

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ОБМІНУ ВРАЖЕННЯМИ ВІД
МУЗИЧНИХ КОНЦЕРТІВ
SOFTWARE FOR SHARING MUSIC CONCERT IMPRESSIONS

Виконав: студент(ка) 4 курсу, групи КНТ-129сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Котляров В.І.

(прізвище та ініціали)

Керівник Скрупський С.Ю.

(прізвище та ініціали)

Рецензент Голуб Т.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Котлярова Владислава Ігоровича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення обміну враженнями від музичних концертів
Software for Sharing Music Concert Impressions

керівник проєкту (роботи) Скрупський Степан Юрійович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «27» квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) червень 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Розробка програми. 4. Експлуатація та тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Скрупський С.Ю., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання «4» квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Розробка архітектури програми	3 тиждень	Розділ 2
4	Розробка програми	4-5 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	6 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
7	Захист роботи	8 тиждень	

Студент(ка)

(підпис)

Котляров В.І.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Скрупський С.Ю.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 83 с., 28 рис., 2 табл., 3 дод., 11 джерел.

МУЗИКАНТ, МУЗИЧНИЙ КОНЦЕРТ, ВРАЖЕННЯ, ОЦІНЮВАННЯ, ВЕБСИСТЕМА, ВЕБФРЕЙМВОРК.

Об'єкт роботи – процес обміну враженнями від музичних концертів. Предмет роботи – програмне забезпечення обміну враженнями від музичних концертів. Мета роботи – розробити програмне забезпечення обміну враженнями від музичних концертів для підвищення рівня інформованості потенційних відвідувачів концертів.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування Python, вебфреймворк Django, система керування базами даних MySQL.

Результати. У результаті аналізу програмних аналогів виявлено недостатню кількість інструментів обміну враженнями відвідувачів концертів для залучення користувачів програм у поточний стан індустрії живих виступів у програмних рішеннях, які використовуються для пошуку або отримання інформації про музичні концерти. Розроблена вебсистема направлена на вирішення виявленої проблеми, дозволяючи зберігати власні враження про концерти, виставляти оцінки за різними критеріями, підписуватися на враження інших користувачів.

Висновки. Розроблена програма може використовуватися для пошуку концертів як традиційним шляхом, визначаючи музикантів, концертні зали, так і користуючись враженнями відвідувачів концертів музиканта загалом, у конкретному турі зокрема.

Галузь використання. Концертна індустрія, зокрема пошук музичних концертів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 83 pages, 28 figures, 2 tables, 3 appendixes, 11 sources.

MUSICIAN, MUSIC CONCERT, IMPRESSIONS, EVALUATION, WEBSYSTEM, WEBFRAMEWORK.

The object of work is the process of exchanging impressions from music concerts. The subject of the work is software for exchanging impressions from music concerts. The purpose of the work is to develop software for exchanging impressions from music concerts to raise awareness of potential concert visitors.

Materials, methods and technical means: object-oriented programming, Python programming language, Django web framework, MySQL database management system.

Results. The analysis of software analogues showed a lack of tools to share the impressions of concert visitors to attract software users to the current state of the live performance industry in software solutions used to search for or obtain information about music concerts. The developed web system is aimed at solving the identified problem, allowing users to keep his own impressions of the concerts, to give ratings according to various criteria, to subscribe to the impressions of other users.

Conclusions. The developed software should be used to search for concerts in the traditional way, identifying musicians, concert halls, and using the impressions of visitors to concerts of the musician in general or in a particular tour specifically.

Field of use. Concert industry, in particular the search for music concerts.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	7
Вступ	8
1 Аналіз предметної області	10
1.1 Огляд програмних аналогів	10
1.2 Порівняння програмних аналогів	15
1.3 Висновки за розділом 1	16
2 Розробка архітектури програми	18
2.1 Вибір мови програмування	18
2.2 Функціональні вимоги до програмного забезпечення	19
2.3 Проектування бази даних.....	24
2.4 Висновки за розділом 2	30
3 Розробка програми	31
3.1 Структура розробленої вебсистеми.....	31
3.2 Опис реалізованих класів програми	32
3.3 Висновки за розділом 3	36
4 Експлуатація та тестування програми.....	37
4.1 Призначення програми.....	37
4.2 Умови виконання програми	37
4.3 Тестування програми	37
4.4 Висновки за розділом 4	46
Висновки.....	47
Перелік джерел посилання	48
Додаток А Технічне завдання.....	49
Додаток Б Текст програми.....	54
Додаток В Слайди презентації	76

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVT – Model-View-Template;

БД – база даних.

ВСТУП

Індустрія живих виступів є однією з потужних складових економік розвинутих країн.

Загальнонаціональний економічний вплив галузі живих виступів у Сполучених Штатах Америки в цифрах оцінюється наступним чином:

- загальний обсяг в розмірі 132,6 мільярдів доларів, що забезпечив 913 000 робочих місць із суміжним доходом від супутньої праці приблизно в 42,2 мільярда доларів у 2019 році;

- у 2019 році галузь принесла прямий вплив у розмірі 55,2 мільярда доларів, що включало місцеві оперативні витрати на місцях проведення подій;

- загалом у 2019 році індустрія живих подій принесла загальний фіскальний результат у розмірі 17,5 мільярдів доларів, включаючи майже 9,3 мільярдів доларів США федеральних податкових надходжень і 8,3 мільярда доларів США податкових надходжень штатів і на місцях.

- концерти та живі виступи створюють значний економічний вплив на державному та місцевому рівнях, оскільки учасники таких подій витрачають гроші в місцевій економіці для підтримки щоденної діяльності, включаючи витрати на виплату заробітної плати, витрати на маркетинг, юридичні послуги та професійні послуги, крім того економічний вплив цих подій виходить за межі самих робочих місць, заходу та місця проведення, розширюючись до місцевих ресторанів, готелів, барів, роздрібних торговців та малого бізнесу: наприклад, якщо іногородній учасник витратив 100 доларів США на квиток на концерт, місцева економіка отримає в середньому додаткові 334,92 долари США, що призведе до загальних витрат в 434,92 долари США [1].

Зважаючи на такий великий економічний ефект даної галузі, звичайно важливим є залучення широкої кількості відвідувачів на концерти, а цьому звичайно сприяють враження людей, які вже відвідали подібний концерт.

Звичайно, що в сучасному світі цьому допомагають соціальні мережі, проте орієнтуватися у всьому музичному світі таким чином достатньо важко, адже спеціалізованих інструментів соціальні мережі не містять. Саму тому в даній роботі розробляється програмне забезпечення обміну враженнями від музичних концертів, що має забезпечити такі спеціалізовані засоби.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд програмних аналогів

До програмних аналогів під час проведення аналізу предметної області було віднесено програми, які стосуються безпосередньо інформації про концерти музикантів.

Songkick [2] є програмним рішенням, яке дозволяє головним чином переглядати інформацію про концерти музикантів, тобто шукати потрібні користувачу концерти (рис. 1.1).

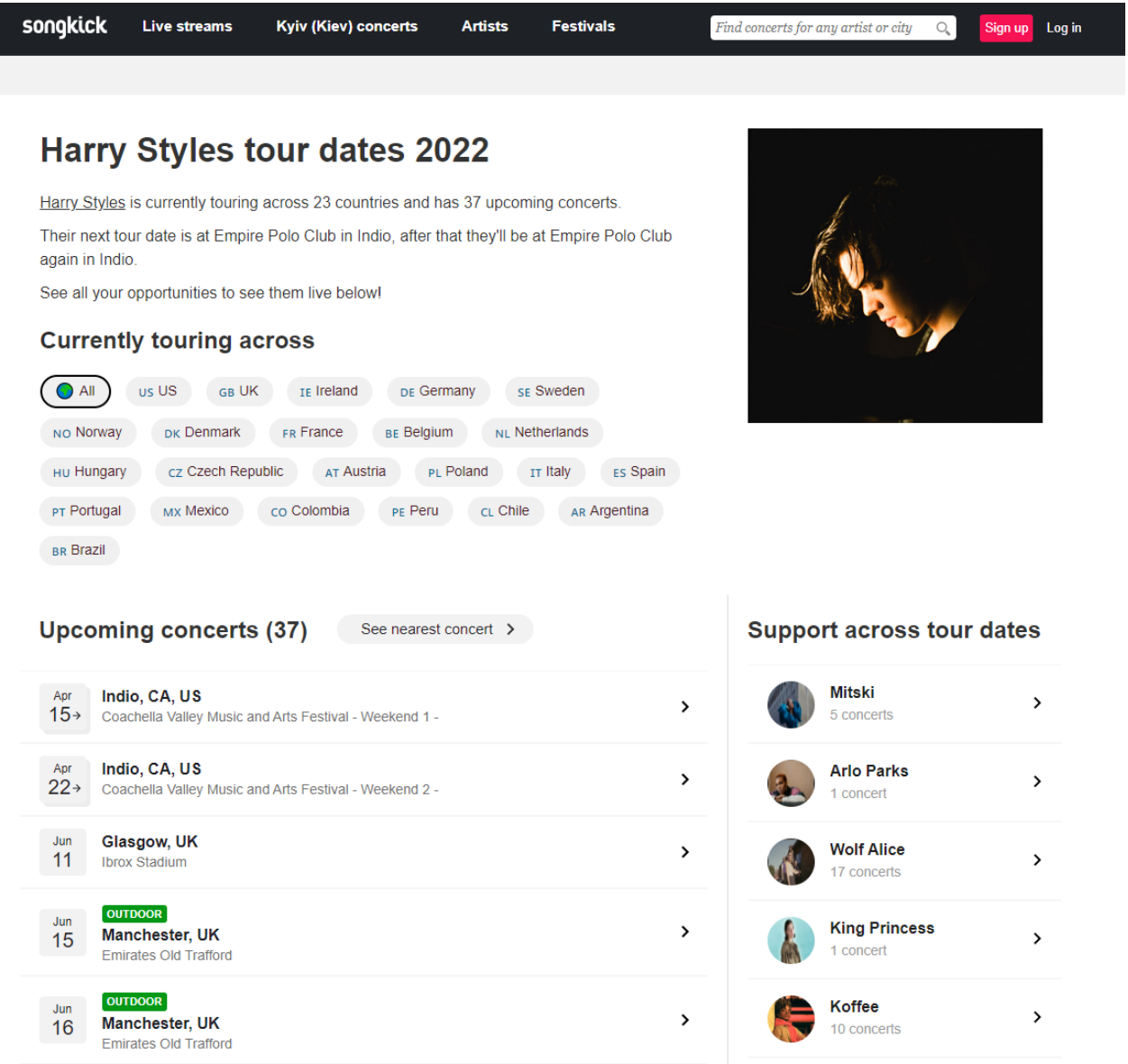


Рисунок 1.1 – Інтерфейс вебсайту Songkick

За кожним музикантом програма дозволяє переглядати його концерти, в тому числі минулі та майбутні. Користувач може підписуватися на музикантів.

Доступні відгуки (враження) користувачів, але слід зазначити, що в даному випадку ці відгуки на виступи виконавця загалом, тобто вони не стосуються конкретних концертів, а доступні одразу і тільки на сторінці виконавця загалом, що значно обмежує користь від них, адже рівень оцінок користувачів може змінюватись в залежності від туру: один тур музиканта є успішним, а інший – ні, відповідно купуючи квитки на концерт, користувачу добре було б знати не про якісь загальні відгуки, а про те, на скільки цей відгук відповідає саме сучасному періоду виступів музиканта. Окрім того від туру до туру змінюється також і загальна програма виступу (основний сетлист концертів).

Доступ до інформації про концерти відбувається або через вибір концертного музиканта, або через пошук за іменем або містом. Інших параметрів пошуку немає.

Окремо виділено доступ до фестивалів, звідки можна задати країну відображення, обираючи серед найбільш популярних.

За користувачем можна переглянути перелік концертів, які він відвідував, проте переглянути перелік його вражень не можна. Тобто враження використовуються як певний шлях загального привертання уваги, проте з ними працювати достатньо незручно. Навігації між враженнями немає.

Програмний застосунок для мобільних пристроїв після авторизації просить сканувати локальну музику або обліковий запис Spotify, а також виконати налаштування облікового запису Spotify, щоб покращити свої рекомендації для виконавців та майбутніх подій:

– якщо користувач не використовує Spotify, Songkick просить вибрати 20 виконавців, щоб створити дещо обмежений перелік, який охоплює лише найпопулярніших виконавців;

– програма надає списки концертів лише тих виконавців, яких користувач відстежує: для невеликих місцевих шоу Songkick використовує систему на основі областей, де можна вибрати місто, щоб відстежувати концерти, ці переліки можна відсортувати за жанром [3].

Загалом слід сказати, що це рішення достатньо зручне щодо отримання інформації про концерти, проте щодо роботи саме з даними про враження гостей концертів повноцінно дане рішення використовуватися не може. Наявні відгуки користувачів у даному розрізі можна розглядати лише тільки як обмежену, в певному сенсі ознайомчу можливість, яка є додатковою до основної функціональності, але ніяк не є основною сама по собі.

Програмне рішення setlist.fm [4] (рис. 1.2) дозволяє зазирнути в інформацію про концерти глибше, адже за кожним концертом пропонується переглянути сетлист, який є основою роботи даного застосунку.

За кожним музикантом в застосунку можна переглянути перелік його минулих та майбутніх концертів, а за концертом, що відбувся, відповідно – сетлист. За концертом і відповідним сетлистом можна створювати власні коментарі, що відсутньо у попереднього рішення.

З концерту можна перейти до концертного майданчика, на сторінці якого можна переглянути перелік концертів, який на ньому відбувається.

Пошук виконується за музикантом, за концертним залом, за назвою туру. Окремо надаються розділи, які дозволяють переглянути найпопулярніших музикантів, концертні зали, фестивалі. Також сайт надає доступ до розділу новин, дозволяє переглядати перелік музикантів, які дали найбільшу кількість концертів за рік, а також спілкуватися на форумі. Спілкування на форумі відбувається щодо можливостей вебзастосунку.

Кожному користувачу, коректно авторизованому, надається можливість вносити і редагувати сетлисти.

setlist.fm
* THE SETLIST WIKI *

Search for Artist, Venue, Tour, ...

FOLLOW [Twitter] [Facebook] [Instagram]

SETLISTS ARTISTS FESTIVALS VENUES STATISTICS NEWS FORUM + ADD SETLIST SIGN IN [Facebook] [Twitter] [Google+]

You are here: [setlist.fm](#) > [Artists](#) > [R](#) > [Red Hot Chili Peppers](#) > November 2, 2019 Setlist

NOV 2 2019 **Red Hot Chili Peppers Setlist**
at [Silverlake Conservatory of Music, Los Angeles, CA, USA](#) | [Edit setlist](#)

Setlist SHARE SETLIST [Twitter] [Facebook]

1. Intro Jam
2. Can't Stop
3. The Zephyr Song
4. Dark Necessities
5. Purple Stain
(first time since 2004)
6. Don't Forget Me
7. Suck My Kiss
8. Soul to Squeeze
9. By the Way
10. Purple Rain
(Prince cover) (with [Eddie Vedder](#))
11. All Along the Watchtower
(Bob Dylan cover) (with [Eddie Vedder](#))

Note: "Snow ((Hey Oh))" was listed on the written setlist as an alternate to "Dark Necessities". This was Josh Klinghoffer's final show before leaving the band on December 15th, 2019.

