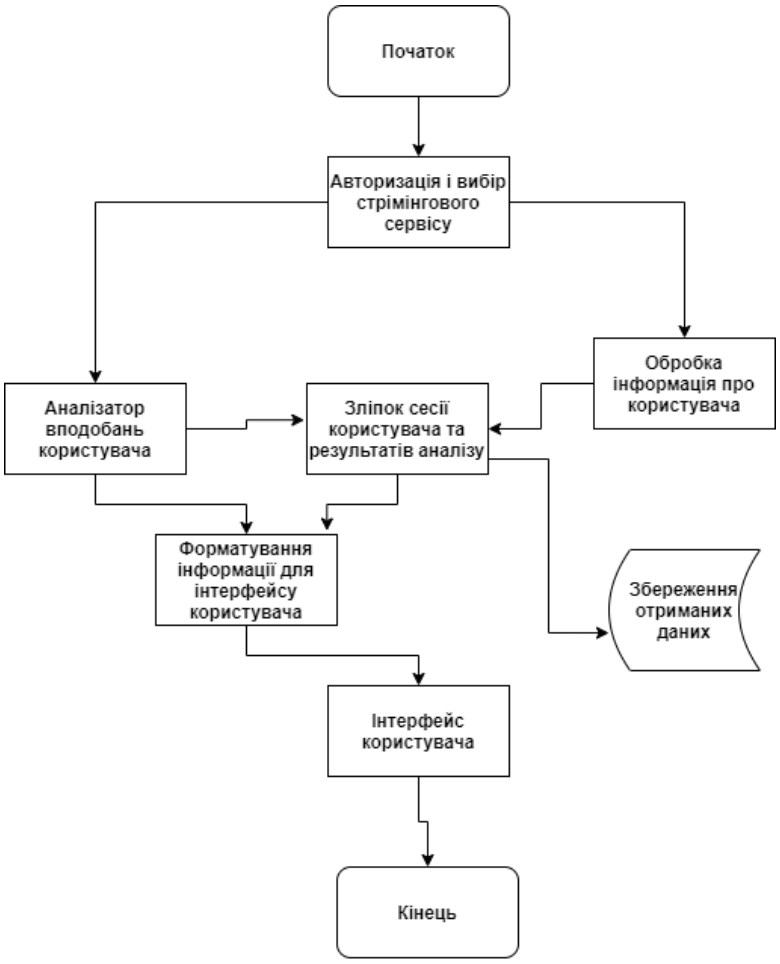


Рисунок 2.1 Блок-схема алгоритмів роботи інформаційної системи аналізу музичних вподобань.

Після авторизації програма отримає інформацію, за домомогою якої з прикладних інтерфейсів стрімінг сервісів можна отримати дані для подальшого аналізу. Далі алгоритм «ганятиме» отримані дані в трьома основними сервісами: обробки інформації про користувача, створення зліпку сесії користувача, та аналізатора вподобань. Наступним кроком проаналізована інформація форматується в комфортний для фронтенду формат, та виводитися в інтерфейс користувача (див. рисунок 2.1).

Діаграма класів — один з типів статичних структурних діаграм, яка демонструє структуру, взаємозв'язки, класи та атрибути об’єктів системи.

Діаграма класів виступає основою при побудові об’єктно-орієнтованого моделювання. ЇЇ використовують для моделювання загального концепту побудови програми, та для детального моделювання трансформаціїї моделей у програмний код. Класи та взаємодії елементів можуть представлятися в ній як програмовані, так і основні.

На діаграмі класи поділені комірками, які містять три секції:

- верхня секція з назвою класу;
- середня секція з атрибутами класу;
- нижня секція містить операції, які виконуються класом.

Проектуючи систему ряди класів ідентифікуються та групуються в схему класів, за допомогою якої визначаються їх статичні зв’язки. Моделюючи їх детальніше, часто їх розділяють ще на ряд підкласів.

Діаграми класів вказують на зв’язки між конкретними інтерфейсами максимально деталізовано. До того ж розбираючи ці діаграми можливо зрозуміти публічність, захищеність або приватність даного методу чи атрибуту.

Точкою входу в дану програму є клас «User», збираючи інформацію після авторизації. До його атрибутів відносяться ідентифікатор користувача, країна походження і ім’я користувача.

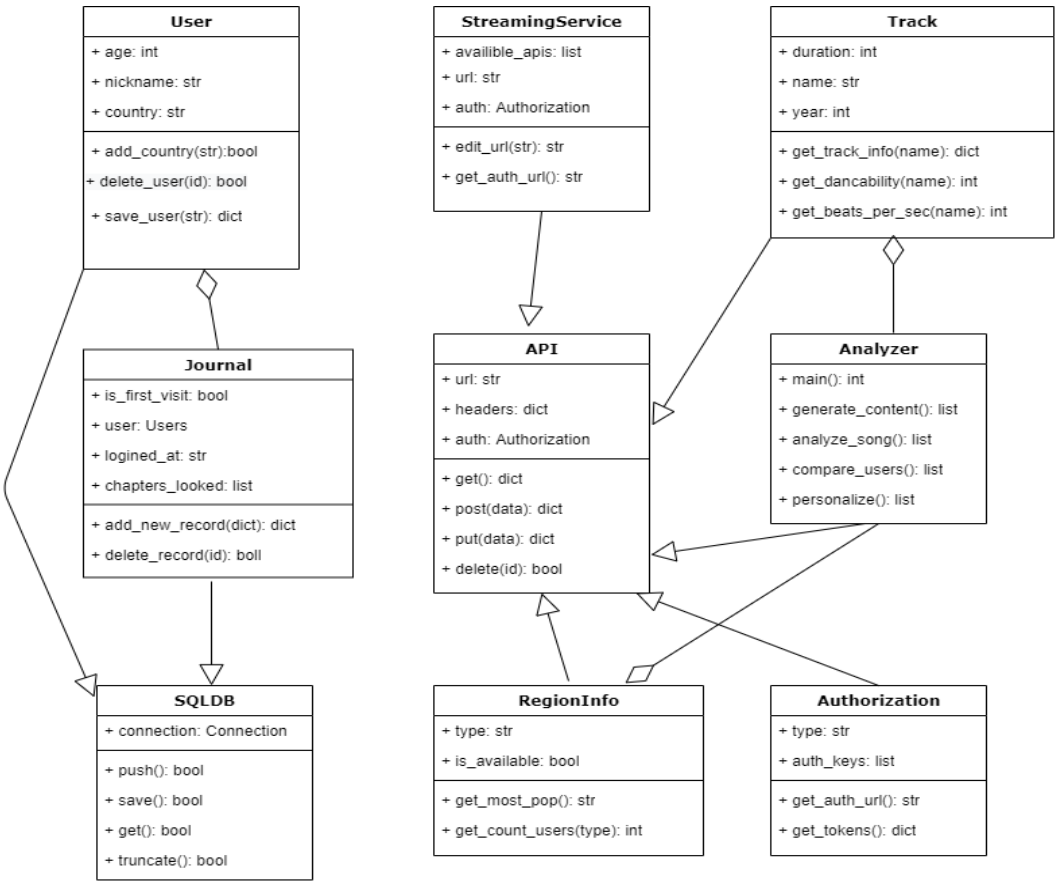


Рисунок 2.2 Діаграма класів програмного забезпечення

Для цього було реалізовано дані методи:

- add_country() – додає або змінює країну користувача;
- delete_user() – видаляє користувача з бази даних;
- save_user() – зберігає користувача до бази даних.

Наступний клас Analyzer, реалізований за шаблоном абстрактної фабрики.

Його задача полягає в інкапсуляції класів, таких як Track та RegionInfo, та виклик їх з власних методів, для отримання необхідних даних. Приклади реалізованих методів:

- generate_content() - генерація контенту для персоніфікованого користувача;
- analyze_song() - дістає всю інформацію по заданому треку, задіюючи клас Track;

- compare_users() – задіює класи RegionInfo та User для порівняння користувачів різних регіонів.

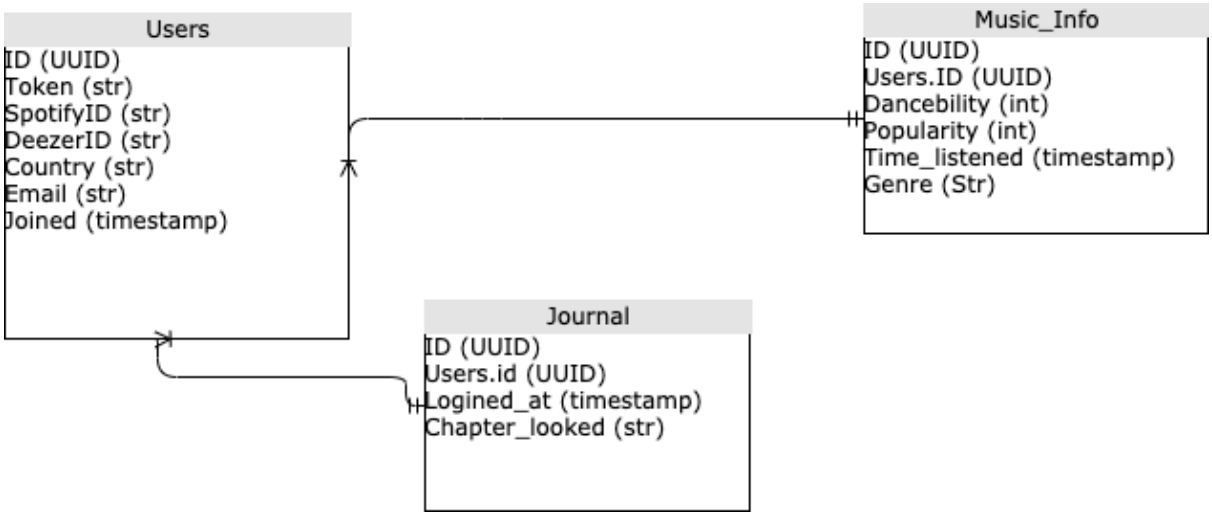


Рисунок 2.3 Структура бази даних.

