

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОШУКУ

(назва теми)

УЧАСНИКІВ МУЗИЧНОГО КОЛЕКТИВУ

SOFTWARE TO SEARCH FOR MUSICAL BAND MEMBERS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-220сп

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ЛУЦИШИН М.В.

(ПРИЗВИЩЕ та ініціали)

Керівник ЛЬОВКІН В.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ТІМЕНКО А.В.

(ПРИЗВИЩЕ та ініціали)

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
« » 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ЛУЦИШИНА Максима Віталійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення пошуку учасників музичного колективу. Software to Search for Musical Band Members

керівник проєкту (роботи) к.т.н., доцент ЛЬОВКІН Валерій Миколайович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «04» квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 1 червня 2023 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. Проєктування програми. 3. Розробка програми. 4. Експлуатація та тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ЛЬОВКІН В.М., доцент		
Нормоконтроль	БЄЛОВА А.В., асистент		

7. Дата видачі завдання « 4 » квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Срок 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)

_____ Максим ЛУЦИШИН
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Валерій ЛЬОВКІН
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 93 с., 27 рис., 3 табл., 3 дод., 14 джерел.

МУЗИЧНИЙ КОЛЕКТИВ, МУЗИЧНИЙ ПРОЄКТ, МУЗИКАНТ, НОТАТКА, ВЕБФРЕЙМВОРК.

Об'єкт роботи – процес пошуку учасників музичного колективу. Предмет роботи – програмне забезпечення пошуку учасників музичного колективу. Мета роботи – розробити програмне забезпечення пошуку учасників музичного колективу для спрощення взаємодії між музикантами і музичними колективами.

Матеріали, методи та технічні засоби: засоби мови програмування Python разом з вебфреймворком Django та системою керування MySQL для роботи з базою даних.

Результати. Програмне забезпечення, описані рішення щодо його проєктування і реалізації є основними результатами роботи. Вебзастосунок надає можливості пошуку не тільки музикантів для участі в колективі на певних позиціях і навпаки, але і пошуку музикантів саме під окремі музичні проєкти для реалізації тимчасових ідей колективів, надаючи для цього засоби створення нотаток з оголошень, публікації оголошень, подачі заявок, керування заявками і оголошеннями.

Висновки. Програма може експлуатуватися для пошуку музикантами колективів, а колективами музикантів з наданням набору інструментів, які роблять цей процес зручним і результативним за наявності відповідних вакансій.

Галузь використання. Музична індустрія: пошук музикантів для участі на постійній основі у складі музичного колективу або на певний період у музичному проєкті, пошук музикантами музичних колективів для взаємодії.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 93 pages, 27 figures, 3 tables, 3 appendixes, 14 sources.

MUSIC BAND, MUSIC PROJECT, MUSICIAN, NOTE, WEBFRAMEWORK.

The object of the work is the process of finding members of a musical band. The subject of the work is software for searching for members of a musical band. The purpose of the work is to develop software for searching for members of a musical band to simplify the interaction between musicians and musical groups.

Materials, methods and techniques: Python programming language tools along with the Django web framework and the MySQL database management system.

The results. The software, the described solutions for its design and implementation are the main results of the work. The web application provides opportunities not only to search for musicians to participate in the band in certain positions and vice versa, but also to find musicians specifically for individual musical projects for the implementation of temporary ideas of bands, providing the means for creating notes from advertisements, publishing advertisements, submitting applications, managing applications and advertisements.

Conclusions. The program can be used by musicians to search for bands, and by bands for searching for musicians with the provision of a set of tools making this process convenient and effective if suitable vacancies are available.

Field of use. Music industry: search for musicians to participate on a permanent basis as part of a music band or for a certain period in a music project, search by musicians for music bands for interaction.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	10
1.1 Огляд програмного забезпечення пошуку учасників музичного колективу	10
1.2 Порівняння аналогів	15
1.3 Висновки за розділом 1	16
2 Проєктування програми.....	18
2.1 Вибір засобів проєктування і розробки	18
2.2 Визначення функціональних вимог до програмного забезпечення	19
2.3 Проєктування структури бази даних.....	23
2.4 Висновки за розділом 2	37
3 Розробка програми	38
3.1 Порядок функціонування програми.....	38
3.2 Рішення з реалізації одиниць додатку	39
3.3 Висновки за розділом 3	48
4 Експлуатація та тестування програми	49
4.1 Опис програми.....	49
4.2 Експлуатація програми.....	49
4.3 Тестування програми.....	54
4.4 Висновки за розділом 4	57
Висновки	58
Перелік джерел посилання	59
Додаток А Технічне завдання	61
Додаток Б Текст програми	66
Додаток В Слайди презентації.....	86

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

ПЗ – пояснювальна записка.

ВСТУП

Пошук однодумців завжди є певною проблемою, яку будь-яка особа, що має власні ідеї для реалізації, намагається певним чином вирішити. Коли ці ідеї стосуються музики, дуже часто потрібно знайти для них деякий колектив людей. Колектив може створюватися, змінювати учасників з часом, розширювати або зменшувати склад учасників загалом. Усе це потребує пошуку музикантів або музичних колективів. Навіть якщо музичний колектив уже сформований, з часом його склад, як і склад будь-якого іншого колективу, змінюється. Також з'являються тимчасові учасники.

Пошук музикантів для участі в музичному колективі може здійснюватися в наступних місцях:

- соціальні медіа: групи у Facebook та рекламні оголошення у Instagram є одним із найпоширеніших способів, за допомогою яких музиканти знаходять один одного, однак групи фрагментовані, в Instagram швидше за все вдасться долучити до колективу друга або друга друга, здійснити широкий пошук з наявними власними вимогами зокрема є все ж на практиці достатньо складним;

- контакти з іншими гуртами та людьми у музичній індустрії: будь-хто, хто вже є в музичному колективі або займається музикою, знає незліченну кількість інших музикантів;

- місцеві студії: вони наповнені як колективами, так і індивідуальними музикантами, які беруть участь в записах різних колективів, їхніх проєктів;

- друзі та родина: вони можуть бути безпосередніми учасниками музичних колективів як це не одноразово ставалося у всесвітньо відомих колективах, мати цінні для пошуку музичні контакти;

- школа, університет або робота: у цих закладах часто є середовище для зустрічі з однодумцями, щоб приєднатися до музичного колективу [1]-[2].

Проте не завжди знайомі, персональні контакти цю проблему здатні вирішити. Саме для таких ситуацій, а також для ситуацій, коли потрібно розширити коло пошуку, має використовуватися програмне забезпечення, що повинно бути створено в роботі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд програмного забезпечення пошуку учасників музичного колективу

Hendrix [3] (рис. 1.1) є першим вебзастосунком, який допомагає музикантам шукати музичні колективи і який розглянуто в цьому огляді. Це вебзастосунок, який працює тільки для США. Він базується на дещо особливій концепції, яка об'єднує і музикантів, і музичні колективи в один потік. Це фактично профілі, за якими можна виконувати пошук. Пошук включає сортування за останнім оновленням, визначення назви, визначення позиції, відстань, на якій розташований учасник профілю, жанр, інструменти, наявність фотографій у профілі та наявність музики.

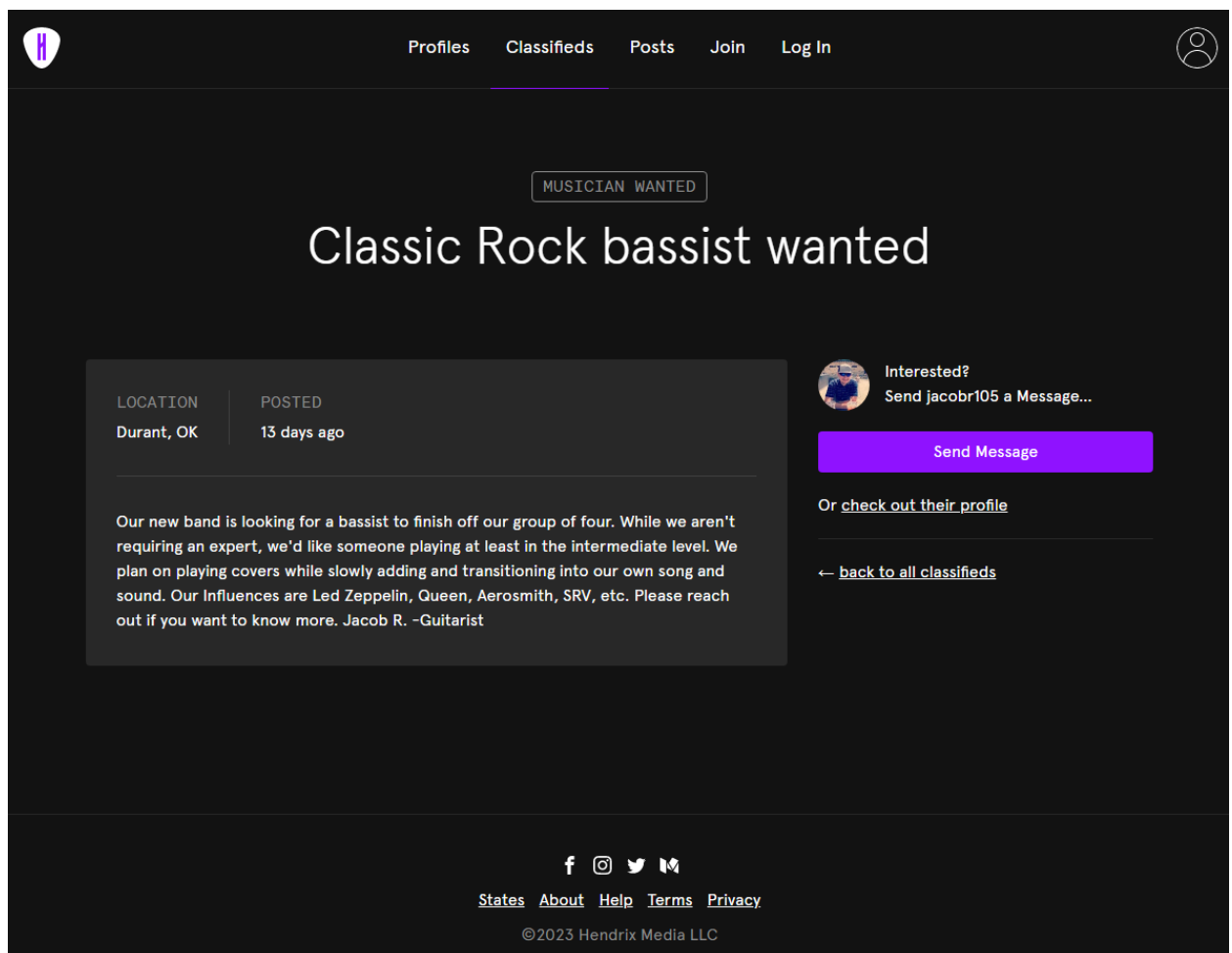


Рисунок 1.1 – Вебсайт Hendrix

Самі оголошення розташовуються окремо в окремому розділі. Вони впорядковані тільки за датою публікації. Після відкриття оголошення надається загальна інформація, приводиться час, який минув з дати публікації, дозволяється відправити повідомлення користувачу, а також перейти до сторінки профілю. На сторінці профілю приводиться інформація про жанри, інструменти колективу, музика колективу, а також можна підписатися на колектив або переглянути на кого користувач підписався, або хто підписався на нього. Кожен музикант, який бажає взаємодіяти, робить відповідну позначку, що відображається в його профілі.

Вебсайт Bandmix [4] (рис. 1.2) дозволяє шукати музичні колективи, музикантів та представників музичної індустрії (зали, вчителі, продюсери тощо). Вебсайт працює тільки у Великобританії.