[I WAS THERE](#) 2 setlist.fm users were there [Edit setlist](#)

Edits and Comments
22 activities (last edit by [allenz](#) 16 Dec 2019, 00:51)

16 Dec 2019, 02:48 [allenz](#) Roadie I was referencing the fact that this is going to be his third time being a member of the band, but I understand what you mean. Yeah, I agree. While I'm happy to hear that John is back in, it doesn't

More from Red Hot Chili Peppers
[More Setlists](#)
[Artist Statistics](#)
[Add setlist](#)

[VIEW GALLERY \(18\)](#)

setlist.fm

Рисунок 1.2 – Інтерфейс вебсайту setlist.fm

Таким чином, даний вебсайт орієнтований на роботу з сетлистами концертів. За бажання користувач може залишити в коментарі власне враження про концерт, проте коментарі рідко використовуються в застосунку саме для цього. Робота з сетлистами загалом є важливою для сприйняття турів і концертів користувачами, проте відсутність зручних засобів для обміну власними враженнями відвідувачів концертів звужує сферу застосування цієї інформації.

Bandsintown [5] є ще одним програмним рішенням, що може використовуватися для пошуку концертів (рис. 1.3).

Glass Animals

Thomas Wolfe Auditorium

Mar. 28th, 2022

1,402 fans interested

I Was There

Leave a Review



Thomas Wolfe Auditorium

87 Haywood St, Asheville, NC 28801

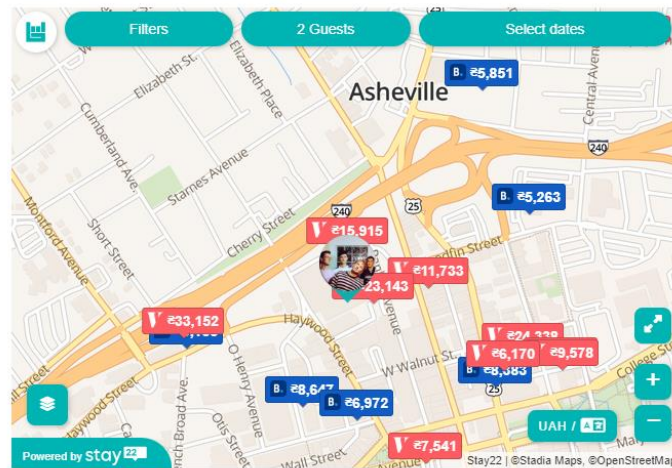


Mar. 28th, 2022

7:00 PM



Find a place to stay



SHARE EVENT



About Glass Animals

DREAMLAND THE ALBUM OUT NOW

Find out more



Live Photos of Glass Animals



Fan Live Reviews



Hunter

April 2nd 2022



WAVEY DAVEYS ON FIRE!!! He also preformed, Life Itself, Gooley, Pork Soda, and Take a slide, among many of the dreamland songs! He didn't play dreamland which was a bit unfortunate but The energy the whole time was amazing ⚡⚡ Considering going to the next show in Nashville 🍌 !!

Рисунок 1.3 – Інтерфейс вебсайту Bandsintown

Bandsintown надає зручний спосіб знайти як невеликі місцеві концерти, так і концерти відомих виконавців, які приїжджають у район користувача, використовуючи наступні засоби:

- робота з програмою також починається з перегляду музичної бібліотеки пристрою або будь-якого з ряду джерел, включаючи Facebook, щоб знайти виконавців, які користувача цікавлять: програма почне відстежувати концерти за участю цих виконавців, а також подібні, і користувач зможе вручну переглядати за жанром або шукати додаткових виконавців, на які слід підписатися;

- програма об'єднує концерти з підписок в повний календар концертів у регіоні з учасниками, яких користувач відстежує, та подібними виконавцями, також є можливість шукати всі місцеві події за музичними жанрами [3].

В розрізі даного аналізу важливим також є те, що програма надає можливість залишити власне враження про концерт. При цьому в даному випадку враження стосується саме конкретного концерту, а не музиканта загалом. Проте слід відзначити, що не можна перейти на профіль користувача, не можна відслідкувати його інші враження. Окрім того враження достатньо обмежені, адже вони передбачають тільки одну оцінку та опис.

1.2 Порівняння програмних аналогів

Основні виявлені в підрозділі 1.1 рішення були виділені для порівняння програмних аналогів. До таких аналогів було винесено Songkick, Setlist.fm, Bandintown. Вони порівнюються з програмою Musimpress, яка розробляється в роботі. Результати порівняння були зведені до табл. 1.1.

В якості критеріїв порівняння програм було виділено наступні:

- наявність можливостей пошуку артистів;
- наявність можливостей пошуку концертів;
- можливість отримання інформації про сетлист концерта, що є спеціалізованою інформацією, який дозволяє отримати загальне уявлення

про програму концерту та концертів певного туру загалом, де зазвичай музичні треки повторюються;

- створення власних вражень про музичні концерти їх відвідувачами;
- підписка на користувачів, які залишають власні враження про музичні концерти;
- оцінка концерту за критеріями, коли замість виставлення узагальненої оцінки та створення загального опису конкретно деталізуються описи і оцінки за окремими критеріями, включаючи звук, рівень шоу тощо.

Таблиця 1.1 – Порівняння програмних аналогів

Критерій	Songkick	Setlist.fm	Bandintown	Musimpress
Пошук артистів	+	+	+	+
Пошук концертів	+	+	+	+
Перегляд сетлиста концерту	—	+	—	+
Створення власних вражень	За музикантом	—	За концертом	За концертом
Підписка на користувачів	—	—	—	+
Оцінка концерту за критеріями	—	—	—	+

1.3 Висновки за розділом 1

У результаті аналізу програмних аналогів, включаючи Songkick, Setlist.fm, Bandintown, виявлено недостатню кількість інструментів обміну враженнями відвідувачів концертів для залучення користувачів програм у поточний стан індустрії живих виступів у програмних рішеннях, які використовуються для пошуку або отримання інформації про музичні

концерти. Саме на розв'язання цієї проблеми направлена розробка, що має виконуватися в роботі.

Завдання роботи:

- дослідження предметної області;
- визначення вимог до програмного забезпечення обміну враженнями від музичних концертів;
- проєктування програмного забезпечення;
- реалізація програмного забезпечення.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір мови програмування

Для розробки програми запропоновано було мови програмування Java [6] і Python [7]-[8], адже вони мають зручні засоби роботи з даними, є доволі потужними і дозволяють реалізовувати вебзастосунки.

До основних критеріїв порівняння віднесено кросплатформність, адже це дозволяє запускати вебсистему на різних серверах, можливість веброзробки за допомогою готових фреймворків, що спрощує розробку, динамічна типізація, швидке написання коду, швидке виконання коду та швидкий запуск, що загалом впливають на час роботи з програмною системою.

Порівняння мов (табл. 2.1) дозволило за сукупністю обрати мову Python.

Таблиця 2.1 – Вибір мови програмування для розробки програми

Критерій	Java	Python
Кросплатформність	+	+
Веброзробка за допомогою готових фреймворків	+	+
Динамічна типізація	—	+
Швидке написання коду	—	+
Швидке виконання коду	+	+—
Швидкий запуск	+—	+

2.2 Функціональні вимоги до програмного забезпечення

Функціональні вимоги до програмного забезпечення є основою подальшого визначення його структури, структури даних, створення інтерфейсу та реалізації. До їх складу було внесено наступні:

- відображення даних концерту;
- пошук музикантів за назвою, країною, інтервалом отриманих оцінок за концерти;
- створення нового концерту;
- оновлення інформації про концерт;
- створення нового туру музикантів;
- відображення переліку треків, які найчастіше виконував музикант на концертах;
- оновлення інформації про тур музикантів;
- створення нового концертного залу;
- оновлення інформації про концертний зал;
- пошук концертів за наступними фільтрами: містом проведення, інтервалом дати проведення, іменем музиканта, інтервалом оцінки (для проведених концертів);
- відображення переліку концертів у турі музиканта;
- відображення концертних залів;
- підписування на враження користувача;
- перегляд профілю користувача;
- відписування під вражень користувача;
- відображення вражень про музичні концерти користувачів, на які оформлено підписку;
- відображення переліку концертів, на яких виконувався певний музичний трек;
- відображення найкращих турів;
- відображення найкращих концертів;

- відображення майбутніх концертів музиканта;
- відображення концертів музиканта, які вже відбулись;
- відображення найкращих концертів у концертному залі;
- відображення інформації про концертний зал;
- авторизація користувачів;
- реєстрація користувачів;
- відображення інформації про музиканта;
- створення власного враження про музичний концерт;
- створення коментаря до враження про музичний концерт іншого користувача;
- відображення коментарів до враження про музичний концерт;
- створення нового музиканта;
- оновлення інформації про музиканта;
- відображення переліку музикантів, концерти яких було найвище оцінено користувачами;
- відображення власних вражень від концертів;
- оновлення власних вражень від концертів;
- відображення переліку турів музиканта;
- відображення майбутніх і минулих концертів у концертному залі;
- створення сетлиста музичного концерту;
- оновлення сетлиста музичного концерту;
- відображення музикантів за стилем музики.

При визначенні сценаріїв роботи користувачів з програмою спочатку було виділено серед користувачів неавторизованого відвідувача, авторизованого відвідувача, менеджера музиканта та менеджера вебсистеми.

До зони відповідальності менеджера вебсистеми належить створення запису музиканта, пошук записів музикантів, визначення менеджера музиканта, який відповідає за роботу з даним музикантом, реєстрація такого менеджера, створення запису концертного залу, авторизація та відображення

концертних залів. На рис. 2.1 представлено розроблені відповідні сценарії для менеджера вебсистеми.

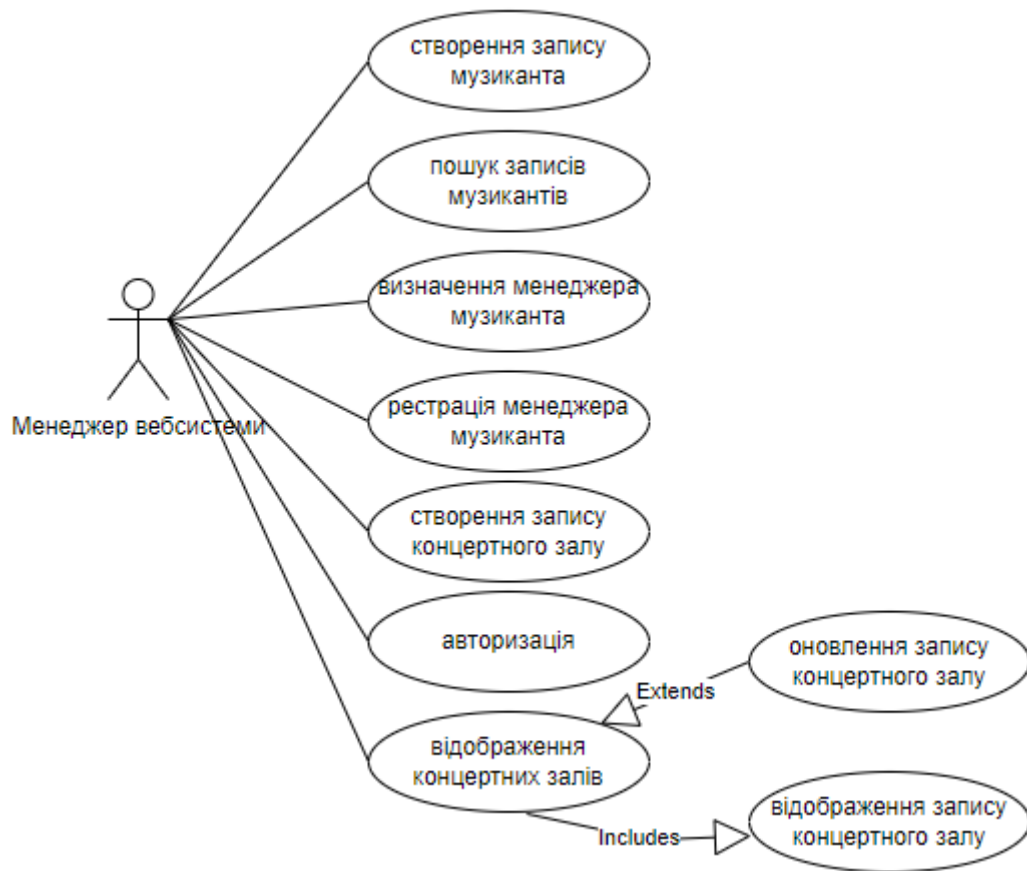


Рисунок 2.1 – Діаграма прецедентів менеджера вебсистеми

Сценарії роботи неавторизованого відвідувача є основою сценаріїв роботи авторизованого відвідувача. Вони включають відображення найкращих музикантів, найкращих турів, найкращих концертів, пошук концертів, реєстрацію відвідувачів, пошук музикантів, відображення музикантів за стилем музики, перегляд профілю користувача, перегляд конкретного туру музиканта, перегляд інформації про музиканта, відображення даних конкретного концерту, відображення концертних залів (рис. 2.2).

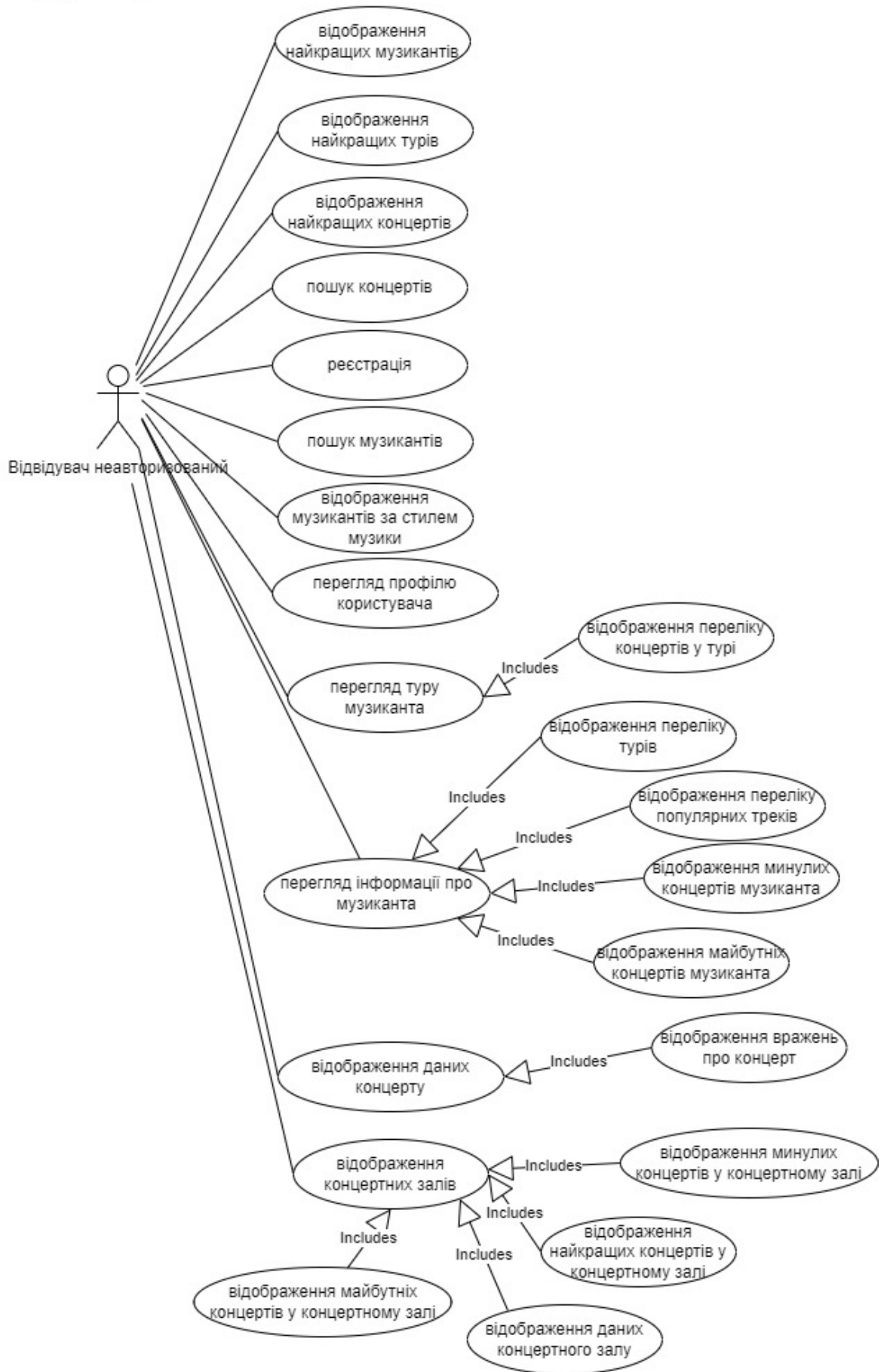


Рисунок 2.2 – Діаграма прецедентів неавторизованого відвідувача

На рис. 2.3 представлено відповідні сценарії роботи авторизованого відвідувача.