На рисунку 2.3 відображено схему бази даних для зберігання даних про користувача та файлів з агрегованими даними.

Побудова блоку Users:

- ID (UUID) – персональний ID користувача;
- Token (str) – токен доступу.
- SpotifyID/DeezerID (str) – ідентифікатор користувача на сервісі Spotify/Deezer;
- Country (str) – країна користувача;
- Email (str) – користувацька пошта;
- Joined (datestamp) – дата додавання користувача до інформаційної системи.
- Побудова блоку Music_Info:
- ID (UUID) – унікальний ідентифікатор музичних даних;
- Users.ID (UUID) – персональний ID користувача;
- Dancebility (int) – значення танцювальності музики;
- Popularity (int) – значення популярності музики;
- Time_listened (timestamp) – час загального прослуховання;
- Genre (str) – жанр.

- Побудова блоку Journal:
- ID (UUID) – унікальний ідентифікатор жанру;
- Users.ID (UUID) – персональний ID користувача;
- Logined_at (timestamp) – час логіну користувача в систему;
- Chapter_looked (int) – відвідані розділи системи.

2.2 Створення програмно-апаратного середовища

Діаграма компонентів

Компоненти - це досить розмита концепція в UML, оскільки як компоненти так і використовують моделюючи те саме.

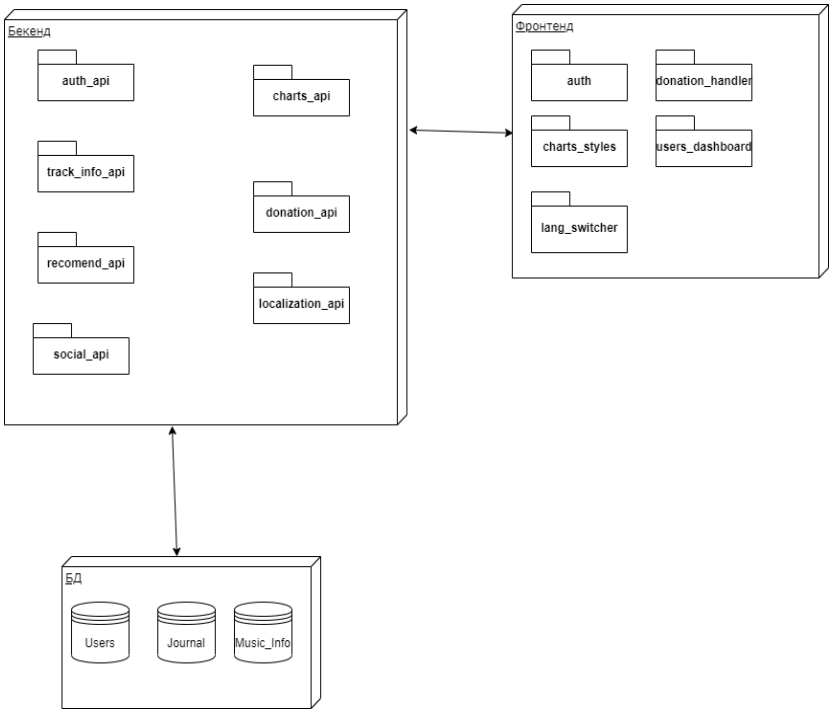


Рисунок 2.4 Діаграма компонентів системи.

Відповідно до офіційних специфікацій UML, компонент є модульною частиної системи, яка інкапсулює вміст, а її прояв є можливість заміняти власному середовищі. Компонент в свою чергу, поводиться в залежності від наданих інтерфейсів. Як наслідок, компоненти служать типом, у відповідності до наданих інтерфейсів [12].

Отож, інформаційна система для моніторингу стрімінгових сервісів розділена на такі групи компонентів:

Основний компонент:

- auth_api – основний модуль, після звернення до якого відбувається вхід по програми та вибір стрімінгового сервісу для аналізу.

Додаткові компоненти:

- charts_api –інтерфейс, який повертає перетворені дані аналізу музичних переваг користувача та додає їх до таблиці Music_Info;
- recommend_api – відповідає за створення плейлистів для користувача;
- social_api – отримує інформацію від класів Users та Music_info, для порівняння конкретногої людини з іншими, та передати дані до фронтенд компонентів;
- localization_api – відповідає за дані пов’язані з локалізацією інтерфейсу користувача.

UI компоненти:

- auth – початкова сторінка, яка зустрічає відвідувача і надає вибір способу авторизації;
- charts_styles – варіанти різних видів графіків та відображення проаналізованих чартів;
- users_dashboard – порівняння діючого користувача з іншими.

База даних:

- Users – інформація про користувача;
- Journal – зліпок проведеної сесії;
- Music_info – дані про переваги даного користувача.

Важливо зауважити, що усі ці компоненти імплементовані в Docker контейнерах, як і деякі сервіси, для ізоляції один від одного та простішої підтримки.

Опис використаних сторонніх бібліотек та модулів

D3.js - це бібліотека JavaScript яку застосовують для візуалізації інформації Фактично вона вважається стандартом інформаційної графіки в мережі.

D3 дозволяє зручно працювати з графікою у веб-браузері. Створючи потрібні вам фігури в програмі для роботи з графікою, зберігаєте їх в потрібному форматі. Після цього, використовуючи можливості D3 всього в декілька кроків можна опубліковувати матеріал використовуючи теги <img> на вашій веб-сторінці[13].

Окрім цього, D3 надає можливість просто та комфортно працювати з анімованою та інтерактивною графікою. Важко переоцінити інтерактивність графіки, адже як у більшості сфер, красива візуалізація грає дуже важливу роль у ефективності програмного продукту. В минулі часи для вирішення таких задач часто використовували недосконалі, не завжди ефективні та часто дороговартісні комерційні пакети .

Окрім роботи з графікою, D3 має простий для освоєння інструментарій для роботи з DOM загального призначення. При виникненні потреб маніпуляції з DOM, D3 може працювати з інакшими фреймворками та API без конфліктів.

Flask-SQLAlchemy - це інструментарій SQL для Python, який робить зручнішою роботу з базами даних при роботі у Flask. SQLAlchemy - це реляційна система БД, з підтримкою роботи декількох серверних баз даних.

Перевагами є об'єктно-реляційне відношення (ORM) високого рівня, що відкриває можливість роботи з шаблонами відображення даних, де класи можна зіставляти з базою даних відкритим способом, що спрощує взаємодію з таблицями та доступ до власних низькорівневих функціональних можливостей SQL.

Принцип роботи SQLAlchemy розглядає базу даних як механізм реляційної алгебри, а не як звичайну сукупність таблиць. Рядки можливо вибрати не тільки з таблиці, а й з об’єднань та інакших операторів вибору. Це дозволяє будь-якій з цих одиниць бути складеною у більшу структуру. Основа синтаксису виразів SQLAlchemy базується за цією концепцією.

Загальний підхід SQLAlchemy суттєво різниться з підходом більшості поширених інструментів SQL / ORM, основою для яких є так званий компліментарний підхід – замість приховування деталей реляційних зв'язків SQL та об'єктів за стіною автоматизації, процеси повністю розкриваються в серії набірних, прозорих інструментів. Бібліотека виконує процес автоматизації надлишкових завдань, а у свій час розробник контролює організацію бази даних та побудову SQL запиту.

Розробка та опис програмних модулів

В даному продукті програмні модулі будуть декомпонуватися за принципом мікросервісної архітектури. Спілкування відбуватиметься за допомогою REST API запитів, задіюючи POST, GET, DELETE та PUT методи.

Для взаємодії з інтерфейсом користувача клієнт сервіс працюватиме з веб-сторінкою побудованою на Vue.js, який буде «піднятий» на Node.js сервері. Вище згадані компоненти, фронтенд частини описані в діаграмі компонентів.

Запити користувача надсилатимуться з користувацького інтерфейсу на бекенд за через публічний прикладний програмний інтерфейс. Вже на самому сервері ці дані обробляєтимуться за допомогою фреймворку Flask, який представлений у всіх класах, що обслуговуючих інформаційну систему. Ці мікросервіси поділяються за відповідальністю на:

- автентифікаційні;
- аналітичні;
- креативні;
- фінансові;
- локалізаційні.

Дані про користувача, на рахунок країни походження, обраного стрімінгового сервісу зберігатимуться в таблиці Users.