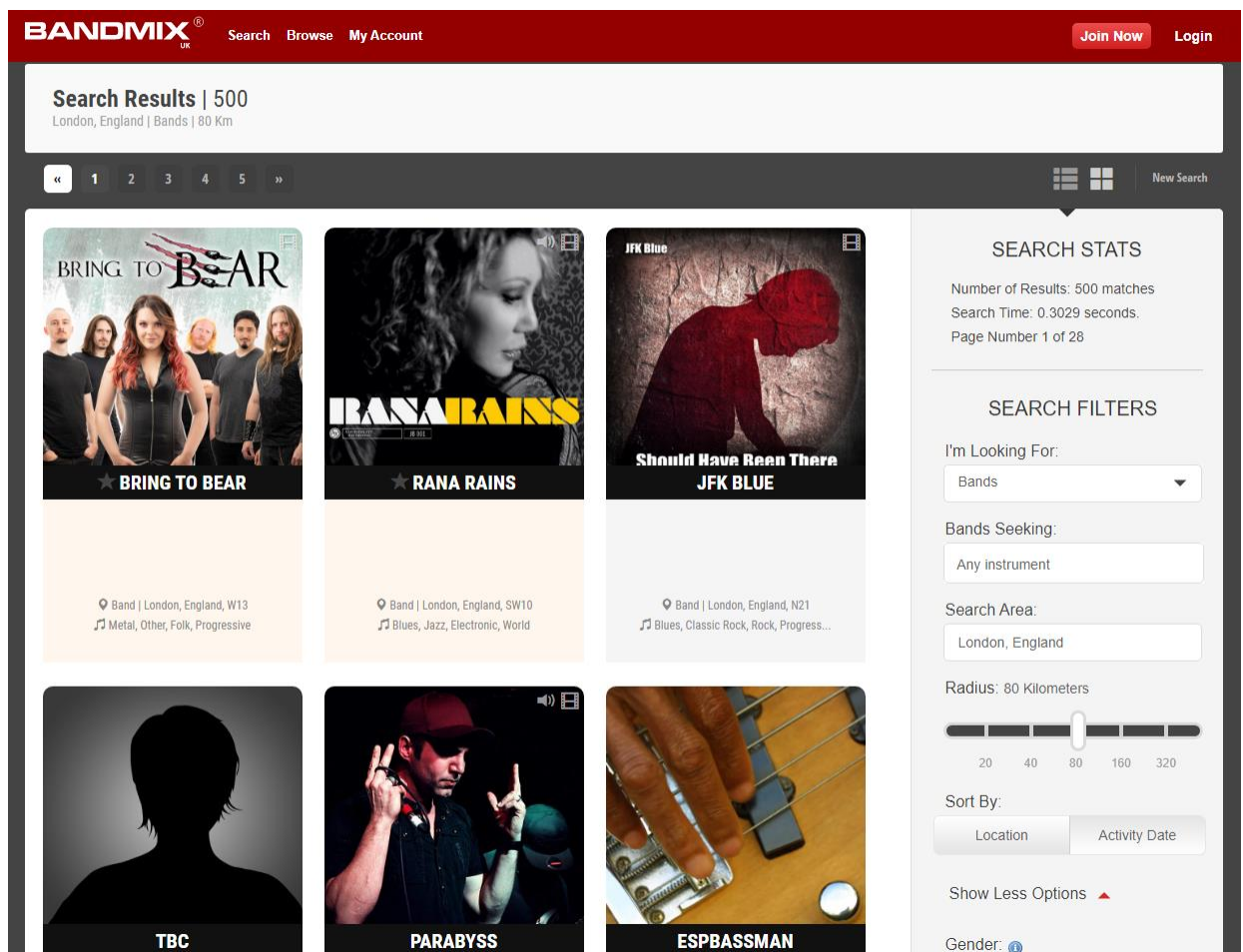


Рисунок 1.2 – Вебсайт Bandmix

У пошуку також вказується інструмент, на якому потрібно грати в колективі або на якому вміє грати музикант, місто з обмеженням радіусу, стать музиканта, віковий інтервал, музичний жанр, рівень залучення до колективу (розвага або участь в турах), необхідність наявності фотографій, аудіофайлів, відео. Це достатньо велика кількість фільтрів, що сприяє отриманню результатів, які відповідають вимогам обох сторін загалом.

Кожен учасник вказує, які колективи і виконавці на нього вплинули, хто бере участь в колективі на даний момент і на якій позиції, які позиції вимагаються, створює власний опис, розміщує фотографії, аудіо, відео, визначає рівень власної залученості, досвід тощо.

Вебсайт Musicianfinder [5] (рис. 1.3) дозволяє виконувати пошук музичного колективу, музиканта або продюсера чи інженера.

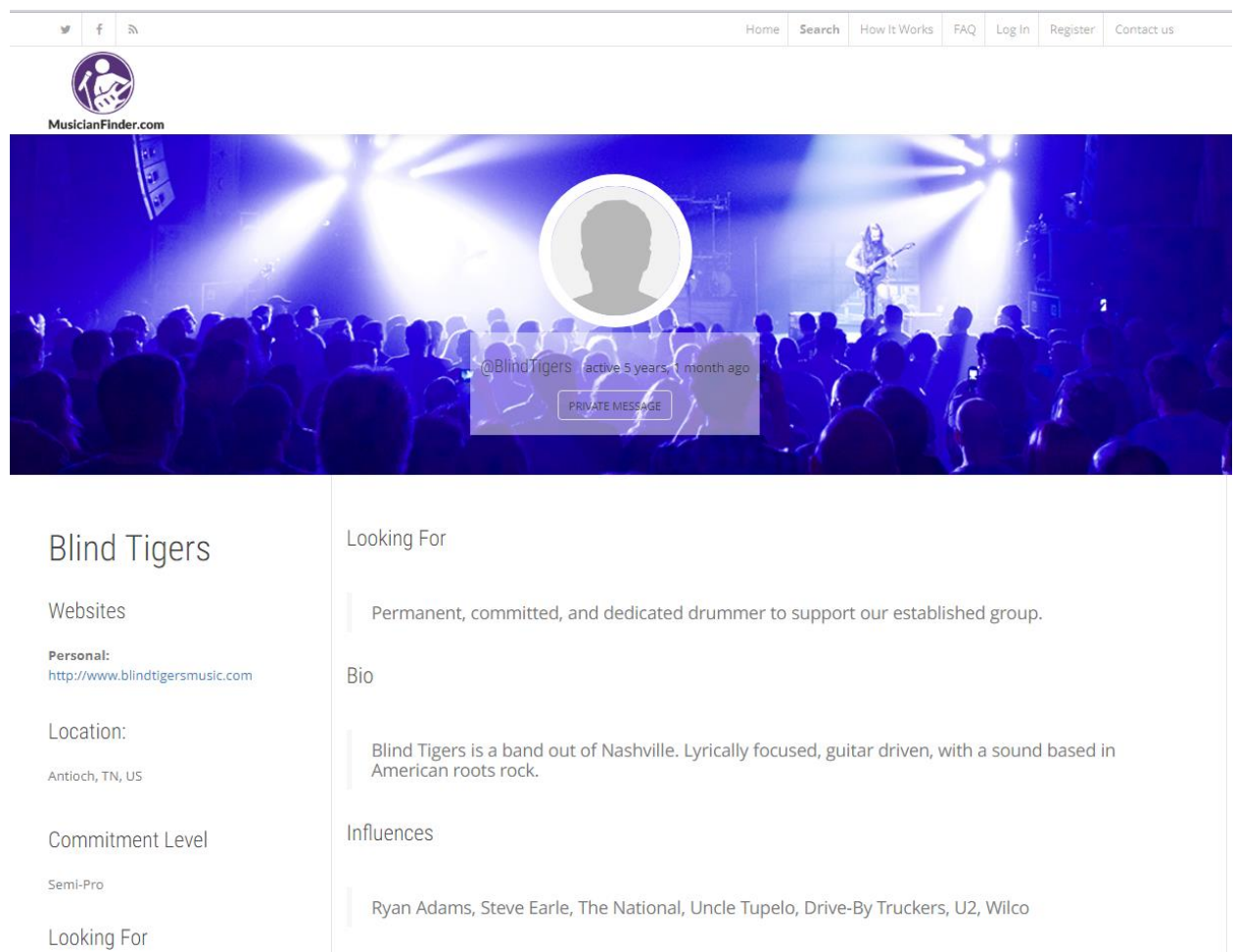


Рисунок 1.3 – Вебсайт Musicianfinder

Пошук виконується за відстанню за поштовим індексом у країні (Бельгія, Канада, Данія, Англія, Франція, Німеччина, Ірландія, Італія, Нідерланди, Норвегія, Польща, Румунія, Шотландія, Швеція, США), за віком, статтю, інструментами, рівнем володіння інструментом, жанром, колективами, які вплинули, а також вказуючи порядок сортування.

Кожне оголошення, а відповідно і профіль містить інформацію про те, які позиції є вакантними, біографічний опис, перелік колективів, які вплинули на виконавця або колектив, фотографії. Дизайн є доволі проблемним, адже не всюди можна навіть прочитати потрібний текст без проблем.

Вебсайт BandFinder [6] (рис. 1.4) дозволяє шукати музичні колективи, музикантів, вчителів у США, Великобританії, Канаді, Швеції.

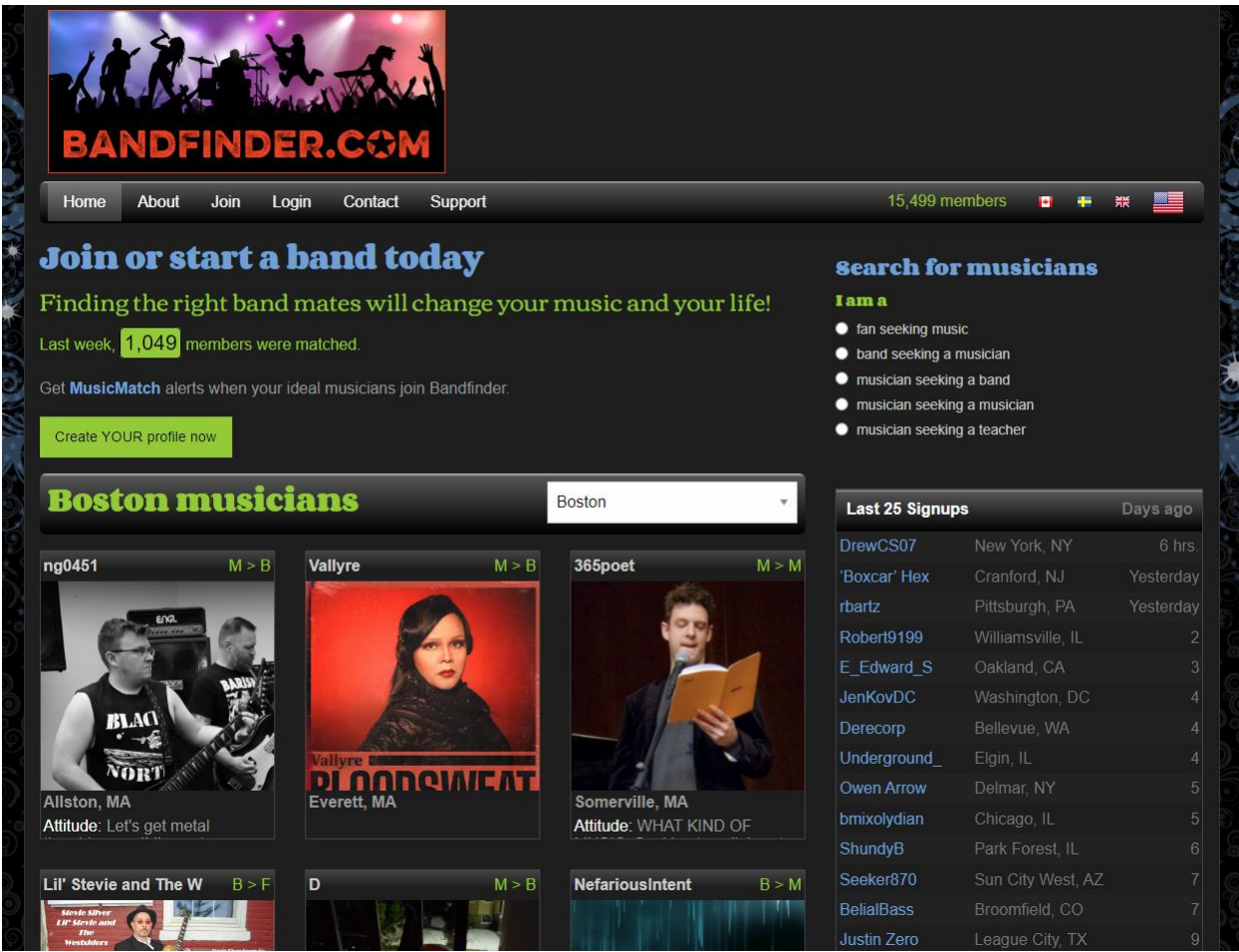


Рисунок 1.4 – Вебсайт BandFinder

При пошуку вказуються країна, поштовий індекс, вміння грати на інструментах, жанри, рівень вмінь, рівень знань музичної теорії, доступність для концертів, для практики. Кожен профіль містить фотографії, посилання, опис, колективи, які вплинули на користувача, наявне обладнання, жанри, в яких грає музику виконавець або колектив, позиції, на які здійснюється пошук, інформація про те, як часто відбуваються тренування, задіяність до концертів, рівень досвіду, знань музичної теорії та досвід участі у записах музики.

Вебсайт Join-A-Band [7] (рис. 1.5) є одним з найвідоміших у цій сфері, хоча має дуже обмежену функціональність.