Рисунок 2.3 – Діаграма прецедентів авторизованого відвідувача

Сценарії роботи менеджера музиканта було об'єднано в три групи, що відповідають роботі з даними власних музикантів, роботі з турами власних музикантів та роботі з концертами власних музикантів. На рис. 2.4 представлено розроблені відповідні сценарії для менеджера музиканта.



Рисунок 2.4 – Діаграма прецедентів менеджера музиканта

2.3 Проектування бази даних

Зважаючи на необхідність доступу користувачів до вебзастосунку віддалено з пристроїв, які можуть значно відрізнятися один від одного, було вирішено створити базу даних (БД) для роботи з даними. Системою керування БД стала MySQL, що виділяється з одного боку тим, що є безкоштовною, а з іншого – якісним і надійним застосуванням.

До сутностей, які є основою БД, належать:

- музичний концерт;
- музикант;
- сетлист, що представляє собою перелік треків, який виконав на концерті музикант;
- враження від музичного концерту;
- коментар до враження;
- профіль користувача;
- підписка на користувача;
- стиль музики;
- концертний зал;
- музичний трек;
- тур, що включає перелік концертів (кожен концерт відбувається в деякому турі, а концерти без турів відділені в окремий загальний тур для зручності збереження даних);
- місто.

Таблиця `musimpress_city` потрібна для визначення структури міст, в яких відбуваються концерти і відповідно знаходяться концертні зали або майданчики:

- `id` – номер міста;
- `name` – назва міста;
- `country` – країна, в якій розташоване місто.

Таблиця `musimpress_hall` потрібна для визначення концертних залів або майданчиків, де відбуваються концерти:

- `id` – номер концертного залу;
- `hname` – назва концертного залу;
- `place` – адреса, за якою розташовано концертний зал у місті;
- `city_id` – місто, в якому розташовано концертний зал (відповідає таблиці `musimpress_city`);
- `fulltext` – опис концертного залу.

Таблиця `musimpress_concert` потрібна для визначення самого музичного концерту:

- `id` – номер музичного концерту;
- `tour_id` – тур музиканта, в межах якого відбувається концерт (відповідає таблиці `musimpress_tour`);
- `hall_id` – концертний зал, в якому відбувається концерт;
- `timepoint` – дата, час, коли почанається концерт;
- `score` – середня оцінка, виставлена за даний концерт;
- `fulltext` – опис концерту;
- `songs` – кількість пісень, виконана на концерті.

Таблиця `musimpress_tour` потрібна для визначення туру музиканта, в межах якого проходить концерт (за відсутності туру всі такі концерти об'єднуються в один загальний тур):

- `id` – номер туру;
- `tname` – назва туру;
- `startdate` – дата початку концертів у межах туру;
- `enddate` – дата завершення концертів у межах туру;
- `score` – середня оцінка, виставлена за даний тур;
- `fulltext` – опис туру.

Таблиця `musimpress_tour_musician` реалізує зв'язок багато до багатьох між таблицями музикантів та турів. Між ними саме такий зв'язок через те, що музиканти можуть впродовж кар'єри організовувати різні тури, про те в

кожному окремому турі можуть брати участь різні музиканти. Прикладом останнього може бути тур спільних концертів або музичний фестиваль.

Склад таблиці `musimpress_tour_musician`:

- `id` – номер;
- `musician_id` – музикант (відповідає таблиці `musimpress_musician`);
- `tour_id` – тур (відповідає таблиці `musimpress_tour`).

Таблиця `musimpress_style` потрібна для визначення стилю музиканта:

- `id` – номер стилю;
- `sname` – назва стилю.

Таблиця `musimpress_musician_style` реалізує зв'язок багато до багатьох між таблицями музикантів та стилів. Між ними саме такий зв'язок через те, що музиканти можуть грати в різних стилях, а кожен стиль зазвичай використовується різними музикантами у свою чергу.

Склад таблиці `musimpress_musician_style`:

- `id` – номер;
- `musician_id` – музикант (відповідає таблиці `musimpress_musician`);
- `style_id` – тур (відповідає таблиці `musimpress_style`).

Таблиця `musimpress_musician` потрібна для визначення музиканта:

- `id` – номер музиканта;
- `mname` – ім'я музиканта або назва колективу;
- `country` – країна, з якої походить музикант;
- `score` – середня оцінка, виставлена даному музиканту користувачами.

Таблиця `musimpress_setlist` потрібна для визначення сетлиста концерту:

- `id` – номер сетлиста;
- `concert_id` – концерт, до якого відноситься сетлист (відповідає таблиці `musimpress_concert`);
- `track_id` – конкретний музичний трек, який було виконано на концерті і який внесено в сетлист;

- `comusician` – співвиконавець треку на концерті;
- `order` – місце даного музичного треку в сетлісті.

Таблиця `musimpress_impression` потрібна для визначення вражень користувача про музичний концерт:

- `id` – номер враження;
- `concert_id` – музичний концерт (відповідає таблиці `musimpress_concert`);
- `profile_id` – профіль користувача, за яким було сформульовано враження (відповідає таблиці `musimpress_profile`);
- `timepoint` – дата, час, коли було залишено запис;
- `fulltext` – текст запису;
- `score` – оцінка, виставлена за враженнями від концерту користувачем;
- `energyscore` – оцінка за енергійність, виставлена концерту;
- `energytext` – опис оцінки за енергійність виступу;
- `soundscore` – оцінка за рівень звуку, влаштованого на концерті;
- `soundtext` – опис оцінки про рівень звуку на концерті (окрім деталізації може також бути визначено місце, звідки концерт слухав користувач);
- `dancescore` – оцінка за те, на скільки концерт був розважальним (танцювальність, позитивні емоції);
- `dancetext` – опис оцінки за розважальність концерту;
- `showscore` – оцінка за рівень шоу на концерті (немузична складова);
- `showtext` – опис оцінки за рівень шоу на концерті;
- `managementscore` – оцінка за рівень менеджменту на концерті (питання організації, з якими зіштовхувався користувач);
- `managementtext` – опис оцінки за рівень менеджменту.

Підсумкова схема, що визначає структуру БД, представлена на рис. 2.5.

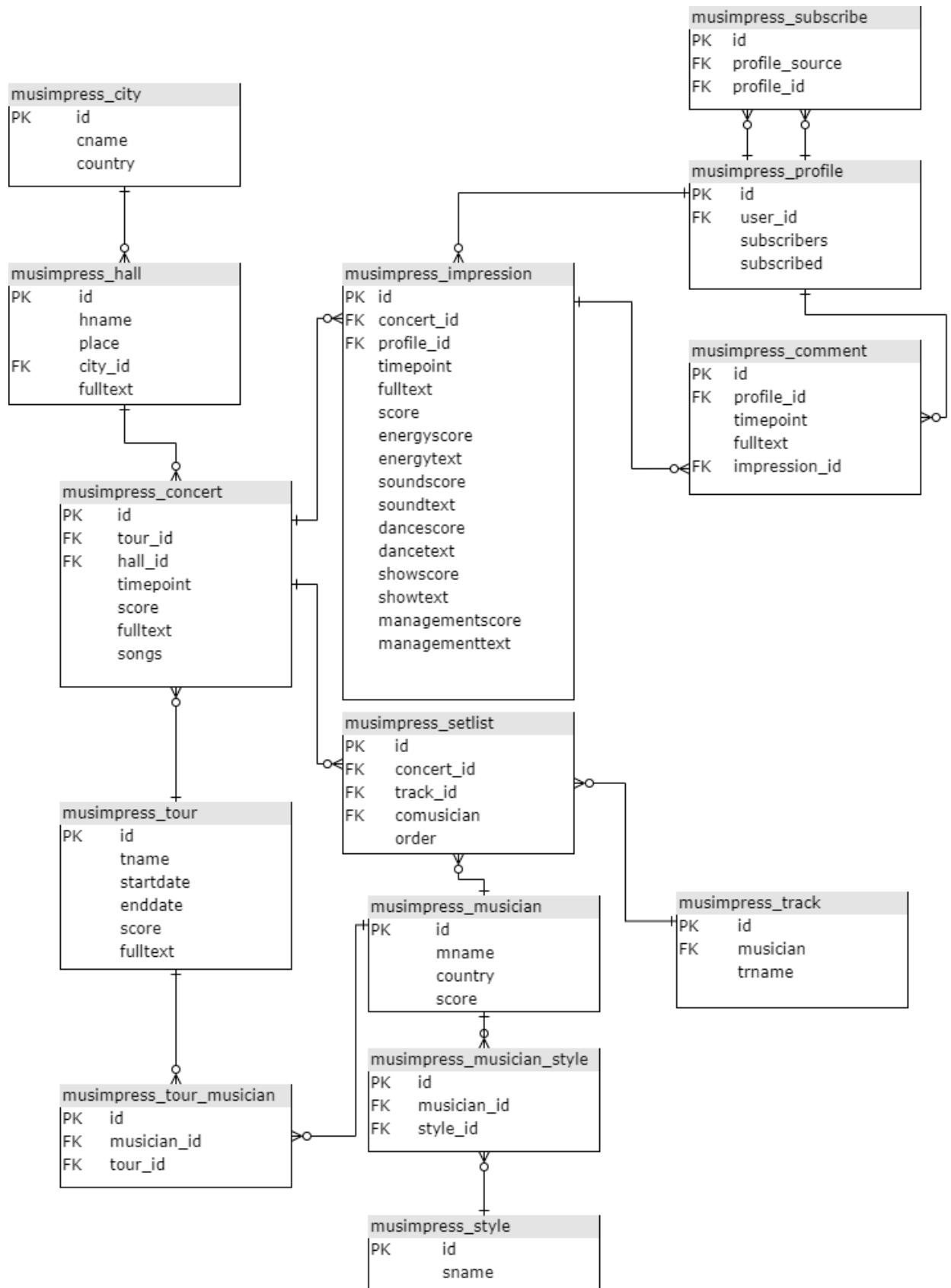


Рисунок 2.5 – Схема БД

Таблиця `musimpress_profile` потрібна для визначення профілю користувача:

- `id` – номер профілю;
- `user_id` – обліковий запис Django, на якого підписався користувач;
- `subscribers` – кількість підписників користувача;
- `subscribed` – кількість підписок користувача.

Таблиця `musimpress_track` потрібна для визначення музичного треку:

- `id` – номер треку;
- `musician` – музикант, який виконує треку (відповідає таблиці `musimpress_musician`);
- `tname` – назва треку.

Таблиця `musimpress_subscribe` потрібна для визначення підписки користувача на відслідковування вражень іншого користувача про музичні концерти:

- `id` – номер підписки;
- `profile_source` – профіль користувача, на якого підписався цей користувач (відповідає таблиці `musimpress_profile`);
- `profile_id` – профіль підписника (відповідає таблиці `musimpress_profile`).

Таблиця `musimpress_comment` потрібна для визначення коментаря до враження про музичний концерт:

- `id` – номер коментаря;
- `profile_id` – профіль користувача, який залишив дане враження (відповідає таблиці `musimpress_profile`);
- `timepoint` – дата, час, коли було залишено даний коментар;
- `fulltext` – повний текст коментаря;
- `impression_id` – враження про музичний концерт, щодо якого було залишено коментар (відповідає таблиці `musimpress_impression`).

2.4 Висновки за розділом 2

Визначено основні принципи створення застосунку, що стосуються етапу його проєктування: обрано мову Python і відповідний стек засобів веброзробки для реалізації програми, визначено сценарії роботи користувачів з програмою, виконано проєктування схеми роботи з даними програми.

З РОЗРОБКА ПРОГРАМИ

3.1 Структура розробленої вебсистеми

Вебсистему розроблено на основі використання фреймворку Django, який в основі побудови використовує шаблон Model-View-Template (MVT) [9]-[10]. Використання шаблону MVT вплинуло безпосередньо на структуру вебсистеми, яку було розроблено в результаті (рис. 3.1).

Вебсистема складається з двох підсистем: підсистеми планування розрахунків, підсистеми відображення музичних концертів.

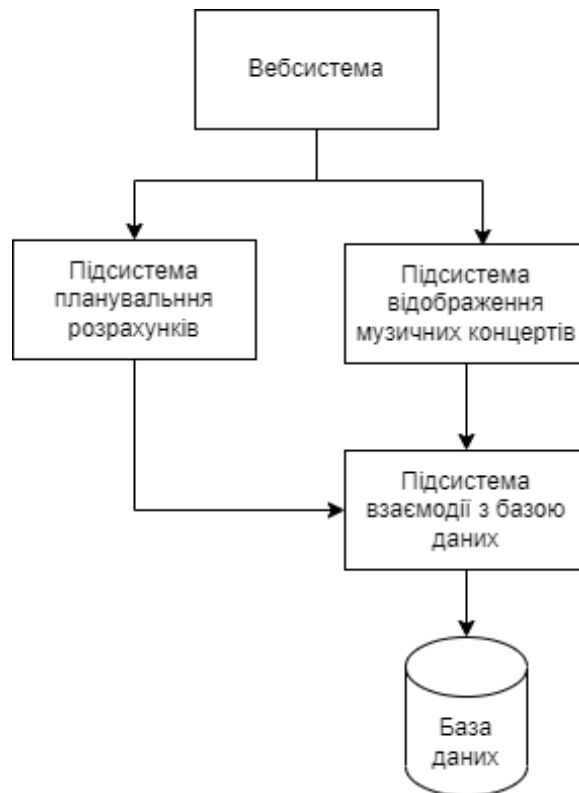


Рисунок 3.1 – Структурна схема розробленої вебсистеми

Підсистема планування розрахунків виконує обчислення середніх балів музикантів, концертів, турів шляхом запуску таких розрахунків раз на день.

Підсистема відображення музичних концертів відповідає за роботу безпосередньо з застосунком на основі шаблону MVT. При цьому роботу з

даними виділено в підсистему взаємодії з БД, а всю іншу частину, представлену в основному уявленнями і шаблонами, виділено в підсистему відображення музичних концертів.

3.2 Опис реалізованих класів програми

Для реалізації вебсистеми було у відповідності зі структурною схемою розроблено набір класів і скриптів для реалізації відповідних можливостей.

Під час реалізації моделей вебсистеми було використано підхід “код спочатку”. Тобто було створено код класів моделей, на основі яких автоматично створювались відповідні таблиці БД.

Під час реалізації шаблонів було використано шаблон [11], на основі якого для кожної функціональної вимоги було створено вебсторінку. Заповнення даними кожного такого шаблону реалізовувалось на основі створених уявлень вебсистеми. Уявлення включали список розроблених класів.

Клас ConcertSearchView забезпечує пошук концертів музикантів на основі створення списку концертів, що відбуваються у заданому користувачем місті, в проміжку між заданими датами, а також в яких бере участь музикант, ім'я якого задано під час пошуку. Заданий музикант повинен бути учасником туру, в межах якого відбувається концерт. Даний клас є універсальним, він підтримує роботу як з афішованими концертами, так і з архівними. Оскільки архівні концерти мають вже оцінку від користувачів за їх враженнями, то додатковим параметром під час пошуку концертів є інтервал балів, у які в середньому було оцінено даний концерт.