Кінцевими точками мікросервісу аналізу, виступатиме збереження даних, про виконавців, жанри чи настрою в таблиці Music_info прив’язані до користувача. Ще в цю таблицю записується дата візиту системи, для надання користувачеві інформації щодо поередніх сеансів.

Крім того цей мікросервіс відповідає за логування інформації, створення зліпку сесії та записує ці дані до таблиці Journal.

Розробка та опис інтерфейсу користувача

Перед розробкою інтерфейсу користувача було виділено такі вимоги:

- інтуїтивність інтерфейсу;
- швидкий доступ до всіх розділів;
- підтримка різними платформами;
- візуалізація чартів у вигляді графіків;
- можливість поділитися в соцмережах;
- підтримка локалізації.

Інформаційна система підтримується всіма популярними веб-браузерами, такими як: Chrome, Microsoft Edge та Mozilla Firefox. Підтримуються також і на мобільні версії цих браузерів.

Стартова сторінка (див.рисунок 2.5) дозволяє користувачу обрати з якого стрімінгового сервісу, авторизуватися. До уваги було взято 3 найпопулярніших сервіси: Spotify, Youtube Music та Deezer. До того ж можливості їх API дозволять отримати потрібні нам дані.

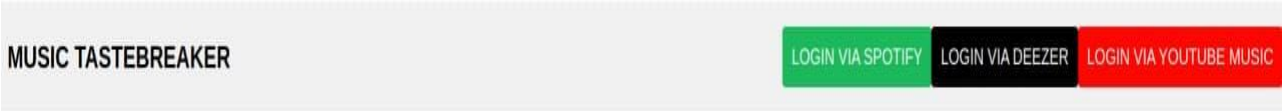


Рисунок 2.5 Стартовий екран авторизації

Коли авторизацію завершено, користувача перекидає на головну сторінку сервісу (див. рисунок 2.6), яка містить вкладки навігації та можливість зміни локалізації.

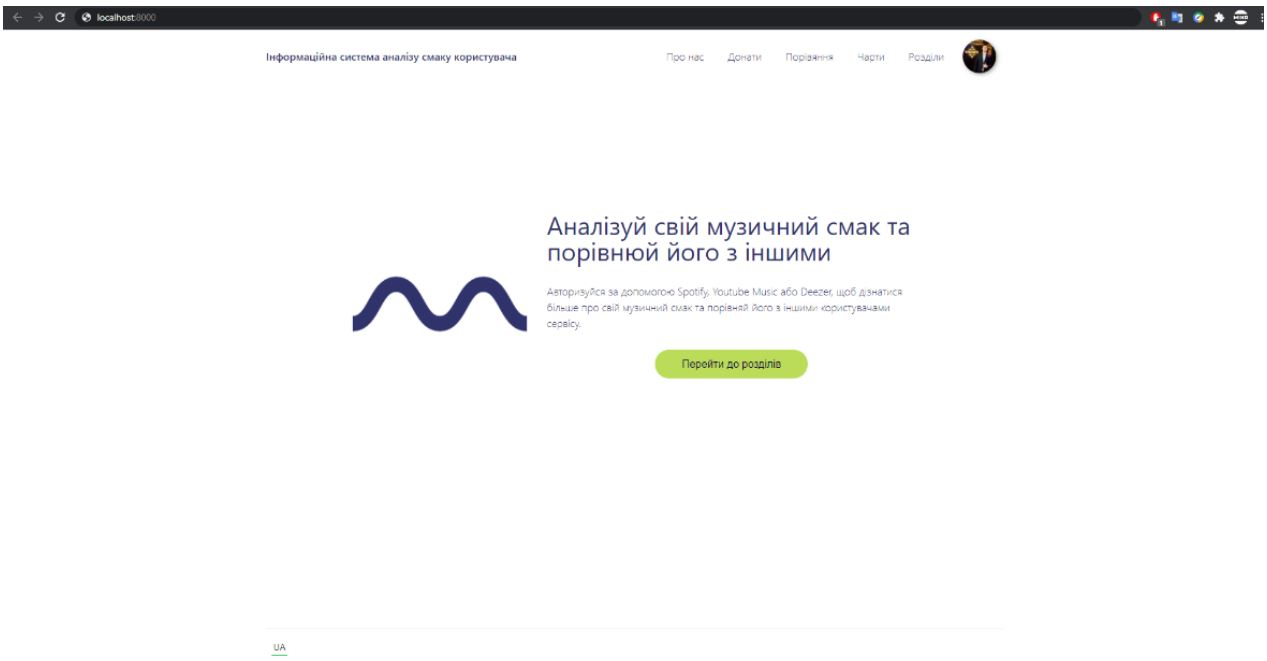


Рисунок 2.6 Головна сторінка

Далі клієнт потрапляє до селектора розділів, даної інформаційної системи. Є три основні розділи (див. рисунок 2.7):

- Чарти;
- Порівняння з іншими користувачами сервісу;
- Створення списків відтворення.
-

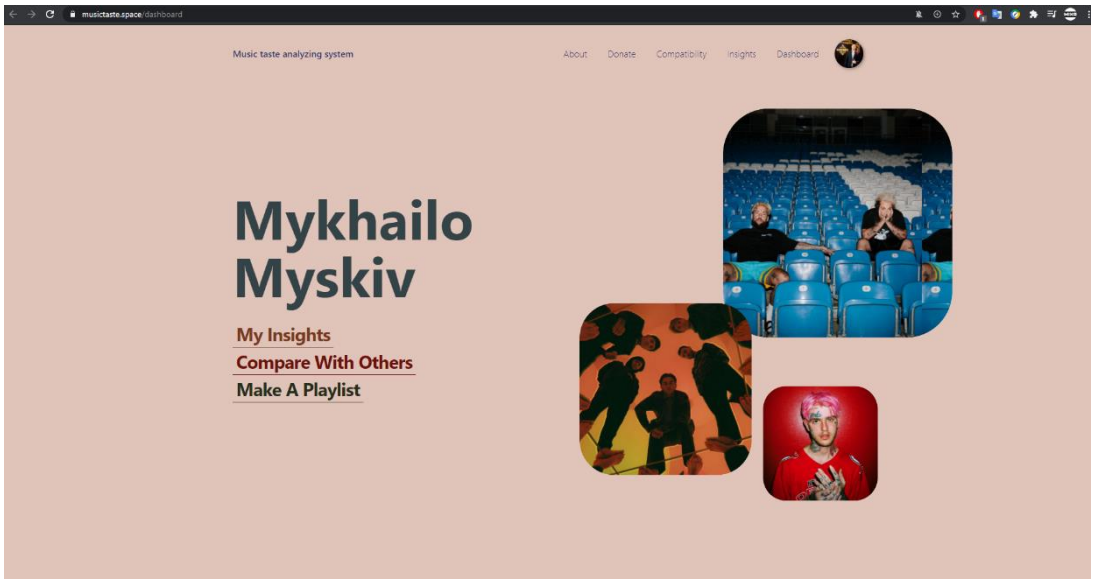


Рисунок 2.7 Сторінка з розділами

На кожні сторінці сервісу є можливість швидкої навігації у «шапці» сторінки. Окрім навігаційних вкладок, доступні посилання на інформацію про проект та можливість благодійної підтримки проекту за кнопкою «Донат».

Рисунок 2.8 «Шапка» з швидкою навігацією

На початковій сторінці вкладки «Чарти» (див. рисунок 2.9), продемонстрована візуальна інформація на рахунок улюблених виконавців за останній місяць.

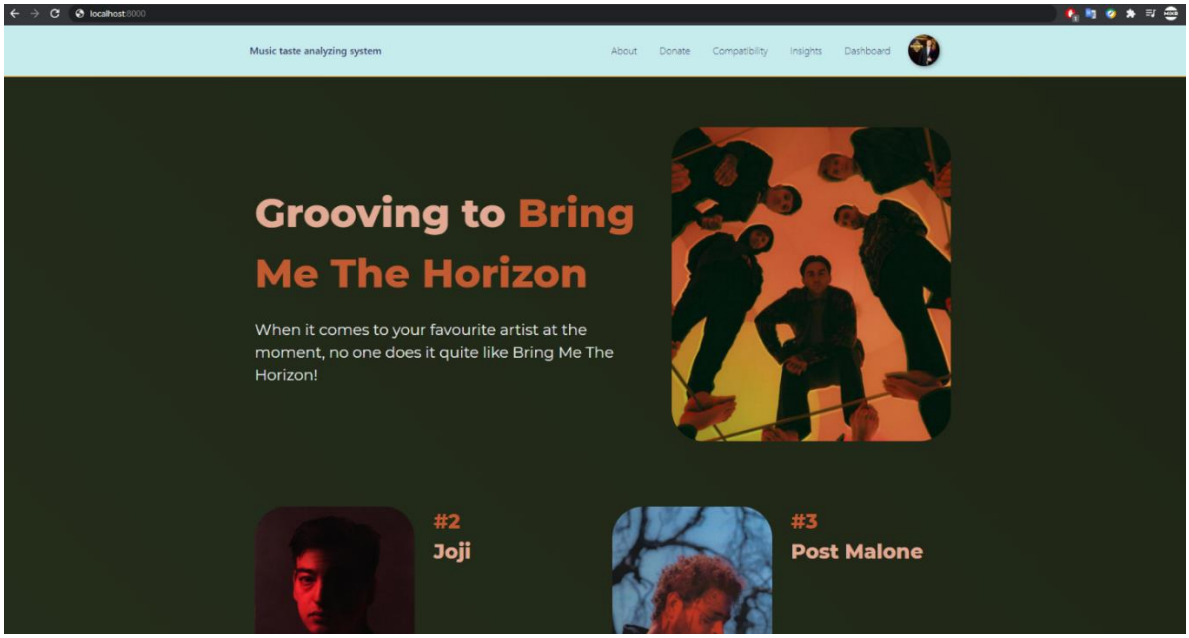


Рисунок 2.9 Секція з найпопулярнішими виконавцями за останній місяць

Також доступна секція, з описом тридцяти найпрослуховуваніших треків за весь час, за останній місяць та за півроку.