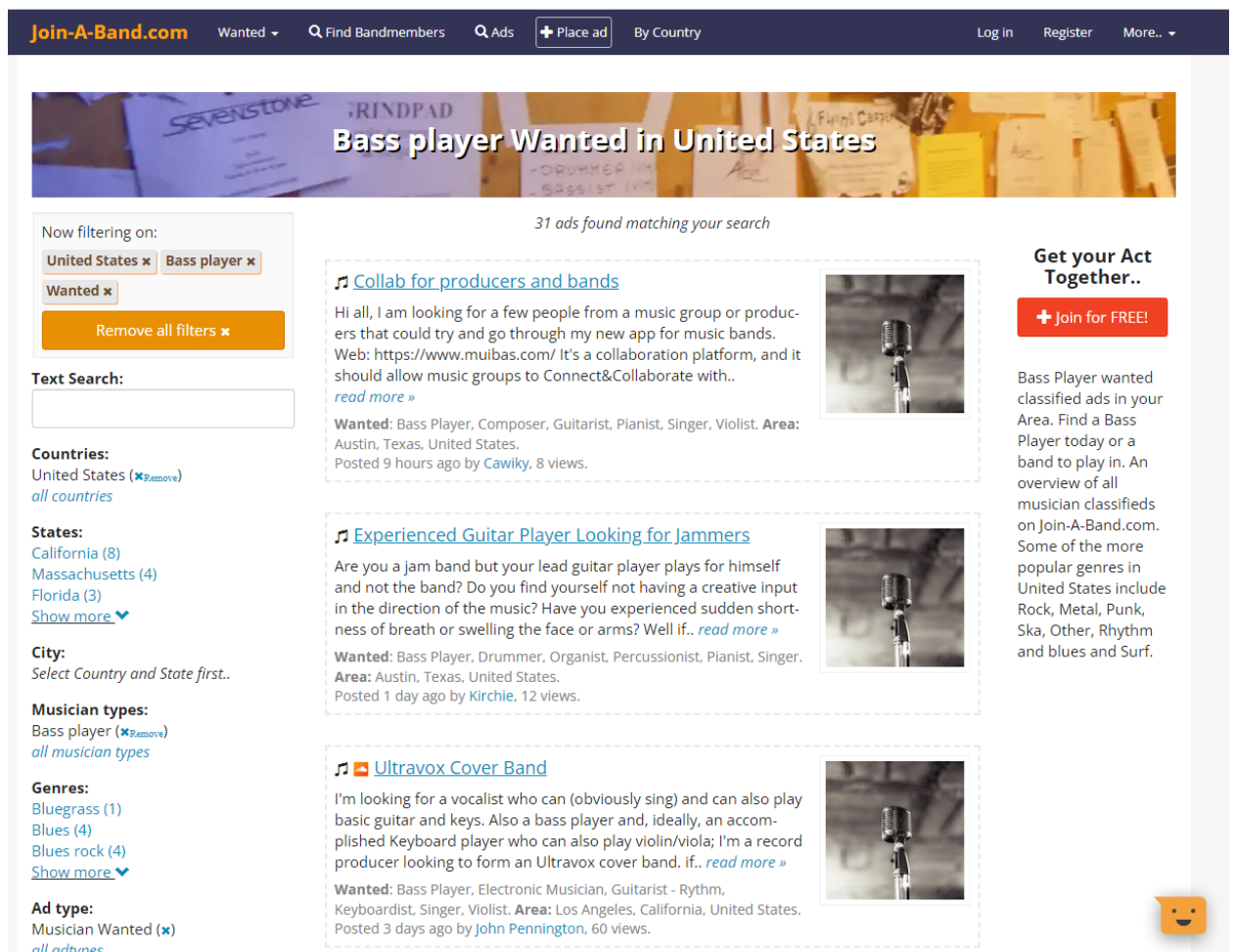


Рисунок 1.5 – Вебсайт Join-A-Band

Відомості вебсайту сприяє зокрема те, що на ньому розташовують оголошення музиканти зі всього світу [8]-[11]. Наприклад, зараз на сайті

міститься 3 оголошення музикантів з України. Кожен музикант, який шукає музичний колектив, може розмістити власну фотографію, описати свою біографію, яка фактично використовується як текст оголошення, розмістити аудіо та відео, визначити місце власного розташування на рівні міста, вказати яку позицію в колективі він готовий займати, а також музичні жанри, які йому подобаються.

Оголошення музичних колективів структуруються за позиціями, за якими шукаються музиканти. Фактично кожне оголошення зі сторони колективу виглядає так само як і оголошення музиканта. Надається можливість відправляти прямі повідомлення.

Пошук виконується за позицією, за текстом в оголошенні, за країною, містом, жанрами.

1.2 Порівняння аналогів

Виконаний огляд дозволив встановити широке коло існуючих аналогів, хоча більшість з них має подібний набір основних функцій, хоча і наявні певні відмінності, наприклад, в параметрах пошуку. Але загалом є певний стандартний набір поглядів на взаємодію музикантів з музичними колективами. Окрім того зазвичай наявне обмежене коло країн, в яких такий пошук можна виконувати, що є проблемою, адже Україна присутня взагалі тільки у одного інструменту для пошуку (табл. 1.1). Тому дипломна розробка з одного боку має цю проблему ліквідувати, дозволивши не виконувати орієнтацію тільки на музичний ринок окремої країни, але і ввести в це коло Україну. З іншого боку дипломна програма має дозволити виконувати пошук музикантів для окремих проєктів. Тобто у музичного колективу може біти декілька ідей, які вони хочуть реалізувати, і ці ідеї можуть не відповідати загальній концепції колективу. У таких випадках музичний колектив чітко позначає, що це обособлений проєкт, він має певну обмежену тривалість, і музикант не стане постійним учасником колективу. Для когось це може бути

додатковою перевагою, для когось ні, але в такому випадку про цю перевагу і недолік музикант знає одразу. Окрім того проєкт дозволяє чітко виділити, вміння грати, на яких саме інструментах потрібне, перелічивши у підсумку всі потрібні. Коли в колективі визначається позиція, то зокрема в багатьох проаналізованих інструментах це не передбачає визначення конкретних інструментів.

Таблиця 1.1 – Порівняння аналогів з дипломною програмою за результатами огляду

Характеристика	Hendrix	Bandmix	BandFinder	Join-A-Band	Дипломна програма
Країни	США	Великобританія	США, Великобританія, Канада, Швеція	Усі, в тому числі Україна	Без обмежень
Пошук музикантами колективів	+	+	+	+	+
Пошук постійних учасників колективів	+	+	+	+	+
Пошук музикантів для проєктів	—	—	—	—	+

1.3 Висновки за розділом 1

До завдань роботи відносяться:

– аналіз існуючих програм пошуку учасників музичного колективу;

- визначення функціональних вимог до програми;
- проєктування програмного забезпечення пошуку учасників музичного колективу;
- реалізація і тестування програми.

Розділ 1 пояснювальної записки (ПЗ) дозволив реалізувати аналіз існуючих програм пошуку учасників музичного колективу, включаючи до нього Hendrix, Bandmix, BandFinder, Join-A-Band. За результатами аналізу визначені відмінності розроблюваного програмного забезпечення, що базуються в основному на можливості роботи щодо пошуку музикантів для участі в окремих проєктах музичного колективу, а також можливості поширення можливостей пошуку на користувачів з України.

2 ПРОЄКТУВАННЯ ПРОГРАМИ

2.1 Вибір засобів проєктування і розробки

Основним засобом розробки обрано мову Python, оскільки вона є простою у використанні, має багато високорівневих засобів, що загалом пришвидшує розробку, підтримує вебфреймворки. Тому далі виконувався вибір вебфреймворку серед Bottle і Django [12] (табл. 2.1).

Таблиця 2.1 – Порівняння вебфреймворків Bottle і Django

Показник	Bottle	Django
Мова програмування	Python	Python
Тип вебфреймворку	Мікрофреймворк	Фреймворк високого рівня
Доступність об'єктно-реляційного відображення	Доступний для підключення	Вбудований
Вбудоване керування даними	Відсутнє	Наявне
Підтримка на рівні документації	Гірша	Краща
Вбудована підтримка вебформ	Відсутня	Наявна
Вбудована аутентифікація	Відсутня	Наявна
Вбудована підтримка безпеки	Відсутня	Наявна

Вебфреймворком обрано Django, зважаючи на те, що порівняно з Bottle він за рахунок більш високого рівня має необхідні засоби на рівні вбудованих, має кращу документацію. Це дозволяє під час розробки

сфокусуватися саме на реалізації функціональних вимог, не використовуючи час на створення засобів безпеки та подібних можливостей, як це було б у випадку використання вебфреймворку Bottle.

2.2 Визначення функціональних вимог до програмного забезпечення

За підсумками огляду, подальшого аналізу існуючих програм і цієї розробки визначено функціональні вимоги до розробки стосовно різних користувачів. За технічним завданням користувач має бути або неавторизованим, або авторизованим музикантом.

Сценарії, що відповідають функціональним вимогам для неавторизованого користувача (рис. 2.1):

- пошук музичного колективу;
- пошук музичного проєкту;
- пошук музиканта;
- перегляд музичних проєктів музичного колективу;
- перегляд оголошень за музичним колективом;
- перегляд оголошень за музикантом;
- перегляд музикантів і музичних колективів у місті;
- ознайомлення з інформацією про музичний колектив;
- ознайомлення з інформацією про музиканта;
- ознайомлення з оголошенням про пошук постійного учасника колективу;
- ознайомлення з оголошенням про пошук учасника музичного колективу;
- ознайомлення з оголошенням музиканта;
- відтворення музики;
- додавання власного профілю музиканта;
- авторизація облікового запису музиканта.

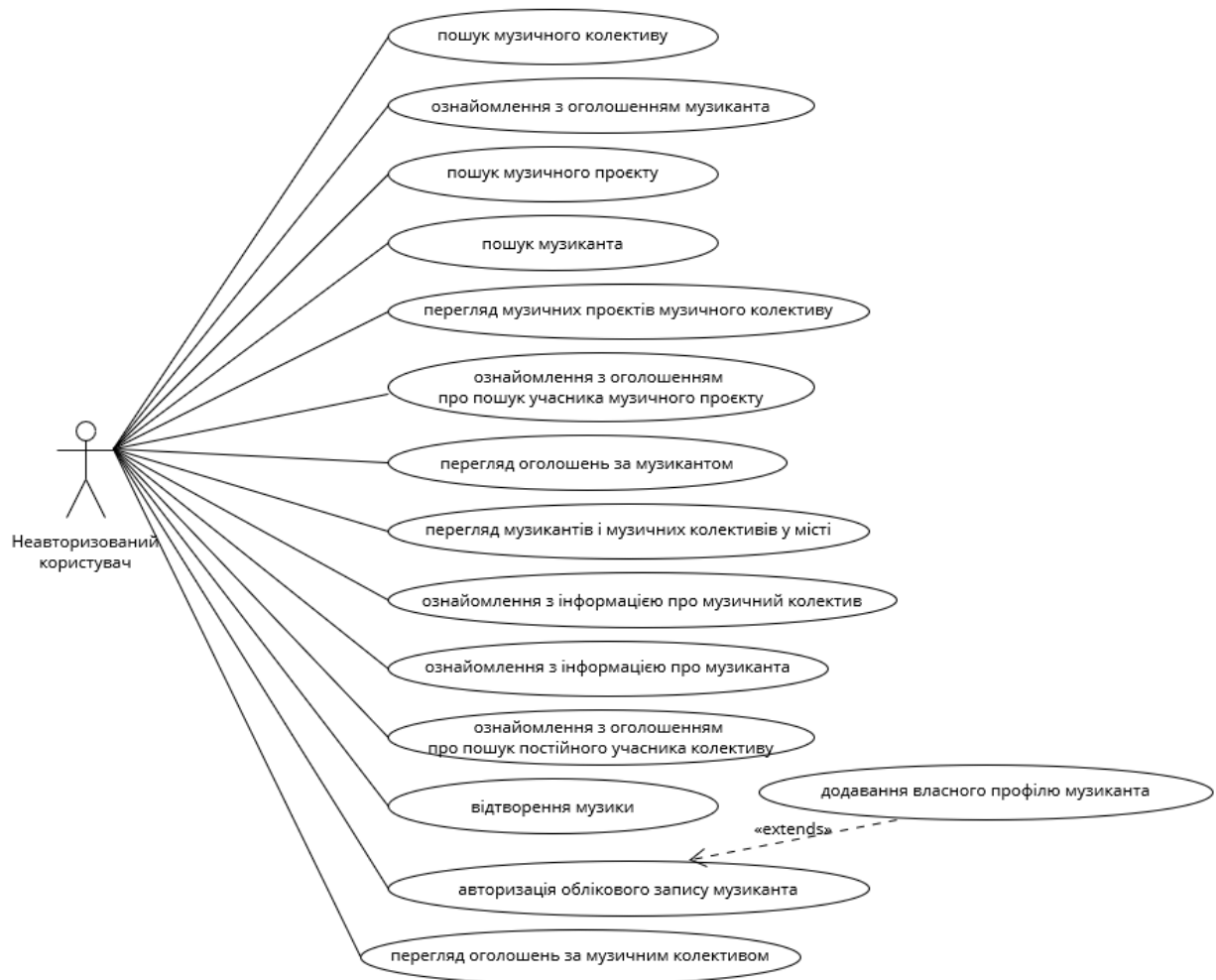


Рисунок 2.1 – Діаграма варіантів використання програми неавторизованим користувачем

Сценарії, що відповідають функціональним вимогам для авторизованого музиканта:

- додавання власного музичного колективу;
- модифікація даних власного музичного колективу;
- додавання учасників музичного колективу;
- вилучення учасників музичного колективу;
- визначення позиції учасника в музичному колективі;
- публікація оголошення про пошук музиканта для постійної участі в музичному колективі;
- модифікація оголошення про пошук музиканта для постійної участі в музичному колективі;

- додавання нового музичного проєкту колективу;
- модифікація даних музичного проєкту;
- визначення керуючих учасників музичного колективу;
- публікація оголошення про пошук музиканта для участі в музичному проєкті;
- модифікація оголошення про пошук музиканта для участі в музичному проєкті;
- публікація оголошення музикантом про пошук музичного колективу;
- модифікація оголошення музикантом про пошук музичного колективу;
- додавання даних музичного обладнання музикантом;
- керування даними музичного обладнання музикантом;
- відправлення короткого запиту і відповіді між музичним гуртом і музикантом;
- перегляд переписки між музичним гуртом і музикантом;
- перегляд переліку переписок між гуртом і музикантом;
- додавання музики музичного колективу;
- керування переліком музики музичного колективу;
- додавання нотатки до оголошення музикантом;
- перегляд занотованих оголошень музикантом;
- перегляд занотованих оголошень усіма учасниками музичного колективу;
- керування нотаткою до оголошення;
- подання заявки на участь у музичному проєкті;
- подання заявки на постійну участь у музичному колективі;
- подання заявки на співправцю з музикантом у музичному колективі;
- перегляд заявок за музичним проєктом;
- перегляд заявок за оголошенням за музичним проєктом;
- перегляд заявок за музичним колективом (рис. 2.2).