Клас ConcertView забезпечує деталізацію інформації про концерт з наповненням списку концертів, які відбуваються в тому ж самому турі, що і поточний, сортуючи даний список за датою концерту.

Клас `NewConcertView` забезпечує створення нового концерту на основі відповідної форми з заповненням полів дати і часу початку концерта, повного опису, концертного залу та туру, в межах якого відбувається концерт.

Клас `UpdateConcertView` забезпечує оновлення даних існуючого концерту з можливим оновленням полів дати і часу початку концерта, повного опису, концертного залу та туру, в межах якого відбувається концерт.

Клас `MusicianConcertsView` забезпечує заповнення списку майбутніх концертів музиканта, тобто концертів, які розпочнуться пізніше за поточний момент часу.

Клас `ArchiveConcertsView` забезпечує заповнення списку минулих концертів музиканта, тобто концертів, які вже відбулись на поточний момент часу, сортуючи їх від останнього концерту за віддаленням від поточної дати і часу.

Клас `BestConcertsView` забезпечує створення списку 100 найкращих концертів за їх середньою оцінкою.

Клас `TrackConcertsView` забезпечує створення списку концертів, на яких виконувався заданий музичний трек, сортуючи їх, починаючи від останнього концерту.

Клас `TourView` забезпечує заповнення інформації про музичний тур виконавця.

Клас `BestToursView` забезпечує заповнення списку 100 найкращих турів за їх середньою оцінкою.

Клас `NewTourView` забезпечує створення нового туру музиканта на основі відповідної форми з заповненням полів назви туру, дати початку і дати завершення туру, повного опису.

Клас `UpdateTourView` забезпечує оновлення даних існуючого туру музиканта.

Клас NewHallView забезпечує створення нового концертного залу на основі відповідної форми з заповненням полів назви залу, адреси, міста розташування та повного опису.

Клас HallBestConcertsView забезпечує заповнення списку 10 найкращих концертів, які відбувались в даному концертному залі.

Клас UpdateHallView забезпечує оновлення даних концертного залу на основі відповідної форми з можливим оновленням назви залу, адреси, міста розташування та повного опису.

Клас NewSetlistView забезпечує створення нового запису сетлиста на основі відповідної форми з заповненням полів концерту, на якому було виконано трек, музичного треку, який було виконано, співвиконавця під час цього виступу та позиції цього треку в загальному сетлисті.

Клас UpdateSetlistView забезпечує оновлення даних сетлиста на основі відповідної форми з можливим оновленням концерту, треку, співвиконавця та порядку треку в сетлисті.

Клас MostMusicianTracksView забезпечує заповнення списку треків, які виконує музикант, сортуючи його за кількістю разів виконання на музичному концерті.

Клас BestMusicianConcertsView забезпечує заповнення списку 100 перших музикантів, які виступили на концертах з найвищою середньою оцінкою. Для цього спочатку створюється рейтинг концертів, а далі по одному додаються до списку музиканти, які брали участь в концертах, починаючи з першого і так далі до моменту, коли в списку буде 100 музикантів.

Клас HallView деталізує дані концертного залу.

Клас AlphabetHallsView заповнює список концертних залів, сортуючи їх за назвою для виведення за алфавітом політерно.

Клас HallArchiveConcertsView заповнює список архівних концертів, тобто концертів, які вже відбулись в даному концертному залі.

Клас HallConcertsView заповнює список майбутніх концертів, тобто концертів, які на поточний момент ще не відбулись в даному концертному залі.

Клас MusicianSearchView забезпечує пошук музикантів на основі створення списку музикантів, які мають задане ім'я, середню оцінку в заданому проміжку, походять з заданої країни.

Клас MusicianView деталізує інформацію про музиканта.

Клас NewMusicianView забезпечує створення нового запису музиканта на основі відповідної форми з заповненням полів імені музиканта, країни його походження.

Клас UpdateMusicianView забезпечує оновлення запису музиканта на основі відповідної форми з оновленням полів імені музиканта, країни його походження.

Клас BestMusiciansView забезпечує заповнення списку 100 найкращих музикантів, сортуючи їх за середнім балом.

Клас StyleMusiciansView забезпечує заповнення списку музикантів, які виконують музику в заданому стилі, сортуючи їх за алфавітом.

Клас NewImpression створення нового враження про музичний концерт на основі відповідної форми з заповненням полів концерту, загального тексту і оцінки, тексту і оцінки енергії, звуку, розваг, шоу та менеджменту. Автоматично вносяться дата і час створення та автор враження – авторизований користувач.

Клас UpdateImpression забезпечує оновлення враження про музичний концерт на основі відповідної форми з оновленням полів загального тексту і оцінки, тексту і оцінки енергії, звуку, розваг, шоу та менеджменту.

Клас UserImpressions забезпечує заповнення списку враженнями заданого відвідувача.

Клас SubscribedImpressions забезпечує заповнення списку враженнями відвідувачами, на яких підписаний авторизований відвідувач.

Клас `ImpressionComments` забезпечує заповнення списку коментарів до враження.

Клас `NewComment` забезпечує створення нового коментаря до враження музиканта на основі відповідної форми з заповненням тексту коментаря і автоматичним внесенням дати і часу створення та автора коментаря – авторизованого користувача.

3.3 Висновки за розділом 3

Вебсистему розроблено на основі використання вебфреймворку Django зі структуризацією на основі шаблону проєктування MVT. Розроблено класи, що відповідають моделям, уявленням вебсистеми, скрипти обчислення середніх оцінок за розкладом та шаблони вебсторінок.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма призначена для пошуку музичних концертів за враженнями відвідувачів. У розрізі цього призначення надаються інструменти пошуку концертів та слідкування за враженнями відвідувачів концертів.

4.2 Умови виконання програми

Мінімальний рівень технічних засобів для забезпечення роботи програмної системи включає процесор не менш ніж 4 ядра і частотою 2,9 ГГц, оперативну пам'ять у 8 Гб, жорсткий диск у 1 Тб.

4.3 Тестування програми

Під час тестування програми було виконано перевірку роботи її функцій у відповідності зі списком функціональних вимог з розділу 2 для кожного типу користувачів. Перевірка роботи відбувалась окремо на мобільному пристрої та окремо на стаціонарному. Тестування підтвердило виконання зазначених функціональних вимог.

Неавторизований відвідувач за результатом звернення до вебсервера отримує доступ до домашньої сторінки програми (рис. 4.1). На даній сторінці окрім загальної інформації доступна кнопка “Реєстрація”, що дозволяє відкрити сторінку реєстрації облікового запису користувача або переключитися з неї на авторизацію облікового запису. Також на даній сторінці доступне стандартне меню неавторизованого відвідувача, що містить:

- “Дім” – пункт меню для перегляду домашньої сторінки;
- “Музиканти” – пункт меню для роботи з музикантами, що включає підпункт “Пошук” для організації пошуку музикантів, включаючи задану

назву, інтервал загальної оцінки та країну музиканта, “Кращі” для перегляду музикантів з найкращим рейтингом загалом, “Найкращі виступи” для перегляду музикантів (відображаються фотографія музиканта, назва артиста, середня оцінка за всіма концертами та кнопка “Детальніше” для відкриття сторінки музиканта), які організували окремі найкращі за рейтингом концерти (рис. 4.2);

- “Тури”, що дозволяє відобразити найкращі тури музикантів;

- “Концерти” – пункт меню для роботи з концертами, що включає підпункт “Пошук” для пошуку концертів, підпункт “Кращі” для перегляду списку найкращих концертів;

- “Зали” – пункт меню для роботи з концертними залами, що включає підпункт “Пошук” для пошуку залів (потрібно обрати країну, місто і зі списку задати конкретний зал, а у відповідь відкривається доступ до сторінки концертного залу), підпункт “Всі” для відображення концертних залів за алфавітом;

- “Стилі”, що дозволяє переглянути список стилів, а від стиля перейти до музикантів, які грають музику в даному стилі.

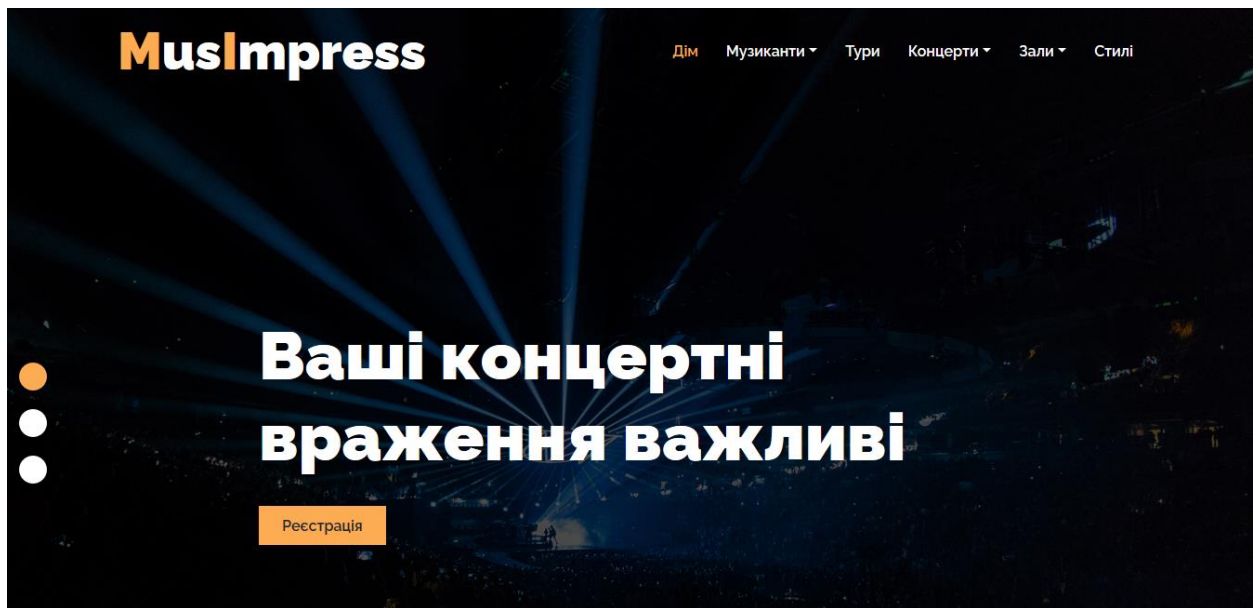


Рисунок 4.1 – Домашня сторінка неавторизованого відвідувача вебсистеми

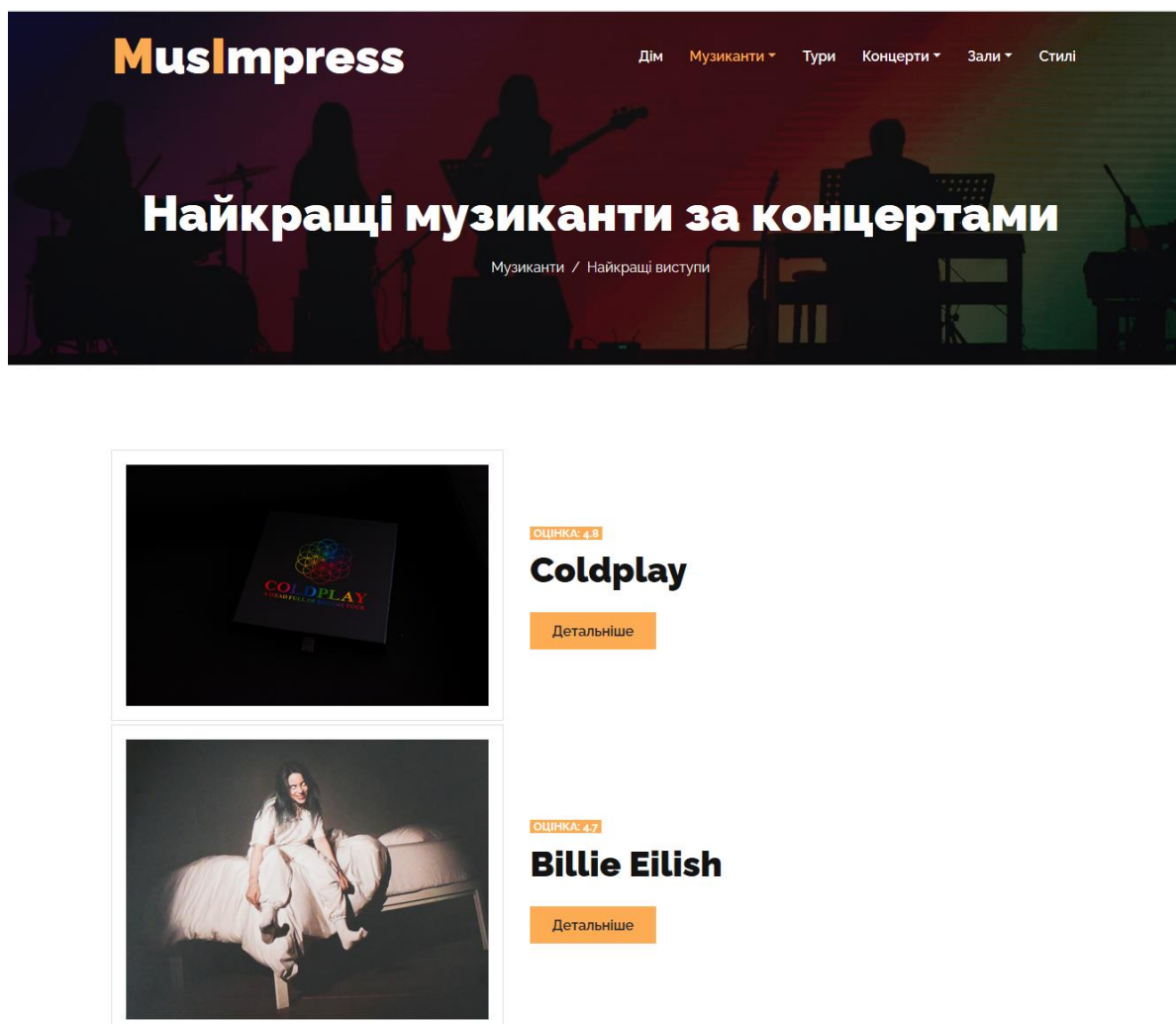


Рисунок 4.2 – Відображення неавторизованому відвідувачу найкращих музикантів за концертами

Після авторизації меню програми змінюється.

Якщо в програмі авторизовано менеджера музиканта, то йому надається меню з наступних пунктів:

- “Музиканти”, що дозволяє працювати з власними музикантами, формуючи список цих музикантів, звідки натискаючи на відповідну назву, можна перейти до сторінки музиканта з можливостями додавання туру, оновлення існуючих турів, оновлення даних музиканта;

- “Тури”, що дозволяє працювати з турами власних музикантів і включає підпункти “Всі” для перегляду списку всіх турів всіх власних музикантів менеджера, “За музикантом” для вибору конкретного музиканта і відображення списку всіх його турів з наданням кнопок для оновлення

інформації за туром, додавання концертів до туру та відповідно перегляду даних туру за натисканням на посилання в назві туру;

– “Концерти”, що дозволяє працювати з концертами власних музикантів і включає підпункти “Всі” для перегляду списку всіх концертів всіх власних музикантів менеджера, “За музикантом” для вибору конкретного музиканта, його туру і відображення списку всіх концертів з цього туру з наданням кнопок для оновлення інформації за концертом, включаючи внесення або редагування сетлиста до концерту, створення нового концерту в турі (рис. 4.3);

– “Профіль”, що дозволяє редагувати дані власного профілю, а також вийти з власного облікового запису.