Використовуючи можливості фреймворку Vue.js вдалось адаптувати користувацький інтерфейс, під мобільні пристрої (див. рисунок 2.10).

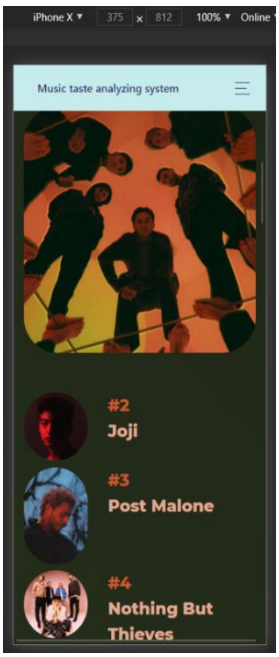


Рисунок 2.10 Вигляд вкладки «Чарти» мобільній версії браузеру Chrome для iPhone

Крім цього є розділ, з інформацією на рахунок рівня «веселості», «енергетики» та «танцювальності» серед збережених композицій користувача. Ця інформація збирається з-поміж усіх користувачів сервісу та записується в БД, де потім обробляється, та визначає середньостатистичний рівень настрою пісень.

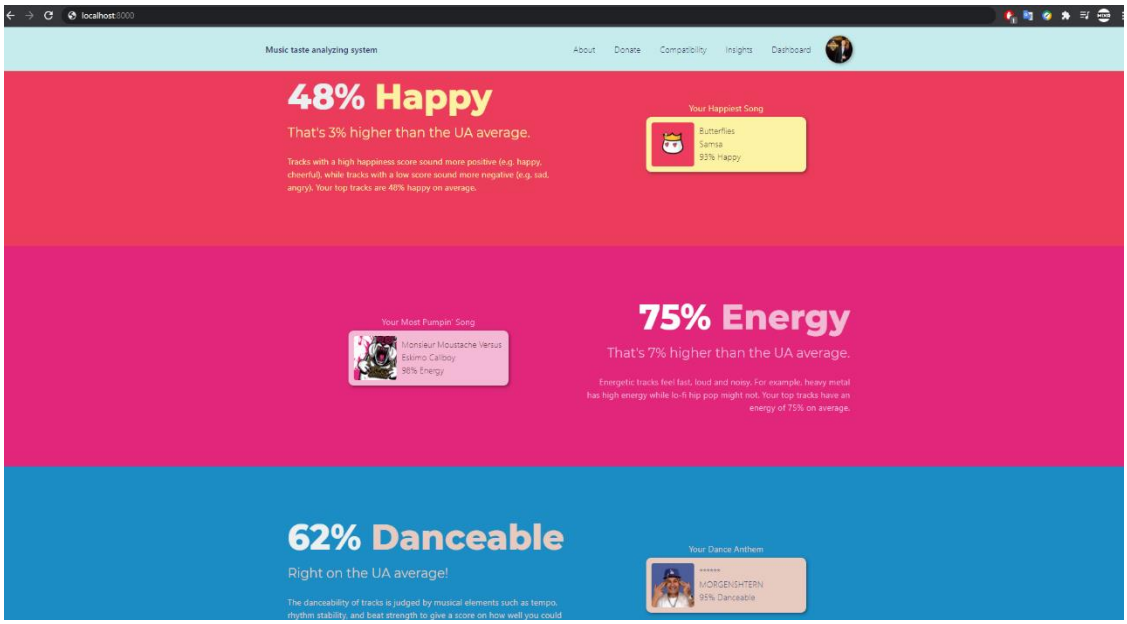


Рисунок 2.11 Секція з настроями, які панують у доданих піснях

Також було реалізовано анімований графік відображення популярності жанрів користувача. (див.рисунок 2.12).

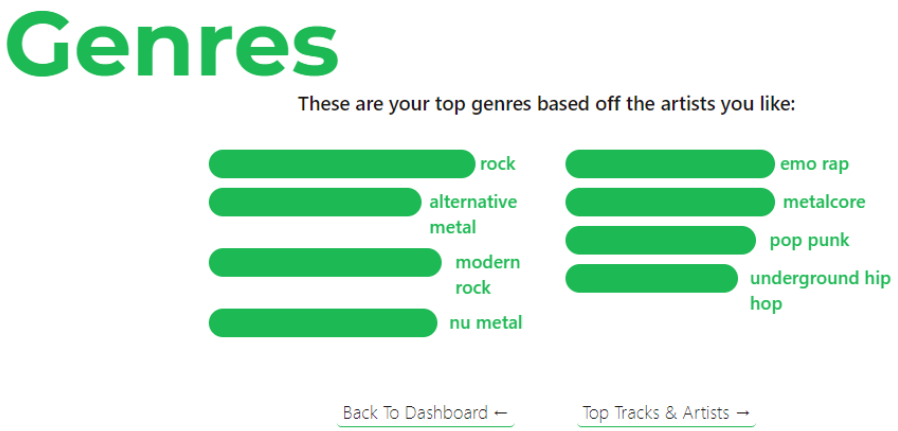


Рисунок 2.12 Статистика улюблених жанрів

Також завдяки алгоритму, можна дізнатися цікавий показник «нормальності» музичного смаку. Формується це значення в залежності від популярності улюблених виконавців слухача. Чим нижча оцінка, тим менше середньостастично популярних артистів ви слухаєте, відповідно росте оцінка, в залежності від кількості популярних виконавців яких ви слухаєте.

Рисунок 2.13 Оцінка та опис показника «нормальності»

Рівень «нормальності» записується для кожного користувача, що дозволяє порівнювати оцінку з іншими, зважаючи на місцезнаходження слухача.

2.3 Експериментальна частина

Для успішної роботи продукту на сервері, потрібно щоб він працював на Unix-подібній операційні системі. Щоб розгорнути проект, він був поміщений в декілька Docker-контейнерів, які можна зручно інсталювати використовуючи інструментарій **docker-compose**.

Docker-compose - інструментарій, за допомогою якого можна описати всю мікросервісну архітектуру у файлі-конфігурації, а згодом працювати з описаною системою. Цей інструмент використовується для заміни вже застарілих та складних у використанні bash-скріптів. У їх використанні є багато недоліків, які використання docker-compose вирішує. Наприклад, для зміни конфігурації використовуючи скрипти такого типу потрібно редагувати кожен з них, що породжує схильність до помилок.

Інструментарій docker-compose дозволяє вирішити значну кількість проблеми з скриптами .sh, хоча можливі ситуації в яких знадобиться використати їх для виклику команди docker-compose з відмінними аргументами командного рядка.

Файли docker-compose повинні вміщати таку інформацію:

- найменування сервісів;
- імена контейнерів;
- маппінг портів
- напрямки до томів де зберігатимуться програмні напрацювання;
- змінні середовища;

Нижче наведено подробиці опису розділу сервісу auth_api (див. рисунок 2.15):

- вибір контейнером мережі;
- шлях до директорії з змінними середовищами;

- потреба в доступі до оболонки Docker;

```
auth-api:
  build: ./app
  tty: true
  user: "${UID}:${GID}"
  container_name: test_app
  volumes:
    - "./app:/app"
  networks:
    - internal
  env_file:
    - .env
```

Рисунок 2.14 Опис сервісу auth_api, який відповідає за авторизацію користувачів

Директорія, яка вміщує Dockerfile, в котрій є дані про образ, який виступає основою контейнера, його робоча директорія, інструкції для інсталювання залежностей, команда запуску API та порт.