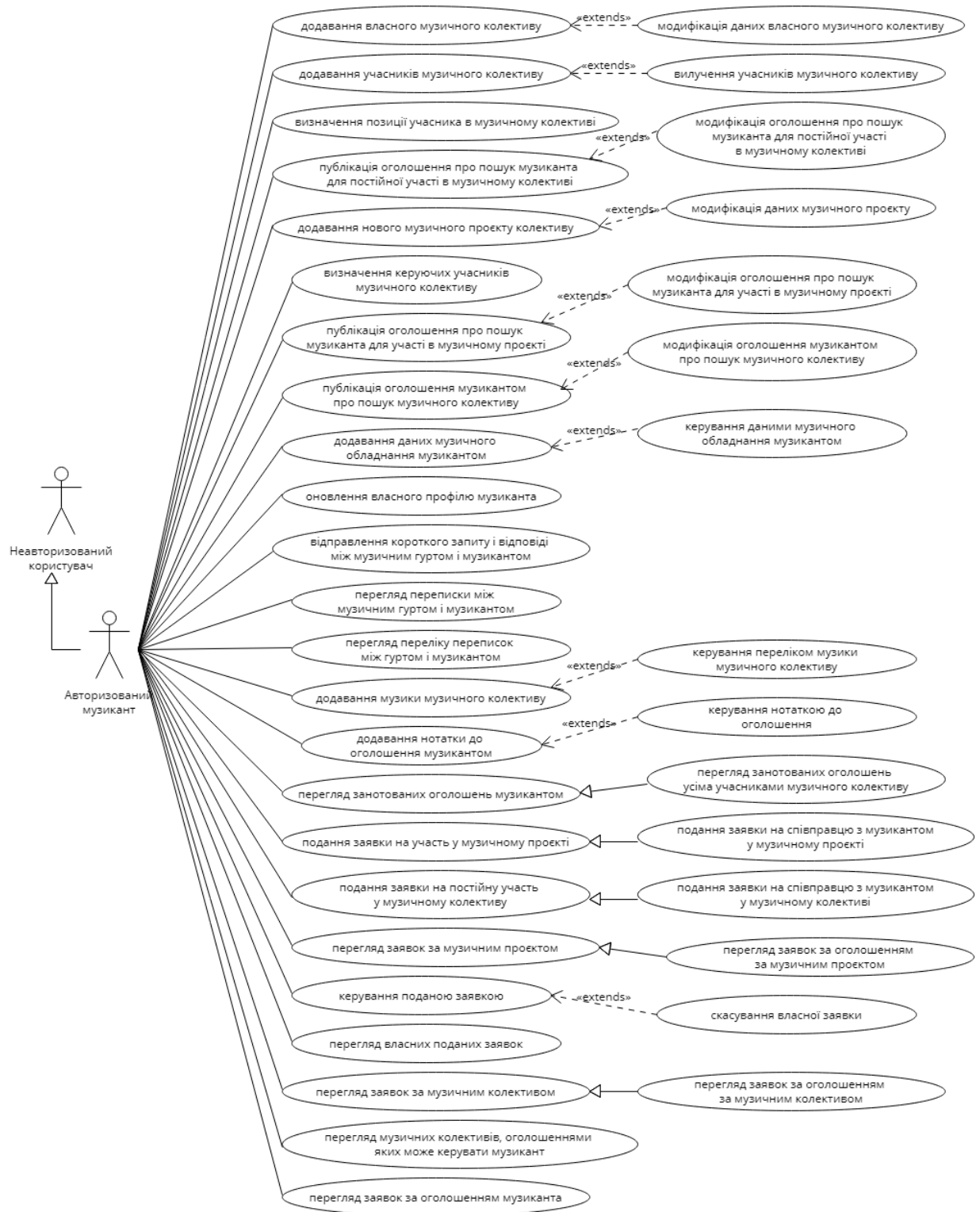


Рисунок 2.2 – Діаграма варіантів використання програми авторизованим музикантом

Також сценаріями авторизованого музиканта є:

- перегляд заявок за оголошенням за музичним колективом;
- перегляд заявок за оголошенням музиканта;

- перегляд музичних колективів, оголошеннями яких може керувати музикант;
- перегляд власних поданих заявок;
- керування поданою заявкою;
- скасування власної заявки;
- подання заявки на співправцю з музикантом у музичному проєкті;
- оновлення власного профілю музиканта.

2.3 Проєктування структури бази даних

Системою керування базами даних (БД) для створення програмного проєкту визначено MySQL, адже це одна з розповсюджених на практиці систем, що до того ж не має додаткових вимог щодо необхідності залучення ресурсів для її підтримки (наприклад, сплати за користування).

Музичний колектив може представляти собою як музичний гурт, так і колектив, який допомагає артисту записувати альбоми, музику та здійснювати виступи. Дані про музичні колективи знаходяться в таблиці Band БД, включаючи поля:

- id – номер музичного колективу, що відіграє в таблиці роль первинного ключа;
- title – назва музичного колективу;
- description – загальна текстова презентація музичного колективу;
- location_city – місто, в якому базується цей музичний колектив;
- photo – фотографія музичного колективу;
- bio – текстова презентація досвіду музичного колективу;
- albums – альбоми, які випустив музичний колектив на даний момент;
- start_year – рік, з якого є активним музичний колектив;
- gigs – орієнтовна кількість концертних виступів, яку дав музичний колектив.

Музичний колектив може не всі дані про себе приводити на розроблюваному вебсайті. Це є нормальною ситуацією, а протилежний випадок сприяв би зменшенню використання розроблюваного вебзастосунку, адже дублювати абсолютно всю інформацію про себе неоптимально. Тому додаткове знайомство з музичним колективом може відбуватися на зовнішніх вебсайтах. Але таких вебсайтів може бути багато. Вони можуть включати і вебсайт музичного колективу, і сторінку на youtube, і сторінку в соціальній мережі facebook, і instagram-сторінку, і сторінку на bandcamp, і сторінку в соціальній мережі twitter, і багато інших ресурсів. При цьому колектив може не мати сторінки на всіх цих вебсайтах, а тому такі посилання винесено в окрему таблицю БД Bandlink, включаючи поля:

- id – номер посилання, за яким представлено інформацію про музичний колектив, відіграє в таблиці роль первинного ключа;
- title – назва посилання, що відображається користувачу (наприклад, youtube);
- band – музичний колектив, якого стосується це посилання, є зовнішнім ключем з таблиці Band;
- link – саме посилання.

Одним зі стандартних засобів категоризації в музиці є музичні жанри. Дані жанрів є дуже простими, адже включають назву самого жанру (поле title) та номер в якості первинного ключа (поле id), і об'єднуються вони в таблиці Genre.

Жанри характеризують стиль музичного колективу, але стиль може виражатися декількома жанрами, в яких колектив грає музику, проте і кожен жанр звичайно ж активно задіюється різними колективами, тому для зв'язування жанрів і музичних колективів потрібна зв'язуюча таблиця BandGenre, що містить тільки зовнішні ключі з двох даних таблиць Band (поле band) і Genre (поле genre).

Музичні колективи складаються з музикантів, з іншого боку музиканти шукають музичні колективи, користуючись програмою. Тому

іншою базовою сутністю в цій предметній області є музиканти, дані про яких знаходяться в таблиці Musician, включаючи поля:

- id – номер музиканта, що відіграє в цій таблиці роль первинного ключа;
- django_user – номер користувача Django, акаунт якого використовується для роботи з програмою музикантом, відіграє роль зовнішнього ключа зі стандартної таблиці User у Django;
- photo – фотографія, якою відображається в програмі музикант;
- bio – текстова презентація досвіду музиканта;
- phone – номер телефону музиканта;
- start_year – рік, з якого займається музикою цей музикант;
- overall_level – загальний рівень музикування цього музиканта;
- music_theory – рівень знання музичної теорії музикантом;
- recording_level – рівень вмінь музиканта стосовно запису музики;
- date_of_birth – дата народження музиканта, необхідна для обчислення віку при пошуку і відображенні на сторінці (повна дата не має відображатися).

Інформація про прізвище, ім'я музиканта не потрібна, адже вона стандартно зберігається в таблиці User Django.

Однак музикант може бути учасником музичного колективу, а може бути самостійним учасником, який шукає музичний колектив або який бере участь у музичному колективі, але шукає ще один музичний колектив або готовий до участі в проєкті. Тому фактично участь музиканта на ринку (оголошення) розміщується в таблиці MusicianAdvert, включаючи поля:

- musician – музикант, який бере участь на ринку музикантів, є зовнішнім ключем з таблиці Musician;
- location_city – місто, в якому проживає і в якому потенційно збирається шукати для себе музичний колектив музикант, відіграє роль зовнішнього ключа з таблиці City;

- `project_opened` – позначка про те, чи готовий музикант брати участь у окремому проєкті чи його цікавить тільки постійна участь в колективі: тільки проєкт, тільки постійна участь або обидва варіанти доступні;

- `practice_availability` – періодичність репетицій, з якою готовий максимально брати участь цей музикант в музичному колективі;

- `gig_availability` – максимальна допустима періодичність, з якою готовий брати участь у концертних виступах музичного колективу цей музикант.

Відношення між таблицями `Musician` і `MusicianAdvert` один до одного, проте музикант може і не мати запису в таблиці `MusicianAdvert`.

Музиканти, які вже беруть участь в музичних колективах, можуть входити одразу до складу декількох колективів, а кожен музичний колектив може складатися з одного і більше учасників. Відповідно інформація про те, який музикант є учасником якого музичного колективу знаходиться в окремій таблиці `Member`, включаючи поля:

- `id` – номер учасника, відіграє роль первинного ключа і характеризує участь музиканта в музичному колективі;

- `band` – музичний колектив, у якому бере участь музикант, є зовнішнім ключем з таблиці `Band`;

- `musician` – музикант, який є учасником музичного колективу, є зовнішнім ключем з таблиці `Musician`;

- `manager` – позначка про те, чи може цей учасник брати на себе роль менеджера в програмі від імені колективу, тобто чи може він створювати оголошення, чи може він писати повідомлення музикантам від імені колективу, керувати запитами інших музикантів щодо участі в музичному колективі;

- `start_time` – час, з якого музикант є учасником цього музичного колективу;

- `active` – позначка, яка визначає, чи є учасником цього музичного колективу на даний момент цей музикант.

Учасники музичних колективів можуть брати участь у колективі на різних позиціях. Наприклад, один учасник може виконувати і роль вокаліста, і гітариста. В іншому колективі це може бути роль вокаліста і барабанщика. Тобто комбінації можуть бути різними і включати різні позиції в колективі. Тому збереження цих даних відбувається через окрему таблицю БД MemberPosition, яка є зв'язною таблицею між таблицями Member і Position, що містить тільки зовнішні ключі з двох даних таблиць Member (поле member) і Position (поле position).

Позиція (роль) у музичному колективі зберігається в таблиці Position і визначається тільки первинним ключем id, назвою позиції title та фотографією photo.

Участь музиканта на ринку музикантів визначається не тільки самим оголошенням в таблиці MusicianAdvert, але і другою складовою, яка визначає, на які позиції претендує на даний момент музикант. Ці дані зберігаються в таблиці Capability, включаючи поля:

- id – номер здібності музиканта, за якою він претендує на участь в музичному колективі, відіграє в цій таблиці роль первинного ключа;
- musician_advert – оголошення про участь музиканта на ринку, якого стосується ця здібність, є зовнішнім ключем з таблиці MusicianAdvert;
- position – позиція, на яку в музичному колективі претендує музикант;
- experience – досвід (кількість років) на цій позиції у колективах у музиканта до цього моменту;
- active – позначка про те, чи є це актуальним для музиканта на цей момент.

Однак сама участь музиканта має бути деталізована далі, оскільки важливою може бути не тільки практика на певній позиції у колективі, але і в принципі досвід гри на певному інструменті. Ці дані зберігаються в таблиці CapabilityInstrument, щоб підкреслити, якими саме інструментами за цією позицією оволодів музикант, включаючи поля:

- `carability` – участь на ринку музикантів, якої стосується цей музичний інструмент, на якому вміє грати музикант, є зовнішнім ключем з таблиці `Carability`;

- `instrument` – музичний інструмент, на якому вміє грати музикант, є зовнішнім ключем з таблиці `Instrument`.

Музичні колективи для привертання уваги можуть виставляти музику, яка була ними записана. Це можуть робити і окремі музиканти, але тоді вони повинні спочатку внести інформацію про колектив, який записував цю музику, а тоді вже ця музика з'явиться відносно них. Колектив може складатися і з одного музиканта, тому це не обмежує презентацію своєї музики. Дані про музику, фактично музичні треки, зберігаються в таблиці `Music`, включаючи поля:

- `id` – номер музичного треку, відіграє в цій таблиці роль первинного ключа;

- `band` – музичний колектив, який виконує цей трек, є зовнішнім ключем з таблиці `Band`;

- `track_title` – назва музичного треку;

- `track_link` – посилання на музичний трек;

- `track_type` – тип музичного треку (відео або аудіофайл).