MusImpress Музиканти Тури Концерти Профіль

Створення нового концерту

Музиканти / Монатік / Створити концерт

МОНАТІК

Створити новий концерт

Тур музиканта
 Без туру

Місто
 Київ **Задати**

Концертна зала
 Stereo Plaza

10/10/2022 19:00

Опис

Завантажити зображення

Створити концерт

Рисунок 4.3 – Створення нового концерту менеджером музиканта

Під час створення концерту можна задати тур музиканта (або змінити, якщо концерт був створений зі сторінки туру), визначити місто проведення концерту, після чого натиснути на кнопку “Задати”, а тоді зі списку нижче задати конкретну концертну залу. У правому стовпці можна з календаря обрати дату проведення концерту і час його початку, ввести опис концерту, завантажити фотографію шляхом визначення для завантаження файлу, після чого зберегти дані натисканням кнопки “Створити концерт”.

Аналогічним чином влаштовано сторінку оновлення даних концерту.

Якщо авторизувався відвідувач у програмі, то його меню складається з пунктів:

- “Дім” – пункт меню для перегляду домашньої сторінки, яка після авторизації включає також перелік стилів, що дозволяє від стиля перейти до музикантів, які грають музику в даному стилі, а також кнопку виходу з облікового запису;

- “Музиканти” – пункт меню для роботи з музикантами, що включає підпункт “Пошук” для організації пошуку музикантів, включаючи задану назву, інтервал загальної оцінки та країну музиканта, “Кращі” для перегляду музикантів з найкращим рейтингом загалом, “Найкращі виступи” для перегляду музикантів, які організували окремі найкращі за рейтингом концерти;

- “Тури”, що дозволяє відобразити найкращі тури музикантів;

- “Концерти” – пункт меню для роботи з концертами, що включає підпункт “Пошук” для пошуку концертів, підпункт “Кращі” для перегляду списку найкращих концертів;

- “Зали” – пункт меню для роботи з концертними залами, що включає підпункт “Пошук” для пошуку залів (потрібно обрати країну, місто і зі списку задати конкретний зал, а у відповідь відкривається доступ до сторінки концертного залу), підпункт “Всі” для відображення концертних залів за алфавітом;

– “Підписки”, що містить список користувачів, на яких підписався авторизований відвідувач, звідти також можна відписатися від користувача (підписка можлива зі сторінки профілю користувача, тобто побачивши враження користувача, можна натиснути на його логін та перейти на сторінку його профілю, а звідти підписатися);

– “Враження”, що містить підпункти “Власні” для відображення списку власних вражень, “Підписані” для перегляду вражень відвідувачів, на які підписався авторизований відвідувач;

– “Профіль”.

Якщо користувач обрав сторінку музиканта, то йому буде доступною інформація про музиканта, перелік його турів, звідки можна буде перейти до сторінки відповідного туру, що надає можливість обрати потрібний концерт, а також відповідні кнопки для відображення майбутніх концертів музиканта, його минулих концертів, списку його популярних треків, тобто треків, які найчастіше входили в сетлисті концертів музиканта.

Якщо відкрити сторінку з відповідною залом, то окрім інформації про неї буде можливість натисканням відповідних кнопок перейти до відображення найкращих концертів у цьому залі, майбутніх концертів, минулих концертів.

На рис. 4.4 продемонстровано відображення майбутніх концертів у вибраному концертному залі. На сторінці при цьому відображається список концертів, які відбудуться в залі, починаючи з поточної дати. На кожен запис можна натиснути (на назву музиканта та туру) і перейти до сторінки концерту. На сторінці в правій частині також міститься поле пошуку для вибору назви залу (список з вибором шляхом введення), а також список інших концертних залів з цього ж міста.

У свою чергу сторінка конкретного концерту містить безпосередньо сетлист даного концерту, якщо він вже відбувся, а також список вражень відвідувачів з кнопкою “Додати враження” в кінці для переходу до сторінки введення власного враження від концерту.

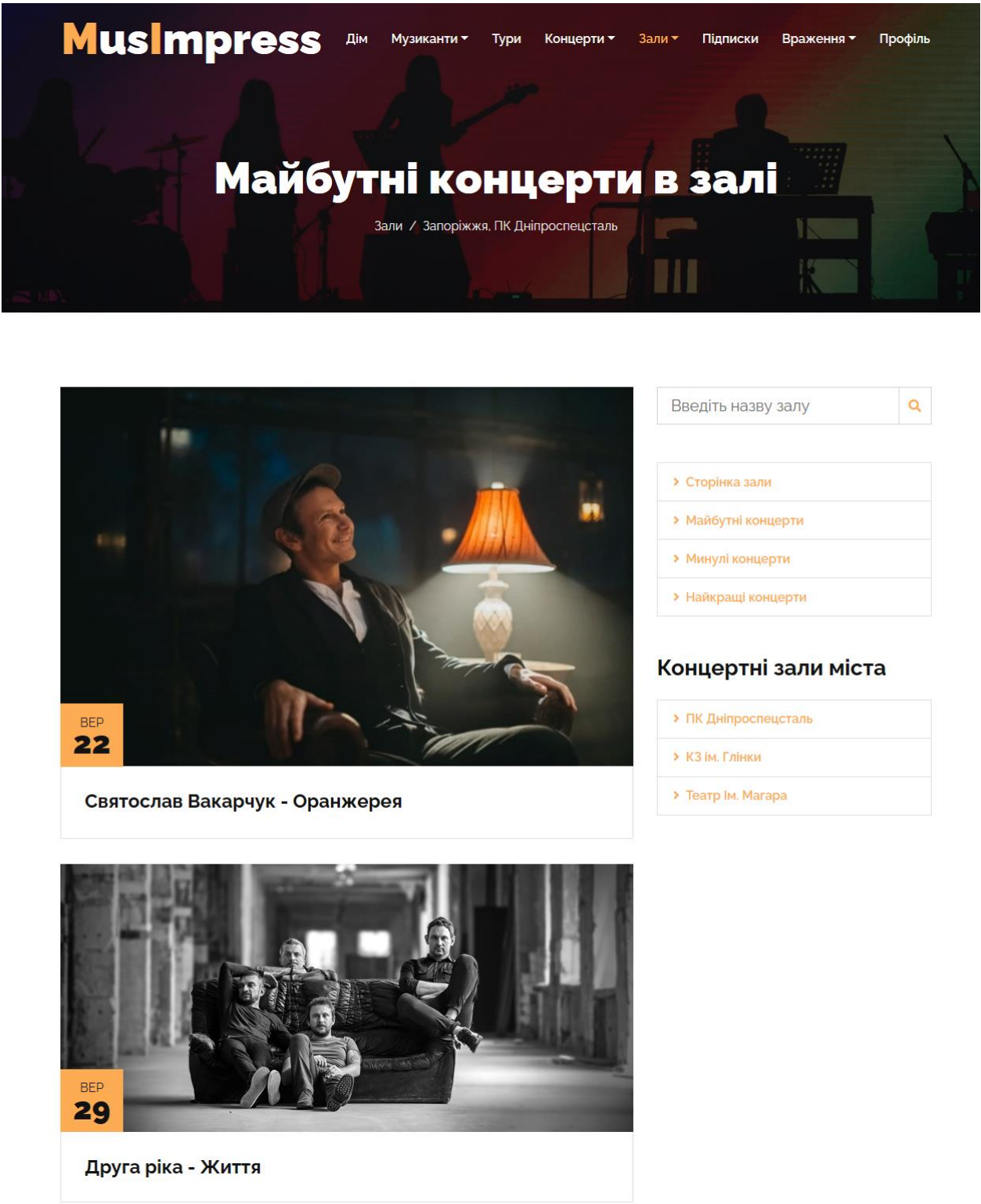
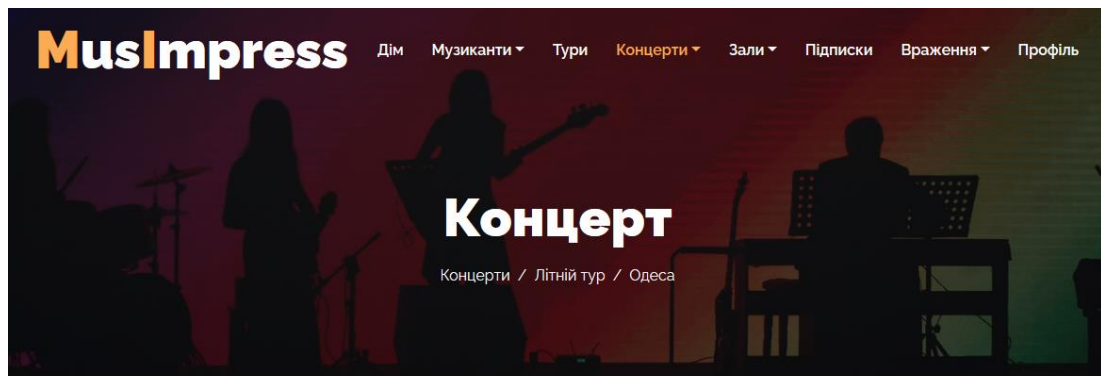


Рисунок 4.4 – Відображення авторизованому відвідувачу концертів у залі

За кожним враженням відображається логін відвідувача, час внесення, повний опис, загальна оцінка, а також окремі описи та оцінки за критеріями (рис. 4.5).



Арт-завод Механіка

Один в каное - Літній тур

Один в каное виступить у Запоріжжі з концертом в турі Осінній, в якому презентуються пісні з нового альбому 2021 року.

Сетлист

1. Хуанхе
2. Кому то тре
3. Будь мені кимось
4. Прекрасний оповідач
5. Розсипаюся
6. У мене немає дому
7. Пам'яті мало
8. Про автора
9. Вовк
10. Лісабон
11. Веснянка
12. Ікони

Враження



petty 15.05.2022 17:18

Загалом: Концерт був неймовірно душевним і теплим. Радість наповнює, коли згадую його.

Оцінка: **5**

Енергія: Енергія відчуттів надприродна, але на танці не розраховуйте.

Оцінка: **5**

Звук: Звук перед сценою і в кінці залу був налаштований прекрасно.

Оцінка: **5**

Розваги: Якщо ви хочете, щоб вас розважали, то звичайно, що на концертах Один в каное вам немає, що робити, але якщо для вас розвагою є розмова з самим собою, то тоді це точно в точку.

Оцінка: **4**

Шоу: Тим, хто був на концертах Один в каное відомо, що це не шоу, а тонкість душі.

Оцінка: **4**

Менеджмент: Організація концерту не викликала жодних проблем.

Оцінка: **5**

Коментарі

Пошук концерту: музикант

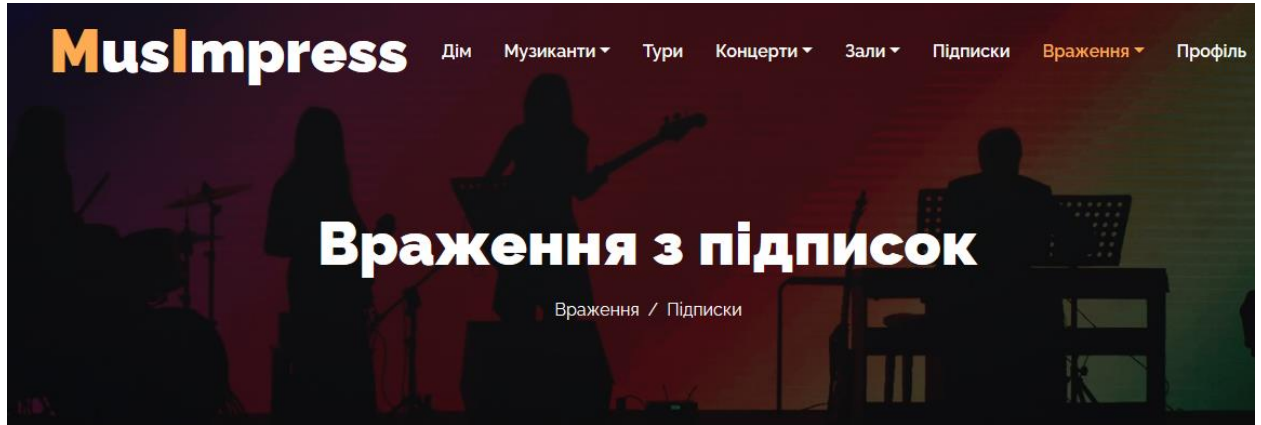


Концерти туру

- > 01.07.2021 Одеса
- > 03.07.2021 Київ
- > 04.07.2021 Львів
- > 07.07.2021 Полтава
- > 13.07.2021 Харків

Рисунок 4.5 – Відображення авторизованому користувачу даних концерта

При виборі авторизованим відвідувачем підпункту меню “Підписані” пункту меню “Враження” відображається на сторінці список вражень відвідувачів, на які підписався даний відвідувач (рис. 4.6).




ram 08.05.2022

Coldplay - Music of the Spheres, Х'юстон

Загалом: Діснейленд для дорослих вартує кожної секунди витраченого на нього вечора. Ви будете з відкритим від щастя та здивування ротом спостерігати, співати та брати участь в концерті.

Оцінка: 5.

Енергія: Море танцювальних треків і провокацій для цього.

Оцінка: 5.

Звук: Прекрасний звук в центрі чаші стадіону.

Оцінка: 5.

Розваги: Величезні м'ячі, прекрасний відеоряд не змусять вас сумувати точно.

Оцінка: 5.

Шоу: Шоу зрежисоване якісно і важко вважати, що в ньому чогось не вистачає.

Оцінка: 5.

Менеджмент: Ви відчуваєте, що про вас турбуються всюди.

Оцінка: 5.

Підписки

- > perry
- > ihor
- > jackie
- > ram

Рисунок 4.6 – Відображення вражень підписок авторизованому відвідувачу

За кожним відвідувачем можна перейти до його профілю, також можна натиснути на відповідний концерт і перейти до його сторінки. За концертом відображається його музикант, назва туру, місто проведення, дата, загальна оцінка і опис до неї, оцінки і описи енергії, звуку, розваг, шоу та менеджменту.

4.4 Висновки за розділом 4

Виконано тестування роботи програми для різних типів користувачів, що забезпечило успішну перевірку реалізації всіх функціональних вимог до програми.

ВИСНОВКИ

У результаті аналізу програмних аналогів, включаючи Songkick, Setlist.fm, Bandintown, виявлено недостатню кількість інструментів обміну враженнями відвідувачів концертів для залучення користувачів програм у поточний стан індустрії живих виступів у програмних рішеннях, які використовуються для пошуку або отримання інформації про музичні концерти. На розв'язання цієї проблеми було вирішено направити розробку програми.

Визначено основні принципи створення застосунку, що стосуються етапу його проєктування: обрано мову Python і відповідний стек засобів веброзробки, включаючи фреймворк Django, для реалізації програми, визначено сценарії роботи користувачів з програмою, включаючи неавторизованого відвідувача, авторизованого відвідувача, менеджера музиканта, менеджера вебсистеми, виконано проєктування схеми роботи з даними програми через створення відповідної БД.

Реалізовано програму за допомогою використання фреймворку Django, що дозволяє виконувати пошук концертів, музикантів, відображати найкращих музикантів, тури, концерти, музикантів за стилем музики, переглядати тури музикантів, інформацію про музикантів, включаючи перелік турів, популярні треки, минулі концерти музиканта, майбутні концерти музиканта, дані конкретних концертів, включаючи відображення вражень про концерт, переглядати дані концертних залів, включаючи майбутні, минулі концерти, найкращі концерти, підписуватися на враження відвідувачів, створювати власні враження про концерт, створювати коментарі до вражень, переглядати враження про концерти від відвідувачів, на які підписався відвідувач.