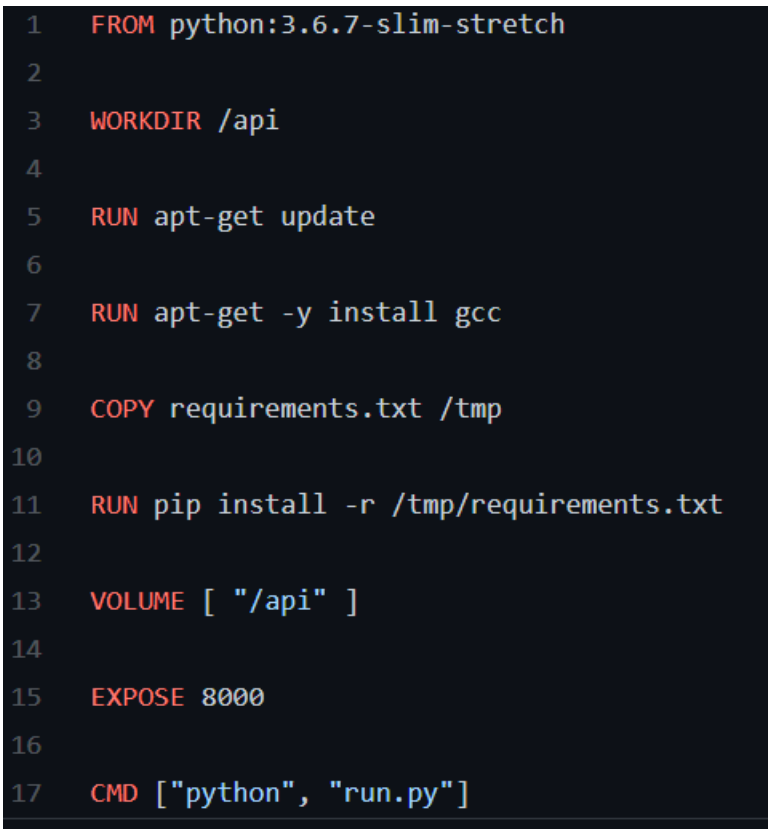
Для можливості запуску усіх сервісів, також потрібно їх «познайомити» з файлом Dockerfile.

Dockerfile – так званий сценарій, який містить послідовність команд та аргументів, потрібних для розгортання та підготовки продукту до запуску. Викликати його користувач може в командному рядку для складання образу[18].

На рисунку 2.15 продемонстровано приклад того, як локального «підняття» мікросервісу API.

У кодному з рядоків міститься інформація, що має виконуватися при «підняті» контейнера.

За допомогою цього файлу ми вказуємо компонувальнику, що для формування нашого контейнера використовуємо останній Docker образ Python версії 3.6.7. Після цього створюється робочий каталог */app* під наш мікросервіс.



```
FROM python:3.6.7-slim-stretch

WORKDIR /api

RUN apt-get update

RUN apt-get -y install gcc

COPY requirements.txt /tmp

RUN pip install -r /tmp/requirements.txt

VOLUME [ "/api" ]

EXPOSE 8000

CMD ["python", "run.py"]
```

Рисунок 2.15 Dockerfile, що описує мікросервіс API

Було взято за правило поміщення усього робочого коду програми до кореневої папки */app* контейнера. Те саме відбуватиметься в ситуаціїї з клієнтським додатком.

Далі встановлюється додатковий пакет gcc, який дозволить інсталювати потрібні pip-пакети з api / requirements.txt, які було вказано трохи вище, при його створенні.

Для запуску усіх docker-контейнерів потрібно застосувати команду «sudo docker-compose up» в термінальному вікні. Ця дія запустить завантаження всіх залежностей, а програмний продукт запуститься на майже будь-якому «лінуксовому» сервері.

Даний продукт розроблявся у системі розподіленого контролю версій файлів Git.

Контроль версій - це керування декількома версіями проекту. Для цього потрібен контроль та відстеження кожного додавання, зміни чи видалення файлів

у проекті. Тому контроль версій реєструє усі зміни внесені у файли (або групи файлів) та пропонує спосіб скасувати або пропустити кожну зміну.

Весь програмний код зберігається в закритому репозиторії на Git. Це вирішує низку таких проблем:

- зміна версій;
- порівняння відмінностей у цих версіях;
- ведення історії змін у файлах;
- відмітки певних версій для швидкого посилання.

Отож, мікросервіси інформаційної системи зберігаються на сервісі, який надає хостинг для девелопменту програмного забезпечення та контролю версій використовуючи систему Git, а саме веб-сервіс GitHub.

GitHub надає можливість для розробки, перегляду та розміщення коду. Крім цього керувати проектами та розроблаяти програмне забезпечення пліч-о-пліч з більше ніж тридцятьма мільйонами розробників [14].

Для використання цього сервісу на власній машині потрібна лише Unix-подібна операційна систеа та інстальований Git.

Для початку роботи необхідно:

- клонувати проект використовуючи команду «git clone» ;
- ознайомитися з README.md файлом, де представлений перелік необхідних залежностей;
- дотримуватися команд описаних в README файлі;
- для роботи з сервісами API потрібно встановити Python версії 3.7, додати локальне віртуальне середовище, встановити залежності цього API (переважно “pip install -r requirements.txt”) та стартувати сервіс (переважно “flask run”);
- для фронтенд сервісу необхідно встановити npm менеджер пакетів («npm install»), та зкомпілювати проект («npm run build»).

Для «підняття» сервісів над котрими розробник не планує працювати, необхідно скористатись інструкцією адміністратора, для запуску цих сервісів у фоновому режимі на своїй машині [17].

Найкомфортніше це реалізувати використовуючи інтегроване середовище розробки – PyCharm (для back-end) і WebStorm (для front-end).

PyCharm - це інтегроване середовище розробки для програмування (IDE) мовою Python. Він підтримує широкий функціонал: підтримка аналізу коду, графічний дебагер, інструментарій модульного тестування, можливість роботи з VCSes, а також підтримується Flask [19].

WebStorm - інтегроване середовище розробки для JavaScript, створене на база платформи IntelliJ IDEA.

У цьому IDE пристуній функціонал аналізу коду під час написання, навігації та налагодження, рефакторингу та автодоповнення. Також інтегровано можливість роботи з системами управління версіями. Ключовою перевагою WebStorm виступила можливість роботи з проектами (зокрема, рефакторинг коду JavaScript, який міститься в роздільних файлах і папках проекту, а також вкладеного в HTML). Більше того, в конструкціях з вкладеним JavaScript в HTML документ, в якому вкладено з ще одну таку пару підтримується множинна вкладеність, тож в них працює коректний рефакторинг.

Це допомагає розпочинати тестування або продовжувати розробку максимально зручно та оперативно.

Інструкція користувачу для використання програмного продукту:

Для моживості зручного використовувати сервіс був реалізований інтуєтивний та функціональний користувацький інтерфейс.

Інструкція користувача інформаційної системи:

1. Потрібно обрати стрімінговий сервіс для авторизації.

2. Після вводу даних для входу проходить перехід на головну сторінку.

3. Обрати комфортну для себе мову інтерфейсу натиснувши на «випадайці» EN/UA.

4. Обрати один з доступних розділів.

5. Зупинившись на розділі «Чарти», можна дізнатись персональну інформацію поділену за часовими рамками.

6. Скористатися можливістю додати список відтворення згенерований цим розділом.

7. Перейти до розділу «Порівняння з іншими», для демонстрації настрою власних композицій на фоні інших користувачів та порівняти рівень «нормальності» смаку.

8. При бажанні скористатися розділом «Донати», для пожертви розробнику сервісу.

9. Відвідати вкладку «Про нас», щоб ознайомитися з розробниками продукту та його функціоналом.

Мінімальні системні вимоги:

- Процесор Intel Pentium 4 або кращий з підтримкою SSE3;
- Оперативна пам’ять об’ємом від 128 МБ;
- Підключеняя до мережі швидкістю завантаження та вивантаження від 128 Kbit/s.
- Операційна система Windows 7 або вище;
- MacOS вище версії 10.5.6;
- Linux Ubuntu 14.04 (64-розрядна версія) або пізініше, Debian 8 або пізніше, openSUSE 13.3 і далі, Fedora Linux 24 або вище;
- Браузер Google Chrome, Opera, Firefox або Microsoft Edge[15].

2.4 Функціональне та структурне тестування

Для тестування програмного продукту обрали дані автоматизовані стратегії тестування:

- Unit test – перевірка невеликого шматку коду, для прикладу функцію чи клас, в ізоляції від решти системи; 1
- Тест інтеграції –перевірка більшої ділянки коду, наприклад декілька класів або підсистему. Переважно це ярлик, який використовують для тестів, об’ємніших за одиничний тест, аледрібніших за системні;

- End-to-end тест – перевірка всієї системи, яка відбувається в оточенні близькому до середовища кінцевого користувача;
- Функціональний тест – перевірка окремої частини функціоналу системи для відслідковування кості змін у класах.

Для застосування цих стратегій було використано pytest.

Pytest - програма для випробування програмного забезпечення, яка за визначенням є інструментом командного рядку, який автоматично знаходить написані вами тести, запускає їх та повідомляє вам результати. Він має різні бібліотеки, які ви можете застосовувати у своїх тестах, для ефективнішого тестування. Також є можливість розширення, шляхом написання плагінів або встановлення сторонніх плагінів. Окрім цього pytest просто інтегрується з інакшими інструментами, такими як безперервна інтеграція та веб-автоматизація [16].

Pytest є надійним засобом тестування коду, також його можна використовувати для будя-яких рівнів та типів тестування програмного забезпечення. Окрім того pytest можуть використовувати групи розробників, команди з контролю якості, незалежні тестувальники, особи, які практикують TDD, та open-source проекти. Фактично, більшість проектів по всій мережі перейшли з unittest або nose на pytest, наприклад Mozilla та Dropbox, адже pytest пропонує потужний функціонал, такий як «затвердження» переписування, моделі сторонніх плагінів та просту, але досить потужну систему кріплення, у якої немає аналогів у будь-яких інакших тестових структурах.