Короткі повідомлення, які надсилаються між музичним колективом і музикантами, зберігаються в таблиці `Notice`, включаючи поля:

- `id` – номер короткого повідомлення, відіграє в цій таблиці роль первинного ключа;

- `member` – учасник музичного колективу, якого стосується це коротке повідомлення, є зовнішнім ключем з таблиці `Member`;

- `musician` – музикант, якого стосується це коротке повідомлення, є зовнішнім ключем з таблиці `Musician`;

- `source` – джерело повідомлення: було створено учасником музичного колективу або музикантом, який виконує запит або відповідає на запит;

- `time_mark` – час, коли було створено повідомлення.

За сформульованою логікою усі оголошення з боку музичного колективу стосуються або пошуку повноцінного учасника, або участі в окремому проєкті музичного колективу. Оголошення, які стосуються участі в музичному колективі загалом, знаходяться в таблиці MemberAdvert, включаючи поля:

- id – номер оголошення про пошук музиканта як учасника музичного колективу, відіграє в цій таблиці роль первинного ключа;
- band – музичний колектив, який створив це оголошення, є зовнішнім ключем з таблиці Band;
- position – позиція, на яку потребується в музичному колективі музикант, є зовнішнім ключем з таблиці Position;
- active – позначка про те, чи є це оголошення в даний момент активним;
- practice_availability – періодичність, з якою плануються репетиції в музичному колективі;
- gig_availability – періодичність, з якою плануються концертні виступи цього музичного колективу;
- experience – мінімальний досвід, який вимагається для участі в цьому музичному колективі за цією позицією;
- overall_level – мінімальний загальний рівень музикування, який вимагається від музиканта для участі в музичному колективі на цій позиції;
- music_theory – мінімальний рівень знання музичної теорії, який вимагається від музиканта для участі в музичному колективі на цій позиції;
- recording_level – мінімальний рівень вмінь відносно запису музики, який вимагається від музиканта для участі в музичному колективі на цій позиції;
- requirements – інші вимоги у текстовому вигляді, які вимагаються від музиканта для участі в музичному колективі на цій позиції;
- create_time – час, коли було створено це оголошення;

- `age_from` – мінімальний вік музиканта, необхідний для участі в музичному колективі на цій позиції;

- `age_to` – максимальний вік музиканта, допустимий для участі в музичному колективі на цій позиції.

Для оголошення, яке стосується участі в музичному колективі загалом, можуть додатково формулюватися вимоги щодо вміння гри на окремих музичних інструментах. Тобто гітарист повинен вміти грати на акустичній гітарі, на банджо тощо. Дані цієї деталізації зберігаються в таблиці `MemberAdvertInstrument`, включаючи поля:

- `member_advert` – оголошення, що стосується участі музиканта в музичному колективі загалом, до якого висувається вимога володіння інструментом, є зовнішнім ключем з таблиці `MemberAdvert`;

- `instrument` – інструмент, на якому повинен вміти грати музикант, є зовнішнім ключем з таблиці `Instrument`.

Друга частина оголошень стосується участі в окремому проєкті музичного колективу. Тоді музикант стає не постійним учасником музичного колективу, а тільки, наприклад, бере участь в записі певного альбому або турі музичного колективу. Дані про такі самі проєкти музичного колективу зберігаються в таблиці `Project`, включаючи поля:

- `id` – номер музичного проєкту, відіграє в цій таблиці роль первинного ключа;

- `band` – музичний колектив, який збирається здійснити цей проєкт, є зовнішнім ключем з таблиці `Band`;

- `title` – назва музичного проєкту;

- `idea` – текстова презентація ідеї проєкту;

- `photo` – фотографія, якою візуалізується проєкт;

- `create_time` – час внесення даних про музичний проєкт;

- `period_start` – час, з якого розпочинається проєкт;

- `period_end` – час, коли завершується проєкт.

Проекти можуть відрізнятися за жанрами від тих, які характеризують музичний колектив загалом. Це можуть бути певні експерименти. Тому для проєкту музичного колективу жанри потрібно визначити окремо. Кожен проєкт може характеризуватися різними жанрами, а один жанр може характеризувати різні проєкти. Тому створено зв'язуючу таблицю ProjectGenre, що містить тільки зовнішні ключі з двох даних таблиць Project (поле project) та Genre (поле genre).

Саме оголошення про пошук музиканта для участі в музичному проєкті є окремою сутністю, оскільки кожен проєкт може мати декілька оголошень одразу, тобто потребувати музикантів для гри на різних інструментах, а кожне оголошення стосується одного інструменту. Дані цих оголошень про пошук музиканта для участі в музичному проєкті зберігаються в таблиці ProjectAdvert, включаючи поля:

- id – номер оголошення про пошук музиканта для участі в музичному проєкті, відіграє в цій таблиці роль первинного ключа;
- project – музичний проєкт, якого стосується оголошення, є зовнішнім ключем з таблиці Project;
- active – позначка про активність оголошення на даний момент;
- experience – мінімальний досвід гри на цьому інструменті, який необхідний для участі в цьому проєкті;
- music_theory – рівень знань стосовно музичної теорії, який необхідний для участі в цьому проєкті;
- recording_level – рівень досвіду участі в записі музики, який необхідний для участі в цьому проєкті;
- requirements – інші вимоги у текстовому вигляді, які вимагаються від музиканта для гри в цьому музичному проєкті на цьому музичному інструменті;
- instrument – інструмент, на якому необхідно грати в цьому музичному проєкті, є зовнішнім ключем з таблиці Instrument;

- age_from – мінімальний вік музиканта для участі в музичному проєкті;
- age_to – максимальний вік музиканта для участі в музичному проєкті;
- overall_level – необхідний мінімальний загальний рівень музикування, який вимагається від музиканта для участі в музичному проєкті.

Звичайно, що кожного оголошення стосується інша сторона взаємодії, яка передбачає подачу заявок на участь в музичному колективі або проєкті за оголошенням. Тобто це заявки від окремих музикантів. Вони розділені на 2 окремі сутності.

Перша сутність визначає заявки музикантів на участь в музичному колективі. Заявки можуть походити не тільки від музикантів до музичних колективів, але і від музичних колективів до музикантів, таким чином представляючи пропозицію долучитися до них. Дані цих заявок у загальному вигляді (як від музикантів до колективів, так і від колективів до музикантів) зберігаються в таблиці MemberApplication, включаючи поля:

- id – номер заявки музиканта на участь в музичному колективі, відіграє в цій таблиці роль первинного ключа;
- musician_advert – оголошення музиканта, якого стосується заявка, є зовнішнім ключем з таблиці MusicianAdvert;
- member_advert – оголошення про пошук музиканта як учасника музичного колективу, якого стосується заявка, є зовнішнім ключем з таблиці MemberAdvert;
- source – ініціатор заявки: музикант або музичний колектив;
- details – коментар до заявки зі сторони, яка подає заявку;
- note – коментар до заявки зі сторони, яка оцінює заявку;
- mark – оцінка цієї заявки стороною, яка створила оголошення;
- active – позначка про активність заявки музиканта на участь в музичному колективі (стан заявки);

– create_time – час, коли було створено заявку.

Друга сутність визначає заявки музикантів на участь в проєкті музичного колективу або заявки з пропозицією музичного колективу музиканту долучитися до їхнього проєкту. Дані цих заявок зберігаються в таблиці ProjectApplication, включаючи поля:

– id – номер заявки музиканта на участь в музичному проєкті колективу, відіграє в цій таблиці роль первинного ключа;

– musician_advert – оголошення музиканта, якого стосується заявка, є зовнішнім ключем з таблиці MusicianAdvert;

– project_advert – оголошення про пошук музиканта для участі в музичному колективі, якого стосується заявка, є зовнішнім ключем з таблиці ProjectAdvert;

– source – ініціатор заявки: музикант або музичний колектив;

– details – коментар до заявки;

– details – коментар до заявки зі сторони, яка подає заявку;

– note – коментар до заявки зі сторони, яка оцінює заявку;

– active – позначка про активність заявки музиканта на участь в музичному проєкті (стан заявки);

– create_time – час, коли було створено заявку.

Дані про самі інструменти, які описувалися вище, зберігаються в таблиці Instrument, включаючи поля id, що відіграє в цій таблиці роль первинного ключа, назву інструмента title та фотографію photo.

Дані про міста, в яких можуть знаходитися музиканти або музичні колективи, знаходяться в таблиці City, включаючи поля id, що відіграє в цій таблиці роль первинного ключа, назву міста title та країну міста у вигляді поля country, що є зовнішнім ключем з таблиці Country. Дані ж про країни розміщені в таблиці Country, яка містить назву країни в полі title та номер в полі id, що відіграє в цій таблиці роль первинного ключа.

Одним з орієнтирів при пошуку однодумців є визначення музичних колективів, які подобаються музикантам або які впливали на музику

музичного колективу. Такі музичні колективи фактично не представлені учасниками в програмі, є класичними виконавцями. Вони винесені в окрему таблицю Influence, яка в полях своїх зберігає номер id, що відіграє в цій таблиці роль первинного ключа, назву колективу title.

Відповідно вплив музичних колективів стосується колективів загалом або окремих музикантів. Вплив на колективи визначається в зв'язуючій таблиці BandInfluence, яка містить зовнішні поля band (таблиця Band) та influence (таблиця Influence). Вплив на музикантів визначається в зв'язуючій таблиці MusicianInfluence, яка містить зовнішні поля musician (таблиця Musician) та influence (таблиця Influence).

Окрім інструментів загалом зберігаються дані про конкретне обладнання, яким користуються музиканти. Обладнання – це фактично конкретні екземпляри музичних інструментів. Бренди обладнання зберігаються в таблиці EquipmentBrand, що включає поля номера бренду id, який відіграє в цій таблиці роль первинного ключа, на назви бренду title.

Саме обладнання зберігається в таблиці Equipment, включаючи поля:

- id – номер обладнання, відіграє в цій таблиці роль первинного ключа;
- equipment_type – музичний інструмент, яким представлено цей екземпляр обладнання, є зовнішнім ключем з таблиці Instrument;
- brand – бренд (виробник) обладнання, є зовнішнім ключем з таблиці Brand;
- musician – музикант, який користується цим обладнанням, є зовнішнім ключем з таблиці Musician;
- production_year – рік виробництва;
- description – загальний опис обладнання;
- photo – фотографія обладнання.

Якщо оголошень може бути багато, користувач бажає обдумати власний вибір, а тільки потім подавати заявки, то він повинен мати можливість зберегти оголошення. Це стосується оголошень про постійну

участь у музичному колективі або про участь у музичному проєкті. Музикант повинен мати можливість зберегти такі оголошення. Так само і музикант як учасник музичного колективу, коли цей колектив виконує пошук музикантів або коли він бажає придивитися до кандидатур, а потім запропонувати долучити когось до колективу (тобто вакансія з'явиться як результат пошуку), повинен мати можливість зберегти оголошення інших музикантів.

Дані збережених оголошень представляються нотатками, в яких користувач може робити відмітки про їх важливість для себе (релевантність), текстові відмітки. Дані нотаток про оголошення щодо пошуку постійних учасників музичного колективу зберігаються в таблиці MemberAdvertNote (рис. 2.3), включаючи поля:

- id – номер нотатки до оголошення про пошук постійного учасника музичного колективу, відіграє в цій таблиці роль первинного ключа;
- musician – музикант, який зберіг це оголошення, якого стосується нотатка, є зовнішнім ключем з таблиці Musician;
- member_advert – оголошення про пошук музиканта як учасника музичного колективу, якого стосується нотатка, є зовнішнім ключем з таблиці MemberAdvert;
- level – рівень релевантності або будь-яка інша оцінка, виставлена музикантом цьому оголошенню;
- note – текст нотатки;
- safety – рівень доступу до нотатки.

Дані нотаток про оголошення щодо пошуку музикантів для музичного проєкту колективу зберігаються в таблиці ProjectAdvertNote, включаючи поля:

- id – номер нотатки до оголошення про пошук музиканта для музичного проєкту колективу, відіграє в цій таблиці роль первинного ключа;
- musician – музикант, який зберіг це оголошення, якого стосується нотатка, є зовнішнім ключем з таблиці Musician;

- project_advert – оголошення про пошук музиканта для музичного проєкту колективу, якого стосується нотатка, є зовнішнім ключем з таблиці ProjectAdvert;
- level – рівень релевантності або будь-яка інша оцінка, виставлена музикантом цьому оголошенню;
- note – текст нотатки;
- safety – рівень доступу до нотатки.