Виконано тестування роботи програми для різних типів користувачів, що забезпечило успішну перевірку реалізації всіх функціональних вимог до програми.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The Concerts and Live Entertainment Industry [Electronic resource] : A Significant Economic Engine. – Access mode : <https://ritzherald.com/the-concerts-and-live-entertainment-industry-a-significant-economic-engine/>.
2. Find the Best Concerts Near You, Tour Dates & Tickets Worldwide. Songkick [Electronic resource]. – Access mode : <https://www.songkick.com/>.
3. Congleton, N. The 4 Best Concert Apps of 2022 [Electronic resource] / N. Congleton. – Access mode : <https://www.lifewire.com/best-concert-apps-4586375>.
4. Setlist.fm – the setlist wiki [Electronic resource]. – Access mode : [setlist.fm](https://www.setlist.fm).
5. Bandsintown. Live Music, Concert Tickets, Tour Dates & Live Streams [Electronic resource]. – Access mode : [bandsintown.com](https://www.bandsintown.com).
6. Bates, B. Head First Java [Text] : Your Brain on Java – A Learner's Guide / B. Bates, K. Sierra. – Massachusetts : O'Reilly Media, 2003. – 656 p.
7. Sweigart, A. The Big Book of Small Python Projects [Text] : 81 Easy Practice Programs / A. Sweigart. – San Francisco : No Starch Press, 2021. – 432 p.
8. Ramalho, L. Fluent Python [Text] : Clear, Concise, and Effective Programming / L. Ramalho. – Massachusetts : O'Reilly Media, 2022. – 1012 p.
9. Vincent, W. Django for Professionals [Text] : Production websites with Python & Django (Welcome to Django) / W. Vincent. – New York : WelcomeToCode, 2020. – 288 p.
10. Mele, A. Django 3 By Example [Text] : Build powerful and reliable Python web applications from scratch, 3rd Edition / A. Mele. – Birmingham : Packt Publishing, 2020. – 568 p.
11. DotCom Free Website Template [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page278/dotcom>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення обміну враженнями від музичних концертів має розширювати сферу пошуку концертів за допомогою додавання вражень від музичних концертів, які залишають відвідувачі концертів. За допомогою цих відгуків інші користувачі можуть знаходити концерти, які їх цікавлять, відкриваючи собі нових музикантів. Окрім того, розуміючи актуальність виступів музикантів у концертах певного туру, користувачі можуть вирішити відвідати концерт цього музиканта у власному місті в межах того ж туру. Актуальний програмний інструмент в цій сфері має сприяти привертанню уваги людей до концертів, які мають їх зацікавити.

A.1 Підстави для розробки

Розробка в межах дипломної кваліфікаційної роботи реалізується на підставі наказу № 102 від 27 квітня 2022 р. за Національним університетом «Запорізька політехніка» за темою «Програмне забезпечення обміну враженнями від музичних концертів».

A.2 Призначення розробки

Програма реалізовується для забезпечення наступного призначення: пошуку музичних концертів за враженнями відвідувачів.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Функціональні характеристики, які повинна мати програма, охоплюють:

- відображення даних концерту;

- пошук музикантів за назвою, країною, інтервалом отриманих оцінок за концерти;
- створення нового концерту;
- оновлення інформації про концерт;
- створення нового туру музикантів;
- відображення переліку треків, які найчастіше виконував музикант на концертах;
- оновлення інформації про тур музикантів;
- створення нового концертного залу;
- оновлення інформації про концертний зал;
- пошук концертів за наступними фільтрами: містом проведення, інтервалом дати проведення, іменем музиканта, інтервалом оцінки (для проведених концертів);
- відображення переліку концертів у турі музиканта;
- відображення концертних залів;
- підписування на враження користувача;
- перегляд профілю користувача;
- відписування під вражень користувача;
- відображення вражень про музичні концерти користувачів, на які оформлено підписку;
- відображення переліку концертів, на яких виконувався певний музичний трек;
- відображення найкращих турів;
- відображення найкращих концертів;
- відображення майбутніх концертів музиканта;
- відображення концертів музиканта, які вже відбулись;
- відображення найкращих концертів у концертному залі;
- відображення інформації про концертний зал;
- авторизація користувачів;
- реєстрація користувачів;

- відображення інформації про музиканта;
- створення власного враження про музичний концерт;
- створення коментаря до враження про музичний концерт іншого користувача;
- відображення коментарів до враження про музичний концерт;
- створення нового музиканта;
- оновлення інформації про музиканта;
- відображення переліку музикантів, концерти яких було найвище оцінено користувачами;
- відображення власних вражень від концертів;
- оновлення власних вражень від концертів;
- відображення переліку турів музиканта;
- відображення майбутніх і минулих концертів у концертному залі;
- створення сетлиста музичного концерту;
- оновлення сетлиста музичного концерту;
- відображення музикантів за стилем музики.

A.3.2 Умови експлуатації

Експлуатація програми має забезпечуватися з одного боку звичайними користувачами, які працюють з даними, наповнюють їх, а з іншого боку – адміністраторами системи та БД.

A.3.3 Вимоги до складу та параметрів технічних засобів

Мінімальний рівень технічних засобів для забезпечення роботи програмної системи включає процесор не менш ніж 4 ядра і частотою 2,9 ГГц, оперативну пам'ять у 8 Гб, жорсткий диск у 1 Тб.

А.4 Порядок контролю та приймання

Для контролю виконання роботи використовується календарний план, а приймання роботи виконується під час проведення процедури захисту роботи перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

Б.1 Текст файла models.py

```
from django.contrib.auth.models import User
from django.db import models
from django.utils import timezone
from django.urls import reverse
```

```
COUNTRIES = (
    ("ALB", "Албанія"),
    ("AND", "Андорра"),
    ("ARE", "ОАЕ"),
    ("ARG", "Аргентина"),
    ("ARM", "Вірменія"),
    ("AUS", "Австралія"),
    ("AUT", "Австрія"),
    ("AZE", "Азербайджан"),
    ("BEL", "Бельгія"),
    ("BGR", "Болгарія"),
    ("BIH", "Боснія і Герцеговина"),
    ("BOL", "Болівія"),
    ("BRA", "Бразилія"),
    ("CAN", "Канада"),
    ("CHE", "Швейцарія"),
    ("CHL", "Чилі"),
    ("CHN", "Китай"),
    ("COL", "Колумбія"),
    ("CRI", "Коста-Рика"),
    ("CYP", "Кіпр"),
    ("CZE", "Чехія"),
    ("DEU", "Німеччина"),
    ("DNK", "Данія"),
    ("DZA", "Алжир"),
    ("ESP", "Іспанія"),
    ("EST", "Естонія"),
    ("FIN", "Фінляндія"),
    ("FJI", "Фіджі"),
    ("FRA", "Франція"),
    ("FRO", "Фарерські острови"),
    ("GBR", "Велика Британія"),
    ("GEO", "Грузія"),
    ("GRC", "Греція"),
    ("HKG", "Гонконг"),
    ("HRV", "Хорватія"),
    ("HTI", "Гаїті"),
```

("HUN", "Угорщина"),
("IDN", "Індонезія"),
("IND", "Індія"),
("IRL", "Ірландія"),
("ISL", "Ісландія"),
("ISR", "Ізраїль"),
("ITA", "Італія"),
("JAM", "Ямайка"),
("JPN", "Японія"),
("KAZ", "Казахстан"),
("KHM", "Камбоджа"),
("KOR", "Південна Корея"),
("LIE", "Ліхтенштейн"),
("LKA", "Шрі-Ланка"),
("LTU", "Литва"),
("LUX", "Люксембург"),
("LVA", "Латвія"),
("MAR", "Марокко"),
("MCO", "Монако"),
("MDA", "Молдова"),
("MDV", "Мальдіви"),
("MEX", "Мексика"),
("MKD", "Північна Македонія"),
("MLT", "Мальта"),
("MNE", "Чорногорія"),
("MYS", "Малайзія"),
("NLD", "Нідерланди"),
("NOR", "Норвегія"),
("NZL", "Нова Зеландія"),
("PAN", "Панама"),
("PER", "Перу"),
("PHL", "Філіппіни"),
("POL", "Польща"),
("PRI", "Пуерто-Рико"),
("PRT", "Португалія"),
("PRY", "Парагвай"),
("ROU", "Румунія"),
("SAU", "Саудівська Аравія"),
("SGP", "Сінгапур"),
("SMR", "Сан-Марино"),
("SRB", "Сербія"),
("SVK", "Словаччина"),
("SVN", "Словенія"),
("SWE", "Швеція"),
("SYC", "Сейшельські Острови"),

```

("THA", "Таїланд"),
("TJK", "Таджикистан"),
("TKM", "Туркменістан"),
("TUR", "Туреччина"),
("TWN", "Тайвань"),
("UKR", "Україна"),
("URY", "Уругвай"),
("USA", "США"),
("UZB", "Узбекистан")
)

class City (models.Model):
    cname = models.CharField(max_length=50)
    country = models.CharField(max_length=3, choices=
COUNTRIES)

    def __str__(self):
        return self.cname + ', ' + self.get_country_display

    def get_absolute_url(self):
        return reverse('city-view', kwargs={'pk': self.pk})

class Hall (models.Model):
    hname = models.CharField(max_length=50)
    place = models.CharField(max_length=255)
    city_id = models.ForeignKey(City, related_name='cityhalls',
on_delete=models.CASCADE)
    fulltext = models.TextField()

    def __str__(self):
        return self.hname + ' (' + self.city_id.cname + ')'

    def get_absolute_url(self):
        return reverse('hall-full', kwargs={'pk': self.pk})

class Concert (models.Model):
    tour_id = models.ForeignKey(Tour,
related_name='tourconcerts', on_delete=models.CASCADE)
    hall_id = models.ForeignKey(Hall,
related_name='hallconcerts', on_delete=models.CASCADE)
    timepoint = models.DateTimeField()
    score = models.DecimalField(max_digits = 2, decimal_places =
1, default = 0)
    fulltext = models.TextField()

```

```

songs = models.SmallIntegerField(default = 0)

def __str__(self):
    mustext = ''
    for mus in self.tour_id.musicians.all():
        if mustext:
            mustext = mustext + ', ' + mus.mname
        else:
            mustext = mus.mname
    return mustext + ' - ' self.tour_id.tname + ' (' +
self.hall_id.city_id.cname + ')'

def get_absolute_url(self):
    return reverse('concert-full', kwargs={'pk': self.pk})

def save(self, *args, **kwargs):
    self.songs =
Setlist.objects.filter(concert_id=self.concert_id).count()
    conc = Concert.objects.filter(tour_id =
self.tour_id).order_by('-timepoint')
    tourc = self.tour_id
    if tourc.enddate < self.timepoint:
        tourc.enddate = self.timepoint
        tourc.save()
    super(Setlist, self).save(*args, **kwargs)

class Tour (models.Model):
    tname = models.CharField(max_length=50)
    startdate = models.DateField(blank=True)
    enddate = models.DateField(blank=True)
    score = models.DecimalField(max_digits = 2, decimal_places =
1, default = 0)
    fulltext = models.TextField()
    musicians = models.ManyToManyField(Musician, related_name =
'mustours', db_table = 'tour_musician')

def __str__(self):
    mustext = ''
    for mus in self.musicians.all():
        if mustext:
            mustext = mustext + ', ' + mus.mname
        else:
            mustext = mus.mname
    return mustext + ' - ' self.tname

```

```

    def get_absolute_url(self):
        return reverse('tour-full', kwargs={'pk': self.pk})

class Musician (models.Model):
    mname = models.CharField(max_length=50)
    country = models.CharField(max_length=3, choices=
COUNTRIES)
    score = won_matches_percentage =
models.DecimalField(max_digits = 2, decimal_places = 1, default = 0)
    styles = models.ManyToManyField(Musician, related_name =
'musicians', db_table = 'musician_style')

    def __str__(self):
        return self.mname

    def get_absolute_url(self):
        return reverse('musician-view', kwargs={'pk':
self.pk})

class Style (models.Model):
    sname = models.CharField(max_length=20)

    def __str__(self):
        return self.sname

    def get_absolute_url(self):
        return reverse('style-view', kwargs={'pk': self.pk})

class Impression (models.Model):
    concert_id = models.ForeignKey(Concert,
related_name='concertimpressions', on_delete=models.CASCADE)
    profile_id = models.ForeignKey(Profile,
related_name='profileimpressions', on_delete=models.CASCADE)
    timepoint = models.DateTimeField()
    score = models.SmallIntegerField(default = 0)
    fulltext = models.TextField(blank = True)
    energyscore = models.SmallIntegerField(default = 0)
    energytext = models.TextField(blank = True)
    soundscore = models.SmallIntegerField(default = 0)
    soundtext = models.TextField(blank = True)
    dancescore = models.SmallIntegerField(default = 0)
    dancetext = models.TextField(blank = True)
    showscore = models.SmallIntegerField(default = 0)
    showtext = models.TextField(blank = True)
    managementscore = models.SmallIntegerField(default = 0)

```

```

managementtext = models.TextField(blank = True)

def __str__(self):
    return str(self.concert_id)

def get_absolute_url(self):
    return reverse('impression-view', kwargs={'pk':
self.pk})

def save(self, *args, **kwargs):
    self.timepoint = datetime.now
    super(Impression, self).save(*args, **kwargs)

class Setlist (models.Model):
    concert_id = models.OneToOneField(Concert,
related_name='concsetlist', on_delete=models.CASCADE)
    track_id = models.ForeignKey(Track,
related_name='tracksetlists', on_delete=models.CASCADE)
    comusician = models.ForeignKey(Musician,
related_name='musiciansetlists', on_delete=models.CASCADE)
    order = models.SmallIntegerField(default = 0)

def __str__(self):
    mustext = ''
    for mus in self.concert_id.tour_id.musicians.all():
        if mustext:
            mustext = mustext + ', ' + mus.mname
        else:
            mustext = mus.mname
    return mustext + self.concert_id.hall_id.hname

def get_absolute_url(self):
    return reverse('setlist-view', kwargs={'pk': self.pk})

def save(self, *args, **kwargs):
    conc = Concert.objects.get(id=concert_id.id)
    conc.songs =
Setlist.objects.filter(concert_id=self.concert_id).count()
    super(Setlist, self).save(*args, **kwargs)

class Track (models.Model):
    trname = models.CharField(max_length=255)
    musician = models.ForeignKey(Musician,
related_name='mustracks', on_delete=models.CASCADE)

```

```

def __str__(self):
    return self.musician.mname + ' - ' + self.trname

def get_absolute_url(self):
    return reverse('mus-full', kwargs={'pk': self.pk})

class Comment (models.Model):
    impression_id = models.ForeignKey(Impression,
related_name='imprcomments', on_delete=models.CASCADE)
    profile_id = models.ForeignKey(Profile,
related_name='profilecomments', on_delete=models.CASCADE)
    timepoint = models.DateTimeField()
    fulltext = models.TextField()

def __str__(self):
    return self.fulltext

def save(self, *args, **kwargs):
    self.timepoint = datetime.now
    super(Comment, self).save(*args, **kwargs)

class Subscribe (models.Model):
    profile_source = models.ForeignKey(Profile,
related_name='profsubscribers', on_delete=models.CASCADE)
    profile_id = models.ForeignKey(Profile,
related_name='profsubscribed', on_delete=models.CASCADE)

def save(self, *args, **kwargs):
    subscribes = Subscribe.objects.filter(profile_source =
self.profile_source, profile_id = self.profile_id)
    if not subscribes.count():
        profile1 = self.profile_source
        profile1.subscribers = profile1.subscribers + 1
        profile1.save()
        profile2 = self.profile_id
        profile2.subscribed = profile2.subscribed + 1
        profile1.save()
    else:
        profile1 = self.profile_source
        profile1.subscribers = profile1.subscribers - 1
        profile1.save()
        profile2 = self.profile_id
        profile2.subscribed = profile2.subscribed - 1
        profile1.save()
        subscribes.delete()