Для можливості контролю змін було використано інструмент для безперервної інтеграції Jenkins CI – відкритий інструментарій для постійної інтеграції. Враховуючи те, що Python не потребує компіляції, популярною практикою є застосування Jenkins або схожих систем CI для автоматизації запуску та звітування проектів Python.

Системи такого типу як Jenkins використовують для запуску тестових наборів після кожних змін з кодом. Pytest підтримує можливість створювати

файли у форматі junit.xml, потрібні Jenkins та інакшим системам CI для відтворення результатів тесту.

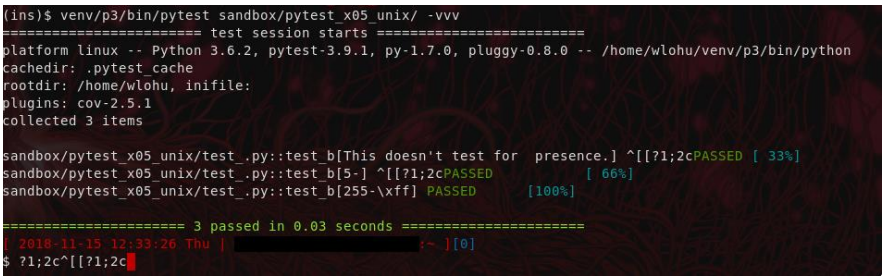


Рисунок 3.1 Результат проведення Unit- тесту.

Приклади тестів, що були реалізовані (див. рисунок 3.1):

- тестування можливості неавторизованого користувача отримати дані про персоніфіковані рекомендації;
- перевірка кінцевих точок доступу до API;
- тестування валідності виводу відповіді від API до front-end сервісу;
- тести класів, які відповідають за вивід чартів;
- перевірка алгоритму, яких прораховує «нормальність» музичного смаку користувача;
- тест на новостворенності профілю користувача і наявності усіх необхідних даних для аналізу;
- перевірка коректності виводу графіки на front-end частині;
- стрес-тест бази даних.

Опис проведеного експериментального дослідження:

Під час проведення експериментів виділили акценти на таких аспектах роботи:

- інтуїтивність користувацького інтерфейсу;
- повнота та працездатність доданих функцій;
- стабільність роботи системи на різних платформах.

Перевіряючи комфорт користування інтерфейсом, тестування проводилося на інтеграційних серверах. Так можна оцінити наскільки інтерфейс є доступним та простим у використанні.

Тестування ефективності та доповненості функцій інформаційної системи проводилось у поєднанні з unit-тестуванням. Крім того, паралельно відбувався пошук нових функцій для програмного продукту. Серед невеликої кількості людей було проведене коротке опитування. Для потенційних користувачів надали модливість залишити відгук, на основі яких можна вивести висновки на рахунок розвитку продукту.

Перевірка запуску на кросс-платформі проводилась шляхом запуску системи на різних платформах використовуючи різні налаштування та операційні системи.

Оцінювання та аналіз результатів

Враховуючи результати проведених тестів, можна підсумувати фінальний результат для клієнта:

- кінцева користь – користувачечі стрімінгового сервісу беззаперечно цікаво отримувати персоніфіковані дані про свої прослуховування музичній платформі;
- інтерфейс користувача виконує не лише декоративну роль, але й функціональну роль;
- кросс-платформа – сервіс дає можливість використання як з персонального комп’ютера так і з мобільних пристроїв;
- вдалий візуальний стиль – після відвідування сервісу у клієнта з’являється бажання відвідати інформацію ще раз, для насолоди деталями;
- соціальна цінність – можливість зібрати однодумців у одному місці та ділитися своїми результатами з користувачами по всьому світі;
- додаткова цінність – можливість збереження до сервісу згенерованих системою списків відтворення.

Звичайно, у інтерфейсу присутні і недоліки:

- недовершеність – деякі функції не до кінця «відполіровані»;

- стиль виконання – можливо, отримавши консультацію професійного UX-дизайнера можливо було б поліпшити дизайн кінцевого продукту;

Також після опитування користувачі підмітили недостачу, таких функцій системи:

- можливість перегляду інформації про виконавця з відкритих джерел;
- оплата донатів за допомогою різних платіжних систем, наприклад, PayPal або Scrill;
- можливість авторизації та аналізу більше стрімінгових сервісів (Tidal, Apple Music).

2.5 Висновки до розділу

В даному розділі детально описані етапи проектування кінцевого продукту. Першим етапом роботи є складання блок-схеми алгоритмів роботи інформаційної системи. Наступним кроком стало проектування діаграми класів нашого програмного забезпечення, які мають наступний вигляд:

- верхня секція з назвою класу;
- середня секція з атрибутами класу;
- нижня секція містить операції, які виконуються класом.

Далі перейшов до структуризації бази даних, яка складається з трьох блоків: users, journal і music_info. Отримавши вихідні дані я перейшов до етапу створення програмно-апаратного середовища. В даному продукті програмні модулі будуть декомпонуватися за принципом мікросервісної архітектури. Спілкування відбуватиметься за допомогою REST API запитів, задіюючи POST, GET, DELETE та PUT методи. Для взаємодії з інтерфейсом користувача клієнт сервіс працюватиме з веб-сторінкою побудованою на Vue.js, який буде «піднятий» на Node.js сервері. Запити користувача надсилатимуться з користувацького інтерфейсу на бекенд за через публічний прикладний програмний інтерфейс, а на самому сервері ці дані оброблятимуться за допомогою фреймворку Flask. Дані про користувача зберігатимуться в таблиці

Users. Кінцевими точками мікросервісу аналізу виступатиме збереження даних в таблиці Music_info. Крім того цей мікросервіс відповідає за логування інформації, створення зліпку сесії та записує ці дані до таблиці Journal.

Наступним етапом виступає розробка та опис інтерфейсу користувача з такими вимогами:

- інтуїтивність інтерфейсу;
- швидкий доступ до всіх розділів;
- підтримка різними платформами;
- візуалізація чартів у вигляді графіків;
- можливість поділитися в соцмережах;
- підтримка локалізації.

В ході проведення експериментальної частини вирішив помістити проект в декілька Docker-контейнерів для реалізації успішної роботи продукту на сервері. Також було створено інструкцію користувачу та виведено мінімальні системні вимоги для якісної роботи кінцевого продукту.

Фінальним етапом цієї роботи виступило функціональне та структурне тестування. Під час проведення експериментів виділили акценти на таких аспектах роботи:

- інтуїтивність користувацького інтерфейсу;
- повнота та працездатність доданих функцій;
- стабільність роботи системи на різних платформах.

Враховуючи результати проведених тестів, можна підсумувати фінальний результат для клієнта:

- користувачеві даного сервісу беззаперечно цікаво отримувати персоніфіковані дані про свої прослуховування на музичній платформі;
- інтерфейс виконує не лише декоративну , але й функціональну роль;
- кросс-платформа;
- вдалий візуальний стиль;
- соціальна цінність;

- додаткова цінність – можливість збереження до сервісу згенерованих системою списків відтворення.

3 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ХОРОНИ ПРАЦІ

3.1 Значення адаптації в трудовому процесі.

Праця як форма життєдіяльності людини здійснюється за певних умов виробничого середовища та за певних нервово-м'язових навантажень. Організм працівника разом з виробничим середовищем складає єдине ціле. Характерною особливістю внутрішнього середовища організму є його відносна постійність. За будь-яких змін зовнішніх умов внутрішнє середовище залишається постійним, що є необхідною умовою життєдіяльності організму.

Пристосування організму до мінливих умов зовнішнього сере-довища і збереження постійності його внутрішнього середовища здійснюються рефлекторним і гуморальним шляхом. Центральна нервова система, посилаючи імпульси різним органам, посилює або послаблює їх діяльність, пристосовуючи до зовнішніх умов. Хімічний, або гуморальний, механізм регуляції полягає в тому, що хімічні речовини розносяться кров'ю по всьому організму і залежно від їх концентрації гальмують або активізують роботу тих чи інших органів. Обидва механізми регуляції взаємопов'язані і становлять єдиний нейрогуморальний механізм, який забезпечує саморегуляцію фізіологічних функцій організму.

Пристосовні реакції організму людини до зовнішнього середовища виявляються у формі складних рефлекторних актів. Крім того, адаптація, як пристосовні зміни в організмі, характеризується розширенням фізіологічних можливостей, збільшенням працездатності або підвищенням фізіологічної опірності організму зовнішнім впливам. Забезпечується ця можливість шляхом: зміни порогів чутливості аналізаторів; підвищення лабільності фізіологічних систем, яка полягає в швидкому поверненні до вихідного стану; переходу фізіологічних систем на більш високі рівні функціонування; розширення діапазону фізіологічних резервів; мобілізації енергетичних ресурсів та захисних сил.

Суттєвий вплив на адаптаційні процеси людини, зокрема у трудовій діяльності, має психічна адаптація. Вона базується на використанні людиною раніше набутого досвіду, враховує потреби і мотивацію, що забезпечує більш точну диференціацію реакцій і поведінки працівника.