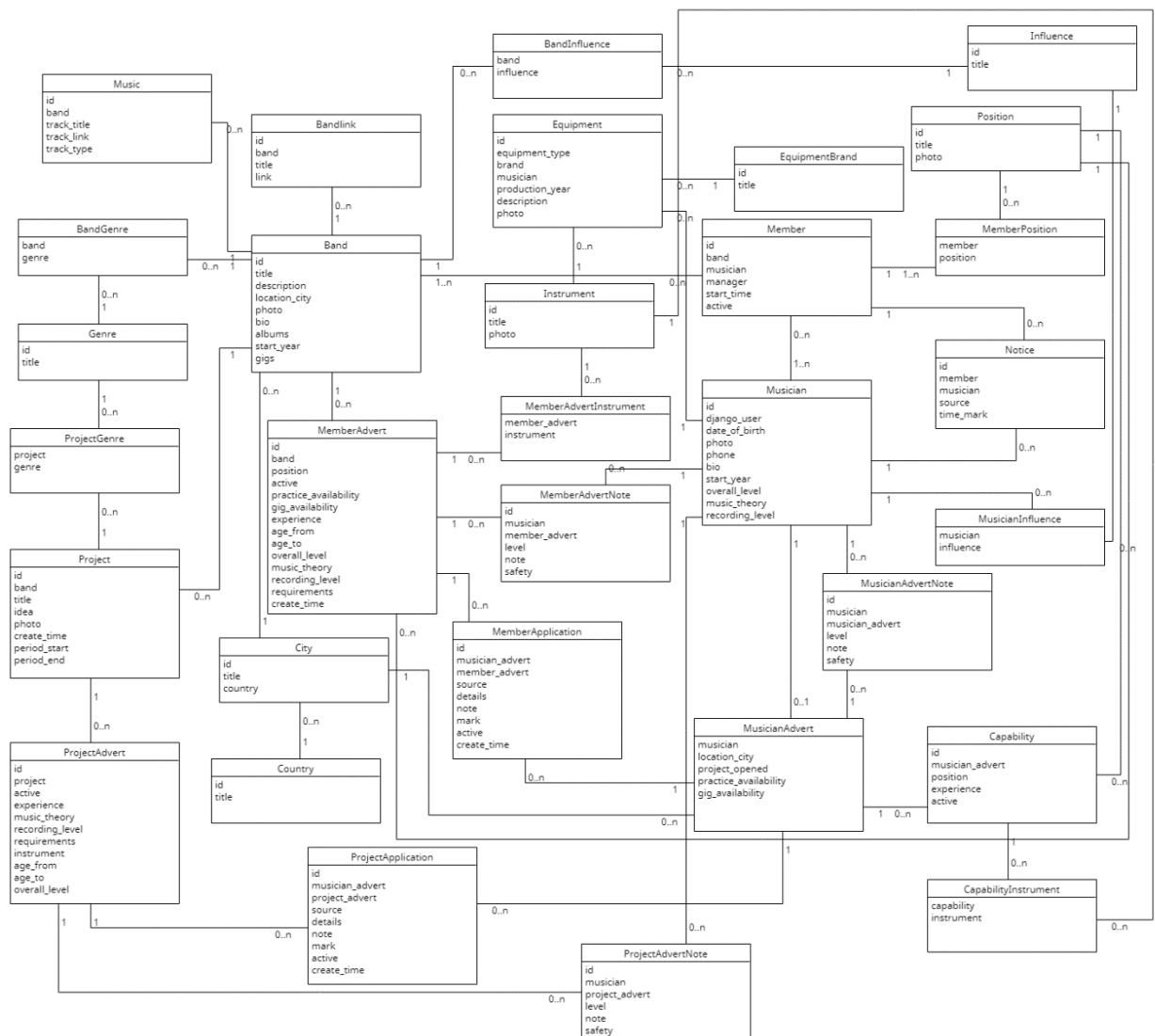


Рисунок 2.3 – Схема БД

Дані нотаток про оголошення щодо пошуку музикантів для музичного проєкту колективу зберігаються в таблиці MusicianAdvertNote, включаючи

поля:

- id – номер нотатки до оголошення музиканта, відіграє в цій таблиці роль первинного ключа;
- musician – музикант, який зберіг це оголошення, якого стосується нотатка, є зовнішнім ключем з таблиці Musician;
- musician_advert – оголошення музиканта, якого стосується нотатка, є зовнішнім ключем з таблиці MusicianAdvert;
- level – рівень релевантності або будь-яка інша оцінка, виставлена музикантом цьому оголошенню;
- note – текст нотатки;
- safety – рівень доступу до нотатки.

2.4 Висновки за розділом 2

Розділ 2 ПЗ містить результати проектування програми: вибір мови Python, системи керування БД MySQL, фреймворку Django, моделювання роботи з програмою на основі вимог, визначення структури БД.

3 РОЗРОБКА ПРОГРАМИ

3.1 Порядок функціонування програми

Зважаючи на важливість пошуку, що є основою роботи програми, необхідно було спершу визначити порядок роботи пошуку перед тим, як його представити у вигляді коду.

Алгоритми, які визначають пошук постійних учасників музичного колективу, учасників музичного проєкту (рис. 3.1), музикантів, є по суті подібними між собою.

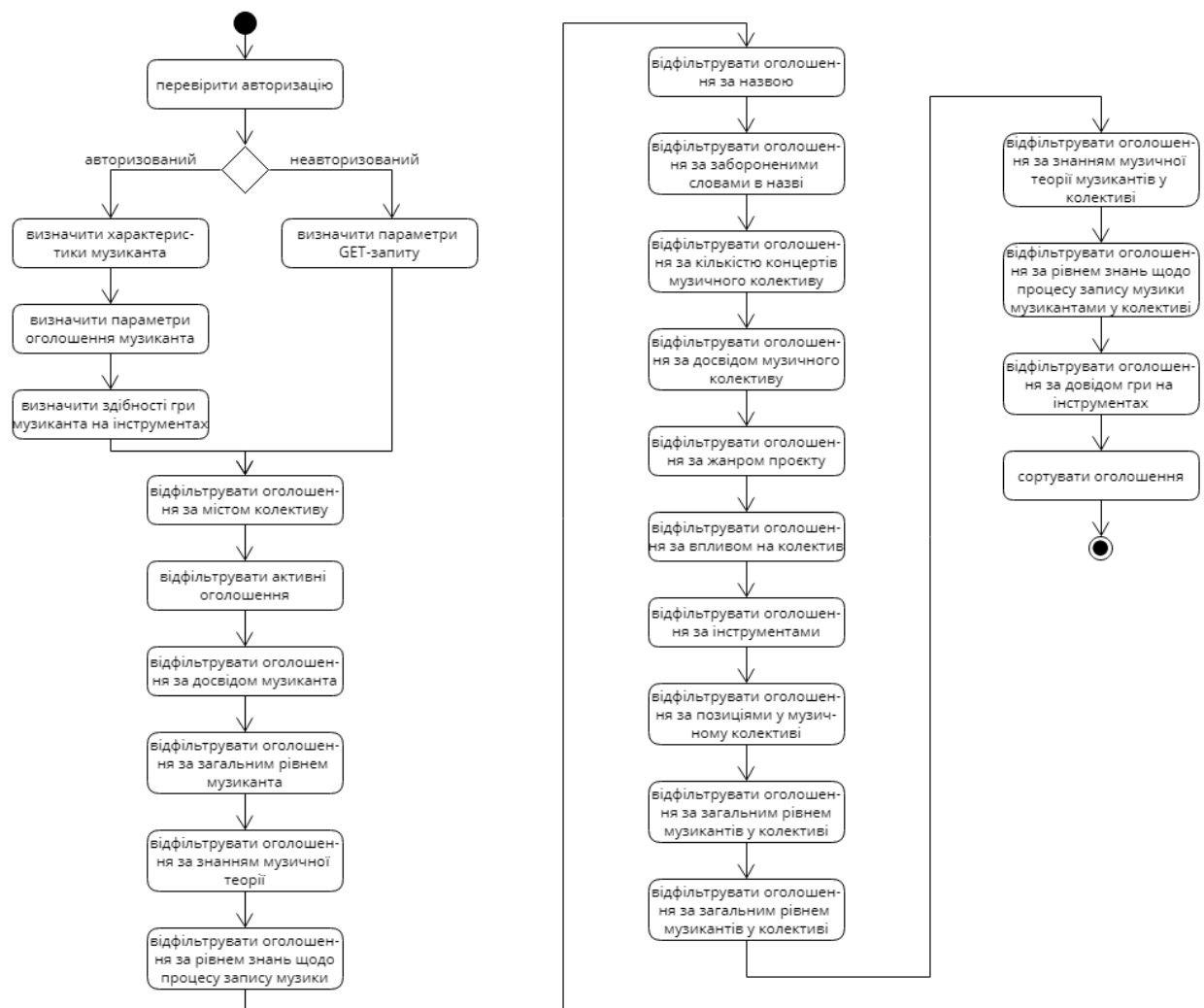


Рисунок 3.1 – Діаграма діяльності пошуку музичних проєктів

3.2 Рішення з реалізації одиниць додатку

Основою функціонування програмного забезпечення для користувача є забезпечення функцій пошуку. До них же належать і пошук музичних проєктів, що є фактично однією з головних особливостей програми порівняно з аналогами.

Пошук музичного колективу реалізований на основі функції відображення `seek_for_band_view` та шаблону вебсторінки `seek_band.html`. Функція відображення виконує наповнення даними шаблону, реалізованого з використанням шаблону [13], через звернення до моделі `MemberAdvert`. Вона працює фактично в 2 режимах. Перший визначає вигляд сторінки для авторизованого музиканта, а другий – для неавторизованого користувача. Пошук для неавторизованого користувача визначає всі параметри через запит до вебсторінки, витягаючи усі значення параметрів з запиту, а пошук для авторизованого музиканта визначає більшу частину з того оголошення, яке створив раніше музикант. Тобто у другому випадку передбачається, що музикант повинен спочатку сам визначити своє оголошення, а потім, якщо він не очікує дій від музичного колективу і виконує пошук колективу самостійно, основні значення параметрів пошуку обираються з цього оголошення. Фактично це спрощує пошук для авторизованого музиканта.

Якщо користувач є авторизованим, то місто, в якому знаходиться музичний колектив, тобто де будуть відбуватися репетиції зокрема, а можливо і виступи музичного колективу, витягається з міста, яке визначено фактично в профілі музиканта на основі моделі `Musician`. Якщо користувач неавторизований, то місто береться з параметра `city`. Після встановлення міста зі всіх об'єктів БД з набору моделі `MemberAdvert` відфільтровуються ті оголошення, в яких музичний колектив знаходиться в цьому місті.

Наступною перевіряється позиція, на якій здатен працювати музикант. Ця інформація відбирається з моделі `Sarability`, що відповідає оголошенню авторизованого музиканта. Усі оголошення, які містять такі позиції,

залишаються в результатах. Якщо користувач не авторизувався, то потрібна позиція відбирається з параметру `position` запиту.

У подальшому за кожним параметром виконується додавання фільтрів, а безпосередній відбір виконується вже на завершення після встановлення всіх фільтрів.

Параметр `practice` запиту або поле `practice_availability` з оголошення авторизованого музиканта дозволяють відібрати всі оголошення, в яких періодичність репетицій менше або дорівнює тій, з якою готовий максимально брати участь авторизований музикант в музичному колективі.

Параметр `gig` запиту або поле `gig_availability` з оголошення авторизованого музиканта дозволяють відібрати всі оголошення, в яких періодичність концертних виступів є меншою або рівною періодичності концертних виступів, з якою готовий у них брати участь авторизований музикант.

Для відфільтровування оголошень за досвідом відбираються ті оголошення, де досвід менше або дорівнює встановленому. Встановлюється досвід або через параметр `experience` для неавторизованого користувача, або через врахування досвіду, вказаного в полі `start_year` для музиканта, що є роком, з якого займається музикою цей музикант. Для цього з поточної дати віднімається ця кількість років і отримане значення враховується при відфільтровуванні вказаним вище чином.

Параметр `level` запиту або поле `overall_level` з профілю авторизованого музиканта дозволяють відібрати всі оголошення, в яких загальний рівень музикування встановлено на рівні, що нижче або дорівнює рівню музикування авторизованого музиканта.

Для відфільтровування оголошень за вимогами до віку музиканта відбираються ті оголошення, де нижня вікова межа в полі `age_from` менша за вік музиканта, який обчислюється через знаходження кількості років між поточною датою і датою народження музиканта, або дорівнює йому, а нижня вікова межа `age_to` більша або дорівнює віку музиканта. Для

неавторизованого музиканта поточний вік не обчислюється, а встановлюється на основі значення параметра `age` запиту.

Параметр `theory` запиту або поле `music_theory` з профілю авторизованого музиканта дозволяють відібрати всі оголошення, в яких рівень знання музичної теорії музикантом встановлено на рівні, що нижче або дорівнює рівню знання музичної теорії авторизованого музиканта.

Параметр `rec` запиту або поле `recording_level` з профілю авторизованого музиканта дозволяють відібрати всі оголошення, в яких рівень вмінь музиканта стосовно запису музики встановлено на рівні, що нижче або дорівнює рівню вмінь стосовно запису музики авторизованого музиканта.

З параметра `search` запиту вибирається текст, який перевіряється в назві або загальній текстовій презентації музичного колективу. Якщо в назву або загальну текстову презентацію він входить, то таке оголошення залишається в результатах пошуку. Одночасно з параметра `forbid` розбирається перелік заборонених слів. Значення параметра розділяється пробілами на окремі слова. Якщо хочаби одне слово вдалось виділити, то всі оголошення, в назві або загальній текстовій презентації яких знаходиться хоча би одне з виділених слів, відкидаються з результатів пошуку.

Частина параметрів пошуку дозволяє і неавторизованому, і авторизованому музиканту визначити свої власні додаткові вимоги до музичних колективів, якщо музикант бажає такі вимоги встановити.

Параметри `gigs1` і `gigs2` визначають мінімальну і максимальну орієнтовну кількість концертів музичного колективу. Відбираються ті музичні колективи, кількість концертів яких не перевищує максимальну межу (`gigs2`) і не менша за мінімальну кількість (`gigs1`).

Параметри `years1` і `years2` визначають мінімальну і максимальну кількість років, протягом яких існує музичний колектив. Відбираються ті музичні колективи, рік створення яких не менше року, обчисленого шляхом віднімання від поточного року максимальної кількості років (`years2`), і не

більше за рік, обчислений шляхом віднімання від поточного року мінімальної кількості років (years1).