```

```

class Profile (models.Model):
    user_id = models.OneToOneField(User,
on_delete=models.CASCADE)
    subscribers = models.BigIntegerField(default = 0)
    subscribed = models.BigIntegerField(default = 0)

    def get_absolute_url(self):
        return reverse('prof-full', kwargs={'pk': self.pk})

```

Б.2 Текст файла concertview.py

```

from django.contrib.auth.mixins import LoginRequiredMixin
from django.urls import reverse_lazy
from django.views.generic import ListView
from django.views.generic import UpdateView
from django.views.generic.edit import CreateView
from django.views.generic.detail import DetailView
from .models import Concert
from .models import Musician
from .models import Tour
from .models import Hall
from .models import City
from .models import Setlist
from .models import Track

class ConcertSearchView(ListView):
    model = Concert
    template_name = 'concertsearch.html'
    paginate_by = 10

    def get_queryset(self):
        city = self.request.GET.get('city')
        concerts = Concert.objects.all()
        if city:
            concerts =
concerts.filter(hall_id__city_id__cname__contains=city)
            startd = self.request.GET.get('start')
            if startd:
                concerts = concerts.filter(timepoint__gte =
startd)
            endd = self.request.GET.get('end')
            if endd:
                concerts = concerts.filter(timepoint__lte = endd)

```

```

        musscore1 = self.request.GET.get('score1')
        if musscore1:
            concerts = concerts.filter(score__gte =
musscore1)

        musscore2 = self.request.GET.get('score2')
        if musscore2:
            concerts = concerts.filter(score__lte =
musscore2)

        musname = self.request.GET.get('musician')
        if musname:
            mus = Musician.objects.filter(mname__contains =
musname)

            con = []
            for m in mus:
                conc =
concerts.filter(tour_id__musicians__id__exact = m.id)
                for c in conc:
                    if c not in con:
                        con.add(c)

            concerts = con
        concerts = concerts.order_by('-timepoint')
        return concerts

```

```

    def get_context_data(self, **kwargs):
        context = super(ConcertSearchView,
self).get_context_data(**kwargs)
        city = self.request.GET.get('city')
        context['city'] = city
        startd = self.request.GET.get('start')
        context['startd'] = startd
        endd = self.request.GET.get('end')
        context['endd'] = endd
        musscore1 = self.request.GET.get('score1')
        context['musscore1'] = musscore1
        musscore2 = self.request.GET.get('score2')
        context['musscore2'] = musscore2
        musname = self.request.GET.get('musician')
        context['musname'] = musname
        return context

```

```

class ConcertView (DetailView):
    template_name = 'concert.html'
    model = Concert

```

```

    def get_context_data(self, **kwargs):

```

```

        context = super().get_context_data(**kwargs)
        t = get_object_or_404(Concert, id = self.object.pk)
        context['concerts'] = Concert.objects.filter(tour_id =
t.tour_id).order_by('timepoint')
        return context

```

```

class NewConcertView(LoginRequiredMixin, CreateView):
    model = Concert
    fields = ['timepoint', 'fulltext', 'hall_id', 'tour_id']
    template_name = 'newconcert.html'

    def get_success_url(self):
        return reverse_lazy('concert-new', kwargs={'pk':
self.object.pk})

    def get_context_data(self, **kwargs):
        self.tour_id = get_object_or_404(Tour,
id=self.kwargs['tour_id'])
        self.hall_id = get_object_or_404(Hall,
id=self.kwargs['hall_id'])
        kwargs['tour_id'] = self.tour_id
        kwargs['hall_id'] = self.hall_id
        return super().get_context_data(**kwargs)

    def form_valid(self, form):
        self.tour_id = get_object_or_404(Tour,
id=self.kwargs['tour_id'])
        self.hall_id = get_object_or_404(Hall,
id=self.kwargs['hall_id'])
        form.instance.tour_id = self.tour_id
        form.instance.hall_id = self.hall_id
        return super().form_valid(form)

```

```

class UpdateConcertView(LoginRequiredMixin, UpdateView):
    model = Concert
    fields = ['timepoint', 'fulltext', 'hall_id', 'tour_id']
    template_name = 'updateconcert.html'

    def get_success_url(self):
        return reverse_lazy('concert-update', kwargs={'pk':
self.object.pk})

    def get_context_data(self, **kwargs):
        self.tour_id = get_object_or_404(Tour,
id=self.kwargs['tour_id'])

```

```

        self.hall_id = get_object_or_404(Hall,
id=self.kwargs['hall_id'])
        kwargs['tour_id'] = self.tour_id
        kwargs['hall_id'] = self.hall_id
        return super().get_context_data(**kwargs)

    def form_valid(self, form):
        self.tour_id = get_object_or_404(Tour,
id=self.kwargs['tour_id'])
        self.hall_id = get_object_or_404(Hall,
id=self.kwargs['hall_id'])
        form.instance.tour_id = self.tour_id
        form.instance.hall_id = self.hall_id
        return super().form_valid(form)

class MusicianConcertsView(ListView):
    model = Concert
    template_name = 'musconcerts.html'
    paginate_by = 10

    def get_queryset(self):
        mus = self.request.GET.get('musician_id')
        curr = datetime.now
        con = []
        conc =
Concert.objects.filter(tour_id__musicians__id__exact = mus,
timepoint__gte = curr)
        for c in conc:
            if c not in con:
                con.add(c)
        concerts = con.order_by('timepoint')
        return concerts

    def get_context_data(self, **kwargs):
        context = super(MusicianConcertsView,
self).get_context_data(**kwargs)
        mus = self.request.GET.get('musician_id')
        context['musician_id'] = mus
        return context

class ArchiveConcertsView(ListView):
    model = Concert
    template_name = 'musconcerts.html'
    paginate_by = 10

```

```

    def get_queryset(self):
        mus = self.request.GET.get('musician_id')
        curr = datetime.now
        con = []
        conc =
Concert.objects.filter(tour_id__musicians__id__exact = mus,
timepoint__lte = curr)
        for c in conc:
            if c not in con:
                con.add(c)
        concerts = con.order_by('-timepoint')
        return concerts

    def get_context_data(self, **kwargs):
        context = super(ArchiveConcertsView,
self).get_context_data(**kwargs)
        mus = self.request.GET.get('musician_id')
        context['musician_id'] = mus
        return context

class BestConcertsView (ListView):
    template_name = 'bestconcerts.html'
    model = Concert

    def get_queryset(self):
        return Concert.objects.all().order_by('-score')[:100]

class TrackConcertsView(ListView):
    model = Concert
    template_name = 'trackconcerts.html'
    paginate_by = 10

    def get_queryset(self):
        trackid = self.request.GET.get('track')
        concerts =
Concert.objects.filter(concsetlist__track_id__id = trackid)
        return concerts.order_by('-timepoint')

    def get_context_data(self, **kwargs):
        context = super(TrackConcertsView,
self).get_context_data(**kwargs)
        trackid = self.request.GET.get('track')
        context['track'] = trackid
        return context

```

```

class TourView (DetailView):
    template_name = 'tour.html'
    model = Tour

class BestToursView (ListView):
    template_name = 'besttours.html'
    model = Tour

    def get_queryset(self):
        return Tour.objects.all().order_by('-score')[:100]

class NewTourView(LoginRequiredMixin, CreateView):
    model = Tour
    fields = ['tname', 'startdate', 'enddate', 'fulltext']
    template_name = 'newtour.html'

    def get_success_url(self):
        return reverse_lazy('tour-new', kwargs={'pk':
self.object.pk})

class UpdateTourView(LoginRequiredMixin, UpdateView):
    model = Tour
    fields = ['tname', 'startdate', 'enddate', 'fulltext']
    template_name = 'updatetour.html'

    def get_success_url(self):
        return reverse_lazy('tour-update', kwargs={'pk':
self.object.pk})

class NewHallView(LoginRequiredMixin, CreateView):
    model = Hall
    fields = ['hname', 'place', 'city_id', 'fulltext']
    template_name = 'newhall.html'

    def get_success_url(self):
        return reverse_lazy('hall-new', kwargs={'pk':
self.object.pk})

    def get_context_data(self, **kwargs):
        self.city_id = get_object_or_404(City,
id=self.kwargs['city_id'])
        kwargs['city_id'] = self.city_id
        return super().get_context_data(**kwargs)

    def form_valid(self, form):

```

```

        self.city_id = get_object_or_404(City,
id=self.kwargs['city_id'])
        kwargs['city_id'] = self.city_id
        return super().form_valid(form)

class HallBestConcertsView (ListView):
    template_name = 'hallbestconcerts.html'
    model = Concert

    def get_queryset(self):
        hallid = self.request.GET.get('hall')
        return Concert.objects.filter(hall_id =
hallid).order_by('-score')[:10]

    def get_context_data(self, **kwargs):
        context = super(HallBestConcertsView,
self).get_context_data(**kwargs)
        hallid = self.request.GET.get('hall')
        context['hall'] = hallid
        return context

class UpdateHallView(LoginRequiredMixin, UpdateView):
    model = Hall
    fields = ['hname', 'place', 'city_id', 'fulltext']
    template_name = 'updatehall.html'

    def get_success_url(self):
        return reverse_lazy('hall-update', kwargs={'pk':
self.object.pk})

    def get_context_data(self, **kwargs):
        self.city_id = get_object_or_404(City,
id=self.kwargs['city_id'])
        kwargs['city_id'] = self.city_id
        return super().get_context_data(**kwargs)

    def form_valid(self, form):
        self.city_id = get_object_or_404(City,
id=self.kwargs['city_id'])
        kwargs['city_id'] = self.city_id
        return super().form_valid(form)

class HallsView(ListView):
    model = Hall
    template_name = 'halls.html'

```

```

paginate_by = 10

def get_queryset(self):
    self.city_id = get_object_or_404(City,
id=self.kwargs['city_id'])
    kwargs['city_id'] = self.city_id
    return concerts.order_by('-timepoint')

def get_context_data(self, **kwargs):
    context = super(TrackConcertsView,
self).get_context_data(**kwargs)
    self.city_id = get_object_or_404(City,
id=self.kwargs['city_id'])
    kwargs['city_id'] = self.city_id
    return context

class NewSetlistView(LoginRequiredMixin, CreateView):
    model = Setlist
    fields = ['concert_id', 'track_id', 'comusician', 'order']
    template_name = 'newsetlist.html'

def get_success_url(self):
    return reverse_lazy('setlist-new', kwargs={'pk':
self.object.pk})

def get_context_data(self, **kwargs):
    self.concert_id = get_object_or_404(Concert,
id=self.kwargs['concert_id'])
    kwargs['concert_id'] = self.concert_id
    self.track_id = get_object_or_404(Track,
id=self.kwargs['track_id'])
    kwargs['track_id'] = self.track_id
    self.comusician = get_object_or_404(Musician,
id=self.kwargs['comusician'])
    kwargs['comusician'] = self.comusician
    return super().get_context_data(**kwargs)

def form_valid(self, form):
    self.concert_id = get_object_or_404(Concert, id=
self.concert_id = get_object_or_404(Concert,
id=self.kwargs['concert_id'])
    kwargs['concert_id'] = self.concert_id
    self.track_id = get_object_or_404(Track,
id=self.kwargs['track_id'])
    kwargs['track_id'] = self.track_id

```

```

        self.comusician = get_object_or_404(Musician,
id=self.kwargs['comusician'])
        kwargs['comusician'] = self.comusician
        return super().form_valid(form)

class UpdateSetlistView(LoginRequiredMixin, UpdateView):
    model = Setlist
    fields = ['concert_id', 'track_id', 'comusician', 'order']
    template_name = 'updatesetlist.html'

    def get_success_url(self):
        return reverse_lazy('setlist-update', kwargs={'pk':
self.object.pk})

    def get_context_data(self, **kwargs):
        self.concert_id = get_object_or_404(Concert,
id=self.kwargs['concert_id'])
        kwargs['concert_id'] = self.concert_id
        self.track_id = get_object_or_404(Track,
id=self.kwargs['track_id'])
        kwargs['track_id'] = self.track_id
        self.comusician = get_object_or_404(Musician,
id=self.kwargs['comusician'])
        kwargs['comusician'] = self.comusician
        return super().get_context_data(**kwargs)

    def form_valid(self, form):
        self.concert_id = get_object_or_404(Concert,
id=self.kwargs['concert_id'])
        kwargs['concert_id'] = self.concert_id
        self.track_id = get_object_or_404(Track,
id=self.kwargs['track_id'])
        kwargs['track_id'] = self.track_id
        self.comusician = get_object_or_404(Musician,
id=self.kwargs['comusician'])
        kwargs['comusician'] = self.comusician
        return super().form_valid(form)

class MostMusicianTracksView(ListView):
    model = Track
    template_name = 'mostmusiciantracks.html'

    def get_queryset(self):
        musicianid = self.request.GET.get('musician')

```

```

        tracks = Track.objects.filter(musician__id =
musicianid)
        trackcounts = []
        for tr in tracks:
            num = Setlist.objects.filter(track_id =
tr).count()

            trackresult = {}
            trackresult['name'] = tr
            trackresult['num'] = num
            trackcounts.append(trackresult)
        trackcounts.sort(key = lambda x: x['num'])
        return trackcounts

    def get_context_data(self, **kwargs):
        context = super(MostMusicianTracksView,
self).get_context_data(**kwargs)
        musicianid = self.request.GET.get('musician')
        context['musician'] = musicianid
        return context

class BestMusicianConcertsView (ListView):
    template_name = 'bestmusicianscon.html'
    model = Concert

    def get_queryset(self):
        concreat = Concert.objects.all().order_by('-score')
        musicians = []
        num = 0
        for conc in concreat:
            mus = conc.tour_id.musicians
            for m in mus:
                if m not in musicians:
                    musicians.add(m)
                    num = num + 1
            if num == 100:
                break
        return musicians[:100]

class HallView (DetailView):
    template_name = 'hall.html'
    model = Hall

class AlphabetHallsView(ListView):
    model = Hall
    template_name = 'ahalls.html'

```

```

        ordering = ['hname']

class HallArchiveConcertsView(ListView):
    model = Concert
    template_name = 'archhallconcerts.html'
    paginate_by = 10

    def get_queryset(self):
        h = self.request.GET.get('hall_id')
        curr = datetime.now
        concerts = Concert.objects.filter(hall_id = h,
timepoint__lte = curr)
        return concerts.order_by('-timepoint')

    def get_context_data(self, **kwargs):
        context = super(ArchiveConcertsView,
self).get_context_data(**kwargs)
        h = self.request.GET.get('hall_id')
        context['hall_id'] = h
        context['halls'] = Hall.objects.filter(city_id =
h.city_id)
        return context

class HallConcertsView(ListView):
    model = Concert
    template_name = 'hallconcerts.html'
    paginate_by = 10

    def get_queryset(self):
        h = self.request.GET.get('hall_id')
        curr = datetime.now
        concerts = Concert.objects.filter(hall_id = h,
timepoint__gte = curr)
        return concerts.order_by('timepoint')

    def get_context_data(self, **kwargs):
        context = super(HallConcertsView,
self).get_context_data(**kwargs)
        h = self.request.GET.get('hall_id')
        context['hall_id'] = h
        context['halls'] = Hall.objects.filter(city_id =
h.city_id)
        return context

```

Б.3 Текст файла musicianview.py

```

from django.contrib.auth.mixins import LoginRequiredMixin
from django.urls import reverse_lazy
from django.views.generic import ListView
from django.views.generic import UpdateView
from django.views.generic.edit import CreateView
from django.views.generic.detail import DetailView
from .models import Musician
from .models import Style
from .models import Concert

class MusicianSearchView(ListView):
    model = Musician
    template_name = 'musiciansearch.html'
    paginate_by = 10

    def get_queryset(self):
        musname = self.request.GET.get('name')
        musicians = Musician.objects.all()
        if musname:
            musicians = musicians.filter(mname__contains =
musname)

        musscore1 = self.request.GET.get('score1')
        if musscore1:
            musicians = musicians.filter(score__gte =
musscore1)

        musscore2 = self.request.GET.get('score2')
        if musscore2:
            musicians = musicians.filter(score__lte =
musscore2)

        countryname = self.request.GET.get('country')
        if countryname:
            shortcountry = list(filter(lambda x: x[1]==
countryname, COUNTRIES))[0][1]
            musicians = musicians.filter(country =
shortcountry)

        musicians = musicians.order_by('mname')
        return musicians

    def get_context_data(self, **kwargs):
        context = super(MusicianSearchView,
self).get_context_data(**kwargs)