Психічна адаптація в трудовій діяльності — це процес встановлення оптимальної відповідності між особистістю і навколишнім середовищем, в тому числі соціальним, яка сприяє задоволенню актуальних потреб і реалізації значущих цілей за умови збереження фізичного і психічного здоров'я працівника.

Адаптація людини до праці має активний характер. Працівник не лише пристосовується до зовнішнього середовища, а й змінює його, водночас змінюючись і сам. При цьому за сприятливих умов виробничого середовища і за оптимальних навантажень підвищуються стійкість і працездатність організму, за несприятливих умов рівень активності фізіологічних систем знижується.

Пристосування організму працівника до умов виробничого середовища і трудових навантажень забезпечується його резервами.

Резерви організму — це його здатність посилювати свою діяль-ність порівняно зі станом відносного спокою. Розмір резервів окремої функції — це різниця між максимально можливим рівнем і рівнем в стані відносного фізіологічного спокою. Наприклад, хвилинний об'єм дихання в спокої становить в середньому 5…8 л повітря, а максимально можливий за важкої роботи 120…200 л. Резерв становить 115…192 л. Аналогічно змінюються інші фізіологічні показники. При фізичних навантаженнях фізіологічні показники можуть збільшуватися в 2—16 разів порівняно зі станом спокою. Організм людини має морфологічні, біохімічні, фізіологічні, психологічні резерви.

Морфологічні резерви характеризуються особливостями будови тканин і органів, надлишком певних структурних елементів порівняно з потребою. Наприклад, у крові міститься в 500 разів більше протромбіну, ніж потрібно для згортання всієї крові.

Біохімічні резерви пов'язані з запасом енергетичних речовин в організмі.

Фізіологічні резерви зумовлюються функціональним станом окремих органів і організму в цілому, діяльність яких посилюється в результаті нейрорегуляції.

Психологічні резерви пов'язані з психічними функціями людини, є показником розумової працездатності і визначаються високою стійкістю до несприятливих факторів зовнішнього середовища.

Загальні фізіологічні резерви працівника залежать від резервів його рухового апарату, дихальної і серцево-судинної систем. З фізіологічними резервами організму пов'язана фізична працездатність людини. Мобілізації резервів за рахунок активізації фізіологічних функцій сприяє м'язова діяльність. При цьому фізіологічні резерви використовуються послідовно залежно від вимог і умов роботи. Так, перша черга резервів включається зразу при переході організму від стану відносного спокою до неважкої м'язової діяльності. Друга черга резервів використовується під час виконання робіт, які вимагають дуже великих фізичних зусиль. У повсякденному житті людина використовує лише близько 35% своїх резервних можливостей. Якщо робота вимагає використання 50% наявних резервів, то у працівника розвивається фізична та психічна втома, а отже виконання роботи змушує його докладати великих вольових зусиль. У разі використання 65% резервів необхідні надзвичайні вольові зусилля, і така напружена робота швидко припиняється. Понад ці резерви робота взагалі неможлива. У процесі виробничої діяльності людина ніколи не працює на межі своїх можливостей, тобто не використовує максимально свої фізіологічні резерви. Звичайне навантаження працівника в нормальних умовах виробництва становить 30—45% від максимального навантаження, яке людина може виконати, мобілізуючи свої фізіологічні резерви.

Реалізація фізіологічних резервів працівника при виконанні роботи відбувається шляхом підвищення ефективності обмінних процесів в організмі, посилення роботи органів дихання і серцево-судинної системи, перерозподілу крові до працюючих органів, розширення механізмів терморегуляції і т. ін. Значним чинником і механізмом мобілізації фізіологічних резервів є емоції.

Загалом рівень активності фізіологічних систем у процесі праці залежить від вихідного функціонального стану працівника перед роботою, інтенсивності навантажень і умов праці, стійкості організму, індивідуальних особливостей, пов’язаних з віком, статтю, властивостями нервових процесів, м’язовою силою і витривалістю тощо. Знання закономірностей зміни фізіологічних функцій працівника має вважливе значення для обгрунтування навантажень, оптимізації умов праці і відпочинку.

3.2 Загальні вимоги безпеки з охорони праці для користувачів ПК

Згідно статті 8 Конституції України – основним правовим документом України є Конституція України. Конституція України має найвищу юридичну силу. Закони і інші нормативно-правові акти приймаються на підставі Конституції України і повинні відповідати їй. На підставі Конституції України прийнятий Закон України “про охорону праці”.

Згідно закону України "про охорону праці" ст.1, охорона праці - це система правових, соціально-економічних, організаційно-технічних заходів, а так само санітарно-гігієнічних і лікувально-профілактичних засобів, направлених на збереження здоров'я і працездатності людини в процесі праці.

Згідно ст.2 закону «про охорону праці» дія Закону "про охорону праці" розповсюджується на всі підприємства, установи і організації не залежно від форми власності і видів їх діяльності, на всіх громадян, які працюють, а також привернуті до праці на цих підприємствах.

Згідно ст.4 Закону України «про охорону праці» державна політика в області охорони праці визначається відповідно Конституції України Верховною Радою України і направлена на створення належних, безпечних і здорових умов праці, запобігання нещасним випадкам і професійним захворюванням.

За порушення законів і інших нормативно-правових актів про охорону праці, створення перешкод в діяльності посадовців органів державного нагляду за охороною праці, а також представників профспілок, їх організацій і об'єднань

винні особи притягуються до дисциплінарної, адміністративної, матеріальної, кримінальної відповідальності згідно закону (ст. 44 Закону "про охорону праці").

Виходячи із загальних завдань в області охорони праці, в даному дипломному проекті розглядаються наступні завдання:

- характеристика робочого приміщення;
- мікроклімат робочого приміщення;
- освітлення приміщення;
- шум та вібрація у робочому приміщенні;
- оцінка електробезпеки;
- електромагнітне випромінювання.

Характеристика робочого приміщення.

В даному приміщенні є 2 робочих місця. Для зберігання документів використовується маленька тумба. Вікно розташоване навпроти робочих місць. Двері розташовані в куті приміщення. Повна характеристика виглядає так:

- довжина (a) – 5 метрів;
- ширина (b) – 5 метрів;
- висота (h) – 3 метра ;
- кількість робочих місць (n) – 2;
- площа (S) – 25 $м^2$;
- об’єм (V) – 75 $м^3$.

Порівнявши дані приміщення з вимогами до організації робочого місця, бачимо, що воно відповідає вимогам щодо охорони праці при організації роботи з ВДТ електронно-обчислювальних машин.

Мікроклімат робочого приміщення.

Згідно з ДСН 3.3.6.042-99 роботи, які виконуються користувачами електронно – обчислювальних машин відносяться до легких фізичних робіт категорії Іа, тому на даних робочих місцях мають забезпечуватись оптимальні значення параметрів мікроклімату.

Освітлення приміщення.

Згідно приміщення повинне мати природне і штучне освітлення.

Природне освітлення приміщення відбувається за системою однобічного бічного освітлення. Природне світло проникає у приміщення через два вікна. Також наявні жалюзі з можливістю регулювання рівня освітленості. Вікна приміщення орієнтовані на схід. Всередині приміщення стіни та стеля білого кольору.

В даному приміщенні використовується система загального рівномірного штучного освітлення. Мається ряд світильників Л201Б 4х40-0.3, у кожному з яких знаходиться по чотири лампи типу ЛБ-40.

Для загального штучного освітлення нормуються такі параметри:

- найменша припустима освітленість – Е (лк);
- показник дискомфорту – М;
- коефіцієнт пульсації освітленості – Кп (%).

Визначимо фактичну освітленість в кабінеті з формули світлового потоку:

$$E_{\Phi} = \frac{\Phi_{л} \cdot \eta \cdot N}{S \cdot K_{з} \cdot Z} \qquad (3.1)$$

де

N – число світильників у приміщенні, N = 4*4=16;

η – коефіцієнт використання світлового потоку;

Фл – світловий потік лампи;

Кз – коефіцієнт запасу, Кз = 1.5;

Z – коефіцієнт нерівномірності, Z=1.1 для люмінісцентних ламп;

S – площа приміщення;

Еф – фактична освітленість, створювана всіма світильниками.

Підвісна стеля білого кольору має коефіцієнт відбиття ρст = 0.5, стіни пофарбовані теж в білий колір -ρсн = 0.3.

Розрахуємо параметр і:

$$i = \frac{a \cdot b}{h(a+b)} \qquad (3.2)$$

Для вказаного параметру і та коефіцієнтів відбиття стелі і стін коефіцієнт використання світлового потоку η=0,44.

Нормативне значення освітленості для кабінету, де працюють з використанням ВДТ, згідно таблиці Д.1 дод. Д ДБН В.2.5-28-2018 становить 200 лк. Допустимі межі відхилення 10%. Згідно формули (6.1) $E\phi$=180.9, що потрапляє в допустимі відхилення.

Шум та вібрація у робочому приміщенні.

У приміщенні є такі джерела постійного шуму:

- вентилятори комп'ютерів,
- принтер,
- аудіо-система.