Параметр `genres` визначає перелік жанрів, у яких має грати музичний колектив. Цей параметр є рядком, але він перетворюється на перелік шляхом розділення рядка за входженням коми. Далі за кожним виділеним жанром відбувається перевірка на основі формування загальної умови. Якщо в параметрі `genreadd` встановлено істинне значення, то перевіряється, щоб всі вказані жанри входили до переліку жанрів, у яких грає музика музичний колектив. Якщо вказане хибне значення, то перевіряється, щоб колектив грав музику хочаби в одному з вказаних жанрів.

Якщо музикант авторизований, то використовується перелік музикантів або колективів, які вплинули на нього, для пошуку аналогічного музичного колективу. Тобто той набір колективів, що зберігається в переліку `influences` в моделі музиканта. Для неавторизованого користувача параметр `influences` запиту визначає перелік музикантів або колективів, які вплинули на музичний колектив. Цей параметр є рядком, але він перетворюється на перелік шляхом розділення рядка за входженням коми. Далі за кожним виділеним колективом або музикантом відбувається перевірка на основі формування загальної умови. Якщо в параметрі `influenceadd` встановлено істинне значення, то перевіряється, щоб всі вказані колективи (музиканти) входили до переліку колективів (музикантів), які вплинули на музичний колектив. Якщо вказане хибне значення, то перевіряється, щоб колектив грав музику, на яку вплинув хоча би один з перелічених колективів.

Параметр `instruments` визначає перелік інструментів, на яких грають музиканти у музичному колективі: параметр є рядком, але він перетворюється на перелік шляхом розділення рядка за входженням коми, далі за кожним виділеним інструментом відбувається перевірка на основі формування загальної умови. Якщо в параметрі `instrumentadd` встановлено істинне значення, то перевіряється, щоб всі вказані інструменти входили до переліку інструментів, на яких грає музичний колектив. Якщо вказане хибне

значення, то перевіряється, щоб колектив грав музику хочаби на одному з вказаних інструментів.

Параметр `positions` визначає перелік позицій у музичному колективі, які мають бути представлені на бажання музиканта: параметр є рядком, перетворюється на перелік шляхом розділення рядка за входженням коми, далі за кожною виділеною позицією відбувається перевірка на основі формування загальної умови. Якщо в параметрі `positionadd` встановлено істинне значення, то перевіряється, щоб всі вказані позиції входили до переліку позицій, які представлені в музичному колективі. Якщо вказане хибне значення, то перевіряється, щоб хочаби одна з позицій була представлена в музичному колективі.

Параметри `level1` і `level2` дозволяють музиканту за необхідності обмежити потрібний загальний рівень музикування музикантів у колективі. У такому випадку всі музиканти в колективі повинні знаходитися в заданому інтервалі. Так само за необхідності на основі параметрів запити `theory1` і `theory2` обмежується рівень володіння музичною теорією музикантів. Рівень вмінь стосовно запису музики обмежується інтервалом, встановленим через значення параметрів `record1` і `record2` запити.

Усі оголошення про постійну участь у музичному колективі, які відповідають встановленій множині параметрів (якщо якісь не встановлені, то вони не перевіряються), впорядковуються за часом створення, відображаючи спочатку нові оголошення. Окрім того для авторизованих музикантів перевіряються всі нотатки, які були створені за даними оголошеннями. Вони повертаються як словник `notes` шляхом передавання їх у шаблон вебсторінки.

Пошук музичного проєкту реалізований на основі функції відображення `seek_for_project_view` та шаблону вебсторінки `seek_project.html`. Функція відображення виконує наповнення даними шаблону через звернення до моделі `ProjectAdvert`. Вона також працює в 2 режимах, що забезпечують наповнення даними сторінки для авторизованого

музиканта і для неавторизованого. Пошук для неавторизованого користувача визначає всі параметри через запит до вебсторінки, витягаючи усі значення параметрів з запиту, а пошук для авторизованого музиканта визначає більшу частину з того оголошення, яке створив раніше музикант.

Якщо користувач є авторизованим, то місто, в якому знаходиться музичний колектив, який створює цей музичний проєкт, витягається з міста, яке визначено фактично в профілі музиканта на основі моделі Musician. Якщо користувач неавторизований, то місто береться з параметра city. Після встановлення міста зі всіх об'єктів БД з набору моделі ProjectAdvert відфільтровуються ті оголошення, в яких музичний колектив знаходиться в цьому місті.

Наступним перевіряється інструмент, на якому здатен грати музикант. Ця інформація відбирається з моделі CapabilityInstrument. Усі оголошення, які містять такі інструменти, залишаються в результатах. Якщо користувач не авторизувався, то потрібна позиція відбирається з параметру musinstruments запиту: параметр є рядком, але він перетворюється на перелік шляхом розділення рядка за входженням коми, далі за кожним виділеним інструментом відбувається перевірка на основі формування загальної умови. Усі оголошення щодо участі в музичних проєктах, в яких визначений один з цих інструментів, залишаються в результатах. У подальшому за кожним параметром виконується додавання фільтрів, а безпосередній відбір виконується вже на завершення після встановлення всіх фільтрів.

Для відфільтровування оголошень за досвідом відбираються ті оголошення, де досвід менше або дорівнює встановленому. Встановлюється досвід або через параметр experience для неавторизованого користувача, або через врахування досвіду, вказаного в полі start_year для музиканта, що є роком, з якого займається музикою цей музикант. Для цього з поточної дати віднімається ця кількість років і отримане значення враховується при відфільтровуванні вказаним вище чином.

Параметр `level` запиту або поле `overall_level` з профілю авторизованого музиканта дозволяють відібрати всі оголошення, в яких загальний рівень музикування встановлено на рівні, що нижче або дорівнює рівню музикування авторизованого музиканта.

Параметр `theory` запиту або поле `music_theory` з профілю авторизованого музиканта дозволяють відібрати всі оголошення, в яких рівень знання музичної теорії музикантом встановлено на рівні, що нижче або дорівнює рівню знання музичної теорії авторизованого музиканта.

Параметр `rec` запиту або поле `recording_level` з профілю авторизованого музиканта дозволяють відібрати всі оголошення, в яких рівень вмінь музиканта стосовно запису музики встановлено на рівні, що нижче або дорівнює рівню вмінь стосовно запису музики авторизованого музиканта.

З параметра `search` запиту вибирається текст, який перевіряється в назві музичного проєкту. Якщо в назву він входить, то таке оголошення залишається в результатах пошуку. Одночасно з параметра `forbid` розбирається перелік заборонених слів. Значення параметра розділяється пробілами на окремі слова. Якщо хочаби одне слово вдалось виділити, то всі оголошення, в назві яких знаходиться хоча би одне з виділених слів, відкидаються з результатів пошуку.

Параметри `gigs1` і `gigs2` визначають мінімальну і максимальну орієнтовну кількість концертів музичного колективу, який створює цей музичний проєкт. Відбираються оголошення тих музичних колективів, кількість концертів яких не перевищує максимальну межу і не менша за мінімальну кількість.

Параметри `years1` і `years2` визначають мінімальну і максимальну кількість років, протягом яких існує музичний колектив, який створює цей музичний проєкт. Відбираються оголошення тих музичних колективів, рік створення яких не менше року, обчисленого шляхом віднімання від поточного року максимальної кількості років (`years2`), і не більше за рік,

обчислений шляхом віднімання від поточного року мінімальної кількості років (years1).

Параметр `genres` визначає перелік жанрів, у яких має створюватися музичний проєкт: параметр є рядком, перетворюється на перелік шляхом розділення рядка за входженням коми, далі за кожним виділеним жанром відбувається перевірка на основі формування загальної умови. Якщо в параметрі `genreadd` встановлено істинне значення, то перевіряється, щоб всі вказані жанри входили до переліку жанрів музичного проєкту. Якщо вказане хибне значення, то перевіряється, щоб музичний проєкт стосувався хоча би одного з вказаних жанрів.

Якщо музикант авторизований, то використовується перелік музикантів або колективів, які вплинули на нього, для пошуку музичного проєкту, на колектив якого вплинули аналогічні музичні колективи. Для неавторизованого користувача параметр `influences` запиту визначає перелік музикантів або колективів, які вплинули на музичний колектив, що створює проєкт. Цей параметр є рядком, але він перетворюється на перелік шляхом розділення рядка за входженням коми. Далі за кожним виділеним колективом або музикантом відбувається перевірка на основі формування загальної умови. Якщо в параметрі `influenceadd` встановлено істинне значення, то перевіряється, щоб всі вказані колективи (музиканти) входили до переліку колективів (музикантів), які вплинули на музичний колектив, що створює проєкт. Якщо вказане хибне значення, то перевіряється, щоб колектив, що створює проєкт, грав музику, на яку вплинув хоча би один з перелічених колективів.

Параметр `instruments` визначає перелік інструментів, на яких грають музиканти у музичному проєкті, тобто він необхідний, щоб музикант міг обмежити вже фактично стиль для самого проєкту. Тому перевірка відбувається стосовно саме оголошень для проєкту. Параметр є рядком, але він перетворюється на перелік шляхом розділення рядка за входженням коми, далі за кожним виділеним інструментом відбувається перевірка на

основі формування загальної умови. Якщо в параметрі `instrumentadd` встановлено істинне значення, то перевіряється, щоб всі вказані інструменти входили до переліку інструментів, на які виконується пошук музикантів для музичного проєкту. Якщо вказане хибне значення, то перевіряється, щоб було оголошення стосовно хочаби одного з вказаних інструментів у цьому музичному проєкті.

Параметр `positions` визначає перелік позицій у музичному колективі, які мають бути представлені на бажання музиканта: параметр є рядком, перетворюється на перелік шляхом розділення рядка за входженням коми, далі за кожною виділеною позицією відбувається перевірка на основі формування загальної умови. Якщо в параметрі `positionadd` встановлено істинне значення, то перевіряється, щоб всі вказані позиції входили до переліку позицій, які представлені в музичному колективі, який створює цей музичний проєкт. Якщо вказане хибне значення, то перевіряється, щоб хочаби одна з позицій була представлена в музичному колективі, що створює цей проєкт.

Параметри `level1` і `level2` дозволяють музиканту за необхідності обмежити потрібний загальний рівень музикування музикантів у колективі, що створює проєкт. У такому випадку всі музиканти в колективі повинні знаходитися в заданому інтервалі. Так само за необхідності на основі параметрів запити `theory1` і `theory2` обмежується рівень володіння музичною теорією музикантів, що створюють проєкт. Рівень вмінь стосовно запису музики обмежується інтервалом, встановленим через значення параметрів `record1` і `record2` запити.

Для відфільтровування оголошень за вимогами до віку музиканта відбираються ті оголошення, де нижня вікова межа в полі `age_from` менша за вік музиканта, який обчислюється через знаходження кількості років між поточною датою і датою народження музиканта, або дорівнює йому, а нижня вікова межа `age_to` більша або дорівнює віку музиканта. Для

неавторизованого музиканта поточний вік не обчислюється, а встановлюється на основі значення параметра age запиту.

Останнім перевіряється відповідність досвіду музиканта за грою на інструменті вимогам щодо досвіду за цим інструментом з оголошення.

Усі оголошення про участь у музичному проєкті, які відповідають встановленій множині параметрів, впорядковуються за часом створення, відображаючи спочатку нові оголошення. Окрім того для авторизованих музикантів перевіряються всі нотатки, які були створені за даними оголошеннями. Вони повертаються як словник notes шляхом передавання їх у шаблон вебсторінки.

3.3 Висновки за розділом 3

Розділ 3 ПЗ містить опис реалізації процесу пошуку оголошень щодо постійної участі в музичному колективі й щодо участі в музичному проєкті та порядок загальної роботи з програмою.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Опис програми

Програма охоплює і індивідуальних музикантів, і музичні колективи, і музикантів, які є учасниками колективів. Програма використовує для них загальну ідентифікацію щодо функцій, тобто на рівні функцій всі вони мають представлені однаково з забезпеченням необхідного рівня доступу, тобто наприклад, щодо поданих заявок виконувати керування може тільки музикант, для якого в колективі надано можливість керування цими даними. Окремо тільки виділений неавторизований користувач, для якого надаються тільки базові функції пошуку музикантів, колективів та проєктів.

Для загальної групи авторизованих музикантів програма призначена і для пошуку музичних колективів, до яких музикант може доєднатися, і для пошуку музикантів до музичного колективу.

4.2 Експлуатація програми

Для експлуатації програми має бути забезпечено запуск вебсервера, щоб всі клієнти могли звернутися за виділеною адресою і отримати доступ до вебзастосунку. Перед цим має бути всі необхідні підготовчі дії, зокрема за допомогою команд `makemigrations` і `migrate` виконано створення БД, що підтримує роботу вебзастосунку, попередньо запустивши сервер БД MySQL і встановивши відповідні налаштування в файлі `settings.py` проєкту [14].

При першому зверненні доступ забезпечується як для неавторизованого користувача, тому стандартне меню обмежене тільки розділами “Пошук”, “Міста” і “Увійти”. Останній розділ дозволяє перейти до стану авторизованого музиканта або спочатку виконати внесення власних даних, фактично зареєструвавшись. Розділ “Міста” дозволяє переглянути перелік музичних колективів і музикантів за обраним містом. Спочатку

відображаються всі країни, потім за обраною країною міста, а потім за обраним містом – музичні колективи і музиканти.