```

```

        musname = self.request.GET.get('name')
        context['name'] = musname
        musscore1 = self.request.GET.get('score1')
        context['score1'] = musscore1
        musscore2 = self.request.GET.get('score2')
        context['score2'] = musscore2
        countryname = self.request.GET.get('country')
        context['country'] = countryname
        return context

class MusicianView (DetailView):
    template_name = 'musician.html'
    model = Musician

class NewMusicianView(LoginRequiredMixin, CreateView):
    model = Musician
    fields = ['mname', 'country']
    template_name = 'newmusician.html'

    def get_success_url(self):
        return reverse_lazy('musician-new', kwargs={'pk':
self.object.pk})

    def get_context_data(self, **kwargs):
        self.country = list(filter(lambda x: x[1] ==
self.kwargs['country'], COUNTRIES))[0][0]
        kwargs['country'] = self.country
        return super().get_context_data(**kwargs)

    def form_valid(self, form):
        self.country = list(filter(lambda x: x[1] ==
self.kwargs['country'], COUNTRIES))[0][0]
        kwargs['country'] = self.country
        return super().form_valid(form)

class UpdateMusicianView(LoginRequiredMixin, UpdateView):
    model = Musician
    fields = ['mname', 'country']
    template_name = 'updatemusician.html'

    def get_success_url(self):
        return reverse_lazy('musician-update', kwargs={'pk':
self.object.pk})

```

```

    def get_context_data(self, **kwargs):
        self.country = list(filter(lambda x: x[1] ==
self.kwarg['country'], COUNTRIES))[0][0]
        kwarg['country'] = self.country
        return super().get_context_data(**kwargs)

    def form_valid(self, form):
        self.country = list(filter(lambda x: x[1] ==
self.kwarg['country'], COUNTRIES))[0][0]
        kwarg['country'] = self.country
        return super().form_valid(form)

class BestMusiciansView (ListView):
    template_name = 'bestmusicians.html'
    model = Musician

    def get_queryset(self):
        return Musician.objects.all().order_by('-score')[:100]

class StyleMusiciansView(ListView):
    model = Musician
    template_name = 'stylemusicians.html'
    paginate_by = 10

    def get_queryset(self):
        styleid = self.request.GET.get('style')
        musicians = Musician.objects.filter(styles__id__exact
= styleid)
        return musicians.order_by('mname')

    def get_context_data(self, **kwargs):
        context = super(StyleMusiciansView,
self).get_context_data(**kwargs)
        styleid = self.request.GET.get('style')
        context['style'] = styleid
        return context

```

ДОДАТОК В
Слайди презентації



Рисунок В.1 – Слайд № 1

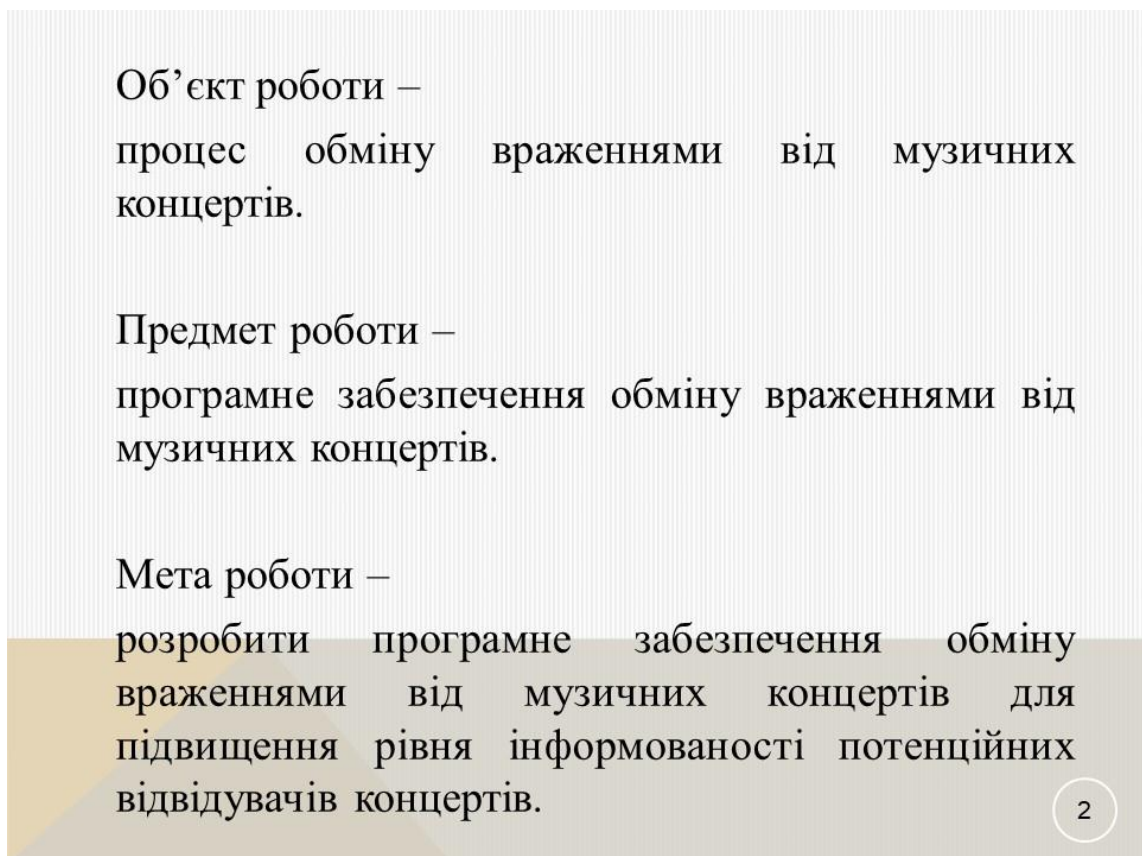


Рисунок В.2 – Слайд № 2

Завдання роботи

- ✓ дослідження предметної області;
- ✓ визначення вимог до програмного забезпечення обміну враженнями від музичних концертів;
- ✓ проєктування програмного забезпечення;
- ✓ реалізація програмного забезпечення.

3

Рисунок В.3 – Слайд № 3

Порівняння програмних аналогів

Критерій	Songkick	Setlist.fm	Bandintown	Musimpress
Пошук артистів	+	+	+	+
Пошук концертів	+	+	+	+
Перегляд сетлиста концерту	-	+	-	+
Створення власних вражень	За музикантом	-	За концертом	За концертом
Підписка на користувачів	-	-	-	+
Оцінка концерту за критеріями	-	-	-	+

4

Рисунок В.4 – Слайд № 4

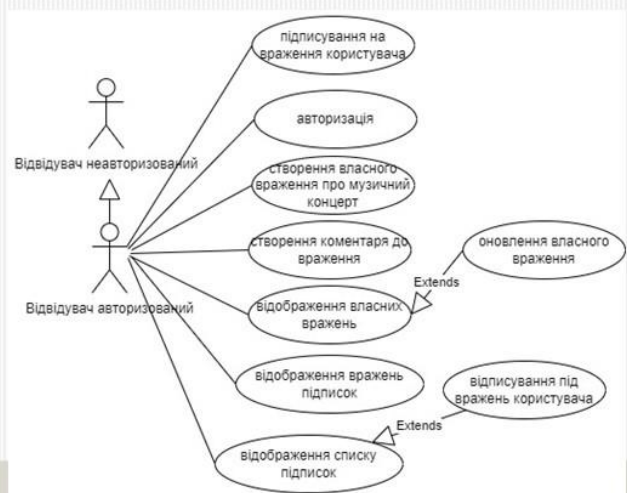
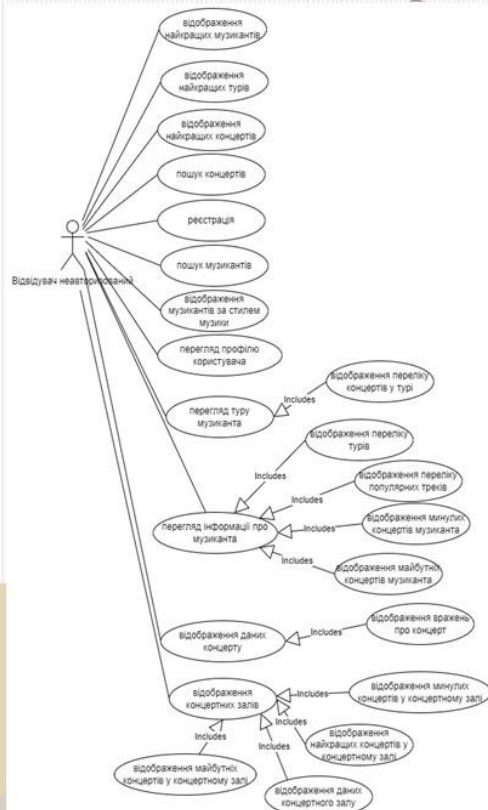
Вибір мови програмування для розробки програми

Критерій	Java	Python
Кросплатформність	+	+
Веброзробка за допомогою готових фреймворків	+	+
Динамічна типізація	-	+
Швидке написання коду	-	+
Швидке виконання коду	+	+-
Швидкий запуск	+-	+

5

Рисунок В.5 – Слайд № 5

Діаграми прецедентів



6

Рисунок В.6 – Слайд № 6

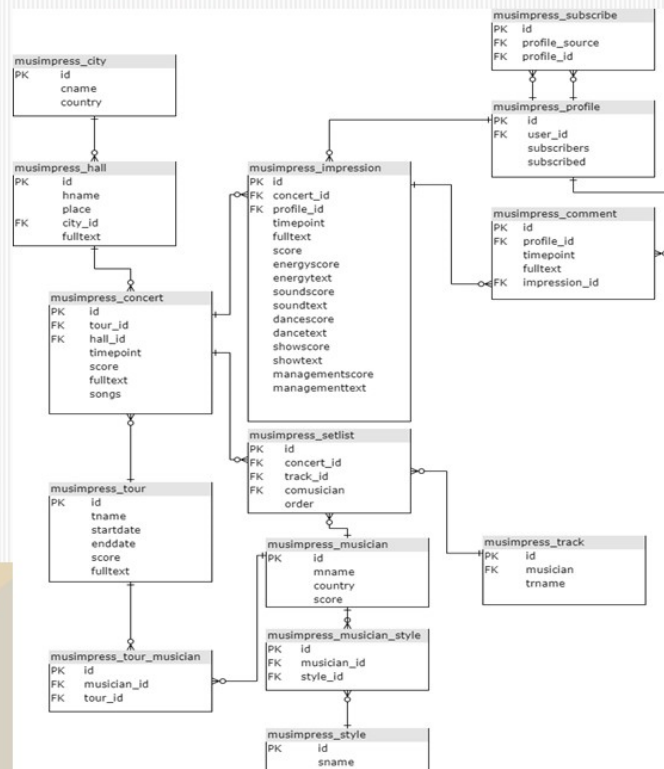
Діаграми прецедентів



7

Рисунок В.7 – Слайд № 7

Структурна схема бази даних



8

Рисунок В.8 – Слайд № 8



Рисунок В.9 – Слайд № 9

Інтерфейси програми

MusImpress Музиканти Тури Концерти Профіль

Створення нового концерту

Музиканти / Моніторінг / Створити концерт

Створити новий концерт

Тур музиканта

Без туру

Місто

Київ

Концертна зала

Stereo Plaza

10/10/2022 19:00

Опис

Завантажити зображення

Створити концерт

Рисунок В.10 – Слайд № 10



Рисунок В.11 – Слайд № 11

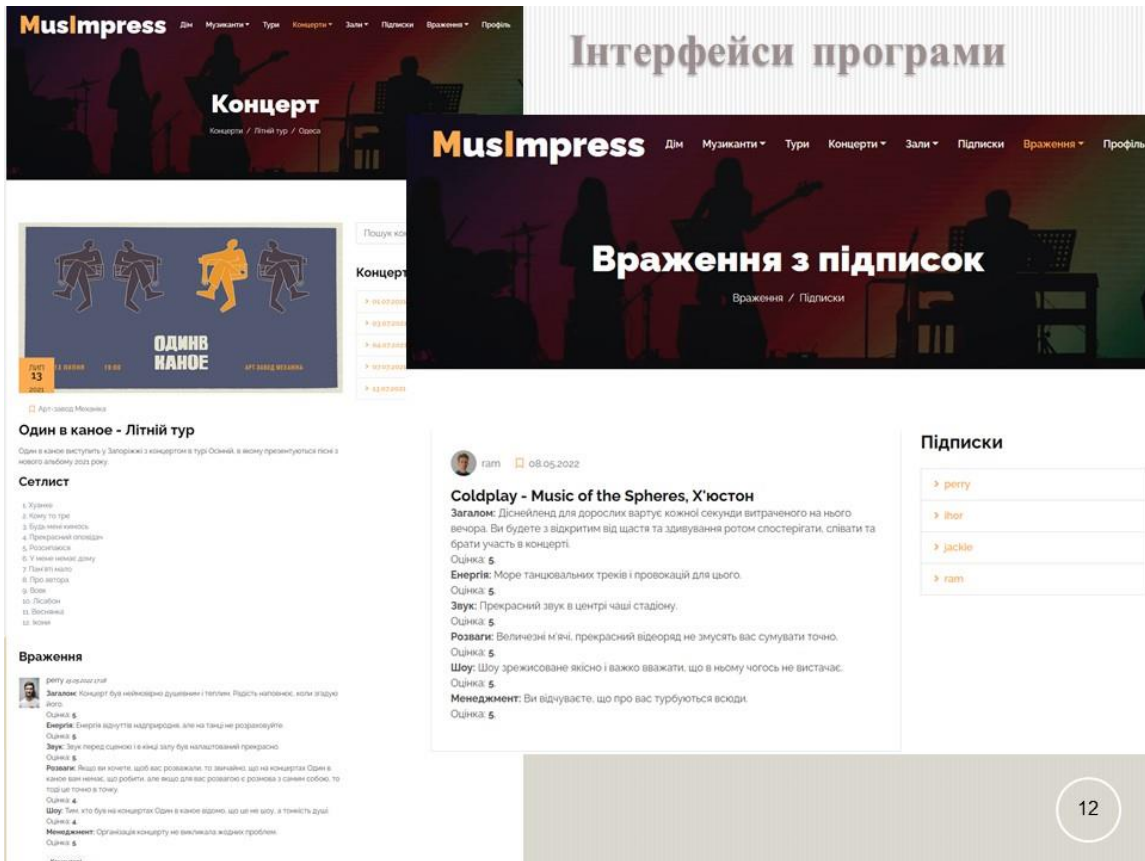


Рисунок В.12 – Слайд № 12

Висновки

Визначено основні принципи створення застосунку, що стосуються етапу його проектування: обрано мову Python і відповідний стек засобів веброзробки, включаючи фреймворк Django, для реалізації програми, визначено сценарії роботи користувачів з програмою, включаючи неавторизованого відвідувача, авторизованого відвідувача, менеджера музиканта, менеджера вебсистеми, виконано проектування схеми роботи з даними програми через створення відповідної бази даних.

Реалізовано програму за допомогою використання фреймворку Django.

Виконано тестування роботи програми для різних типів користувачів, що забезпечило успішну перевірку реалізації всіх функціональних вимог до програми.