Зовнішніми джерелами шуму і вібрації в приміщенні є проїжджаючі транспортні засоби. Даний шум є постійним. Фактичний рівень шуму склав 48 дБА. Норма складає не більше 50 дБА, тому даний аспект покращення не потребує.

Оцінка електробезпеки.

Проаналізуємо стан електробезпеки в робочому приміщенні:

- напруга всіх приладів 220 В;
- проводка захована і ізольована;
- кожне робоче місце обладнане окремими розетками по 220 В;
- підлога ізолююча – лінолеум.

Проаналізувавши наведене вище, можемо сказати, що кабінет відноситься до приміщень без підвищеної електронебезпеки.

Електромагнітне випромінювання.

Джерелом електромагнітного випромінювання в сучасному офісі є візуальні дисплейні термінали. Нормування електромагнітного випромінювання здійснюється згідно положень ДСанПіН 3.3.2-007-98.

Джерелом електромагнітного випромінювання в приміщенні є 2 дисплеї SONY Powercolor 3220, які повністю відповідають міжнародному стандарту TCO-03.

ВИСНОВКИ

В першому розділі кваліфікаційної роботи було описано предмет дослідження, для визначення необхідного функціоналу кінцевого продукту проведено порівняльний аналіз існуючих інструментів для моніторингу стрімінгових сервісів, що показав на практиці всі переваги та недоліки існуючих аналогів. В результаті я отримав кінцевий список потрібних функцій та перейшов до етапу вибору програмних складових. Створено концептуальну модель, визначені основні складові для оцінки музичного смаку користувачів.

В підсумку для розробки було використано такі рішення:

- операційна система – Linux OS;
- мови програмування – Python, JavaScript;
- інтегровані середовища розробки– PyCharm, WebStorm;
- фреймворки – Flask, Vue.js;
- контейнери – Docker;
- середовища – PyCharm та WebStorm;
- систему керування базами даних – PostgreSQl.

Кінцевим етапом даного розділу виступило створення концептуальної моделі інформаційних потоків даної системи.

В другому розділі детально описані етапи проектування кінцевого продукту. Першим етапом роботи стало складання блок-схеми алгоритмів роботи інформаційної системи. Далі перейшов до структуризації бази даних, яка складається з трьох блоків: users, journal і music_info. Отримавши вихідні дані я перейшов до етапу створення програмно-апаратного середовища. В даному продукті програмні модулі будуть декомпонуватися за принципом мікросервісної архітектури. Спілкування відбуватиметься за допомогою REST API запитів, задіюючи POST, GET, DELETE та PUT методи.Для взаємодії з інтерфейсом користувача клієнт сервіс працюватиме з веб-сторінкою побудованою на Vue.js. Запити користувача надсилатимуться з користувацького інтерфейсу на бекенд через публічний прикладний програмний інтерфейс, а на

самому сервері ці дані оброблятимуться за допомогою фреймворку Flask. Дані про користувача зберігатимуться в таблиці Users. Кінцевими точками мікросервісу аналізу виступатиме збереження даних в таблиці Music_info. Крім того цей мікросервіс відповідає за логування інформації, створення зліпку сесії та записує ці дані до таблиці Journal. В ході проведення експериментальної частини вирішив помістити проект в декілька Docker-контейнерів для реалізації успішної роботи продукту на сервері. Також було створено інструкцію користувачу та виведено мінімальні системні вимоги для якісної роботи кінцевого продукту.

Фінальним етапом проведеної роботи виступило функціональне та структурне тестування.

У розділі «Безпека життєдіяльності, основи хорони праці» висвітлено значення адаптації в трудовому процесі та описано загальні вимоги безпеки з охорони праці для користувачів ПК.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Monika Mankarious Music training and emotion comprehension in childhood. Emotion. 12 (5), 2012, - 891p.
2. Allen B. Downey Think Python/ C. Bauer, G. King. — Think Python First Edition, 2015. — 244 p.
3. Mark Lutz Learning Python, Fifth Edition, 2013. — 123 p.
4. David Flanagan JavaScript: The Definitive Guide, 7th Edition, 2020. – 15 p.
5. Илья Кантор Современный учебник JavaScript в 3 книгах, 2019. – 51 ст.
6. Christopher Negus Linux Bible, 10th Edition, 2020. – 212 p.
7. Richard Bullington-McGuire Docker for Developers, 2020. – 41 p.
8. Miguel Grinberg Flask Web Development: Developing Web Applications with Python 2nd Edition, 2018. – 32 p.
9. Adam Freeman Pro Vue.js 2, 2018. – 120 p.
10. Sufyan bin Uzayr JavaScript Frameworks for Modern Web Development: The Essential Frameworks, Libraries, and Tools to Learn Right Now 2nd ed. Edition, 2019 – 235 p.
11. Simon Riggs PostgreSQL 11 Administration Cookbook: Over 175 recipes for database administrators to manag enterprise databases Paperback, 2019, - 32 p.
12. Craig Larman Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition, 2004, - 812 p.
13. Philipp K. Janert D3 for the Impatient: Interactive Graphics for Programmers and Scientists 1st Edition, 2019, - 71 p.
14. Doguhan Uluca Angular for Enterprise-Ready Web Applications: Build and deliver production-grade and cloud-scale evergreen web apps with Angular 9 and beyond, 2nd Edition, 2020, 342 p.
15. Brian Ward How Linux Works/ Brian Ward— 2nd Edition: What Every Superuser Should Know, 2014. — 392p.
16. Brian Okken Python Testing with pytest: Simple, Rapid, Effective, and Scalable 1st Edition, 2017, 89 p.

17. Mariot Tsitoara Beginning Git and GitHub: A Comprehensive Guide to Version Control, Project Management, and Teamwork for the New Developer, 2019, 43 p.
18. J.Turnbull “The Docker Book: Containerization is the new virtualization” , 2019, 43 p.
19. M.Harrison “Effective PyCharm: Learn the PyCharm IDE with a Hands-on Approach” , 2017, 89 p.
20. M.Herman “Test-Driven Development with Python, Flask, and Docker” , 2017, 89 p. URL:https://testdriven.io/courses/tdd-flask/
21. P. Luzanov, E. Rogov, I. Levshin “PostgreSQL for beginners” 2019. – 143 p.
22. Donald Knuth The Art of Computer Programming - Volume 1: Fundamental Algorithms, 1986, - 519p.
23. “The Rise of Streaming Music and Implications for Music Production” , 2020, 342 p. URL: https://doi.org/10.1515/rne-2017-0064
24. Hiller, R. S., “Sales Displacement and Streaming Music: Evidence from YouTube,” URL: http://faculty.fairfield.edu/rhiller/Research/Streamingmusic.pdf
25. L. Aguiar, B. Martens “Digital music consumption on the internet: evidence from clickstream data” URL:https://www.sciencedirect.com/science/article/pii/S0167624516000068
26. Aguiar L., Waldfogel J.Streaming Reaches Flood Stage: Does Spotify Stimulate or Depress Music Sales? URL:https://www.sciencedirect.com/science/article/abs/pii/S0167718717301753?via%3Dihub
27. J. Spitzer, “Analyzing Music Taste” URL:https://towardsdatascience.com/analyzing-music-taste-64202f602bcd/
28. G. McIntiere “A Machine Learning Deep Dive into My Spotify Data” URL:https://opendatascience.com/a-machine-learning-deep-dive-into-my-spotify-data/
29. J. Cabreira “A Music Taste Analysis Using Spotify API and Python.” URL: https://towardsdatascience.com/a-music-taste-analysis-using-spotify-api-and-python-e52d186db5fc

30. J. De Dios Santos “Is my Spotify music boring? An analysis involving music, data, and machine learning”
URL:https://towardsdatascience.com/is-my-spotify-music-boring-an-analysis-involving-music-data-and-machine-learning-47550ae931de
31. Spotify URL:https://developer.spotify.com/documentation/web-api/reference/#endpoint-get-audio-features
32. Rare Loot [Електронний ресурс]
URL: https://rareloot.medium.com/extracting-spotify-data-on-your-favourite-artist-via-python-d58bc92a4330
33. R.Layton “Learning Data Mining with Python: Use Python to manipulate data and build predictive models” Packt Publishing, 2017. – 120 p.
34. 14. M.Herman “Authentication with Flask, React, and Docker”
URL:https://testdriven.io/courses/auth-flask-react/
35. B.Nelson “Getting to Know Vue.js: Learn to Build Single Page Applications in Vue from Scratch” 2019. – 143 p.
36. H.Djirdeh “Fullstack Vue: The Complete Guide to Vue.js”
37. O.Filipova “Learning Vue.js 2” 2019. – 199 p.
38. M. Kennedy and M. Harrison “Effective PyCharm” 2019. – 152 p.
39. S.Rosca “WebStorm Essentials: Build efficient HTML, CSS and JavaScript applications using the powerful WebStorm IDE” 2016. – 147 p.
40. R. O. Obe “PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database” 2017. – 143 p.
41. M. Haverbeke “Eloquent JavaScript: A Modern Introduction to Programming” 2020. – 155 p.
42. E. Elliott “Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Moderns JS Libraries 2020. – 147 p.
43. D. Flanagan “JavaScript: The Definitive Guide” 2019. – 167 p.