Розділ “Пошук” відображає спочатку три перемикача між різними варіантами пошуку: “Музиканти”, “Проекти”, “Колективи” для пошуку відповідно оголошень музикантів, оголошень про участь в музичному проєкті, оголошень про постійну участь в музичному колективі. Після вибору перемикача потрібно задати параметри пошуку (детально описані в розділі 3), а після натискання на кнопку “Шукати” відображається окрема сторінка з результатами пошуку (рис. 4.1).

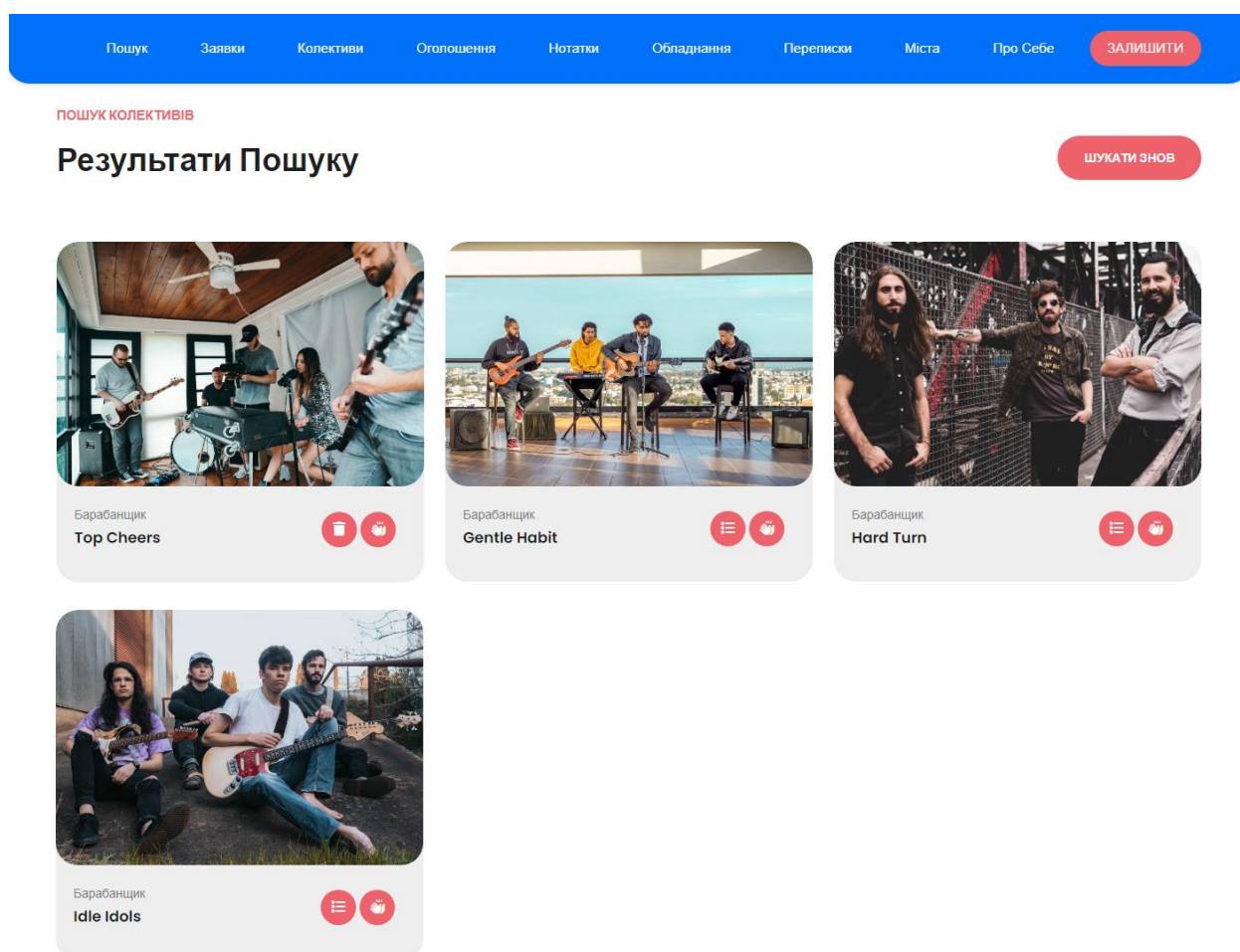


Рисунок 4.1 – Результати пошуку оголошень про постійну участь у музичному колективі

Кожне оголошення містить позицію і назву колективу (для пошуку оголошень про постійну участь у музичному колективі), інструмент і назву

проєкту (для пошуку оголошень про участь у музичному проєкті), ім'я музиканта (для пошуку оголошень музикантів). Натискаючи на фотографію або на підпис знизу, можна перейти на сторінку з повними даними оголошення. На сторінці з результатами міститься кнопка для подачі заявки до оголошення, розташована справа. Лівіше розташована кнопка для створення нотатки за оголошенням, щоб потім мати можливість усі такі оголошення переглянути з розділу “Нотатки”. Ця кнопка має різний вигляд: корзина для оголошення, яке вже було збережено, та перелік для оголошення, яке ще не було збережено.

З розділу “Міста”, також з результатів пошуку, де може відображатися назва колективу, або зі сторінки оголошення, де назва колективу точно відображається, можна перейти до сторінки музичного колективу, а зі сторінки колективу перейти до сторінки музичного проєкту (рис. 4.2). Сторінка музичного проєкту містить фотографію, назву колективу та нижче назву проєкту. Окрім того розташовується опис проєкту.

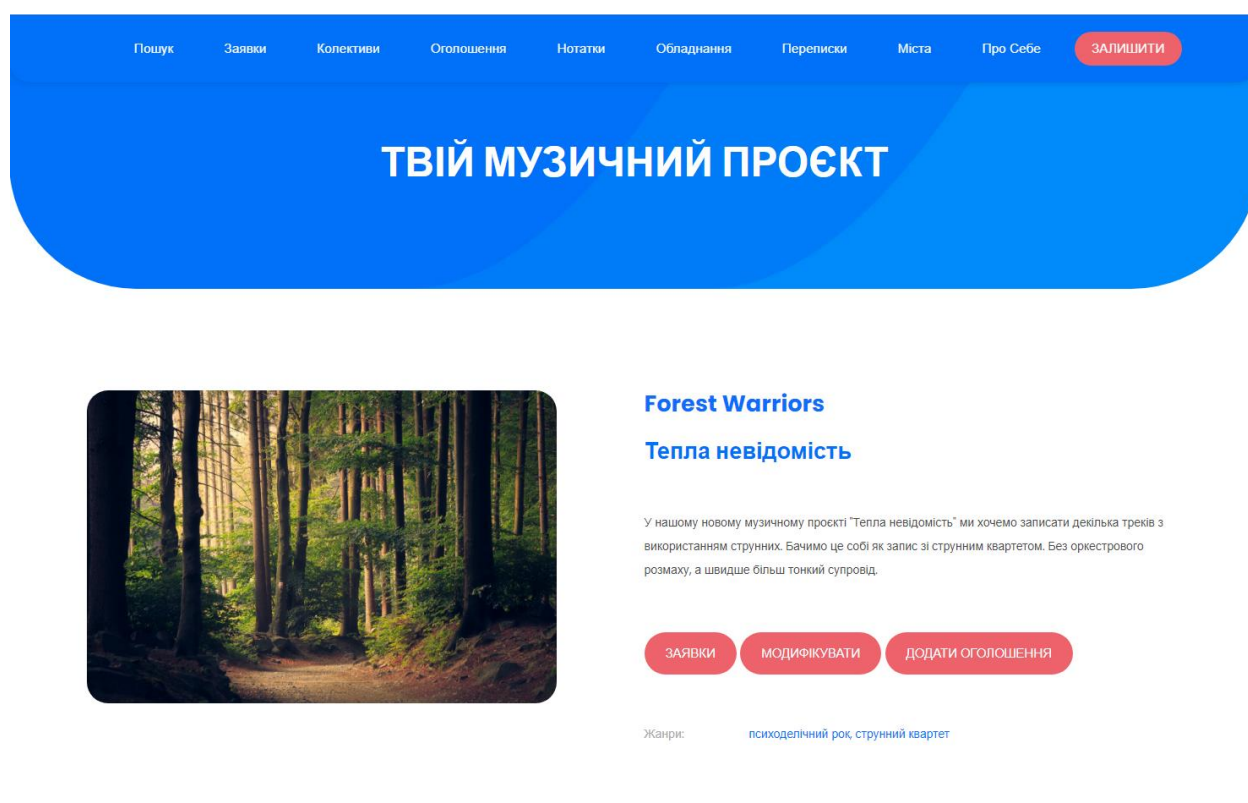


Рисунок 4.2 – Музичний проєкт у програмі

Кнопка “Заявки” дозволяє переглянути перелік заявок, поданих за проєктом. Вона відображається тільки для авторизованого музиканта, який має доступ до можливостей керування в музичному колективі, в якому він є учасником. Для музиканта, який не має таких можливостей, кнопка не відображається. Так само відображається або не відображається кнопка “Модифікувати” для внесення змін в дані проєкту, якщо це необхідно. Кнопка “Додати оголошення” дозволяє створити нове оголошення стосовно участі в музичному проєкті.

У нижній частині сторінки відображаються оголошення за музичним проєктом (рис. 4.3). Кожне оголошення містить назву інструмента, загальний рівень музиканта для участі в проєкті, рівень знань теорії, вміння запису музики. Кнопка “Модифікувати” дозволяє перейти до модифікації оголошення. Вона також відображається тільки для керуючого музиканта. При натисканні на назву інструменту або на фотографію відкривається сторінка з оголошенням, де знаходиться повна інформація.

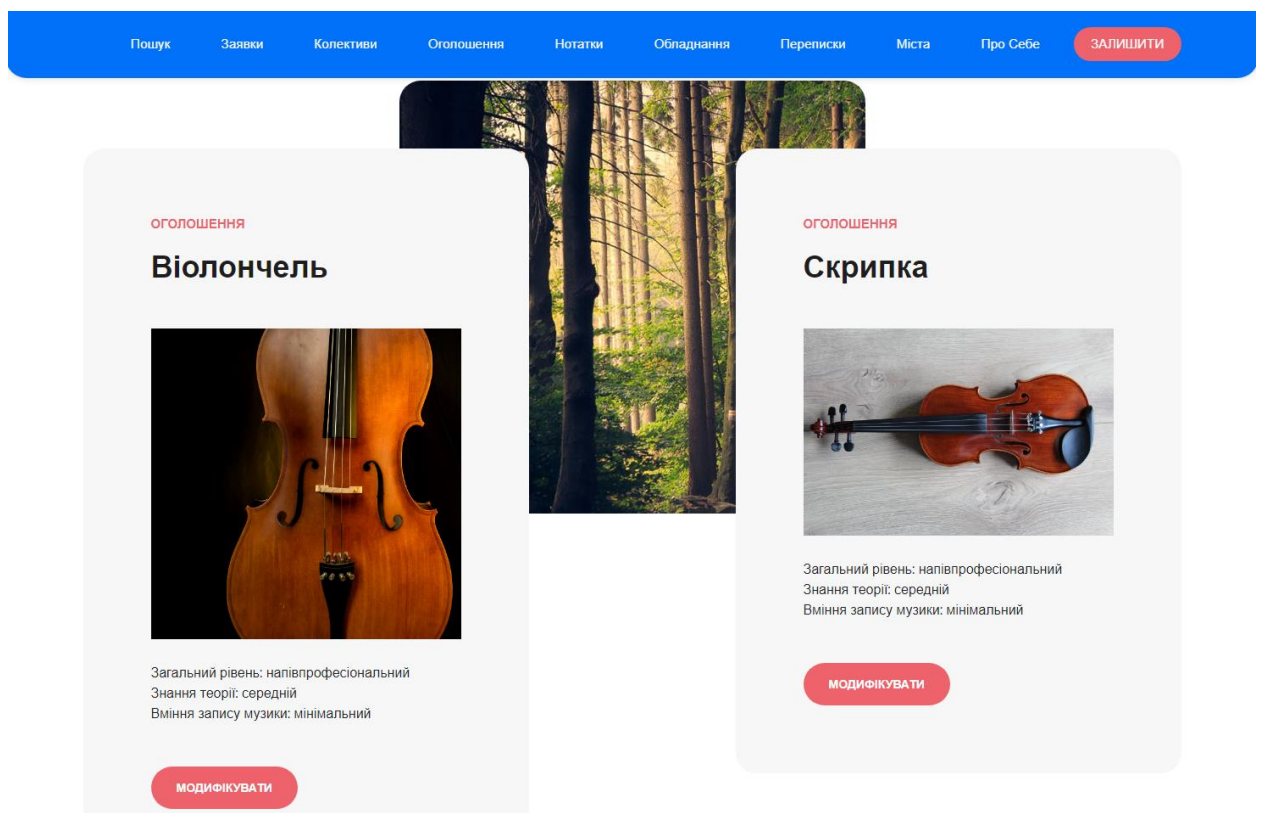


Рисунок 4.3 – Оголошення за музичним проєктом

Для музиканта, який не має відношення до музичного колективу, замість кнопки “Модифікувати” відображається кнопка “Подати заявку” та кнопка “Зберегти” для створення нотатки.

Розділ меню “Заявки” дозволяє переглянути всі заявки, подані авторизованим музикантом (рис. 4.4). Заявки з іншого боку, зі сторони автора оголошення, відображаються за кнопкою на сторінці оголошення (як наприклад, раніше було представлено на сторінці проєкту, або аналогічно на сторінці музичного колективу).

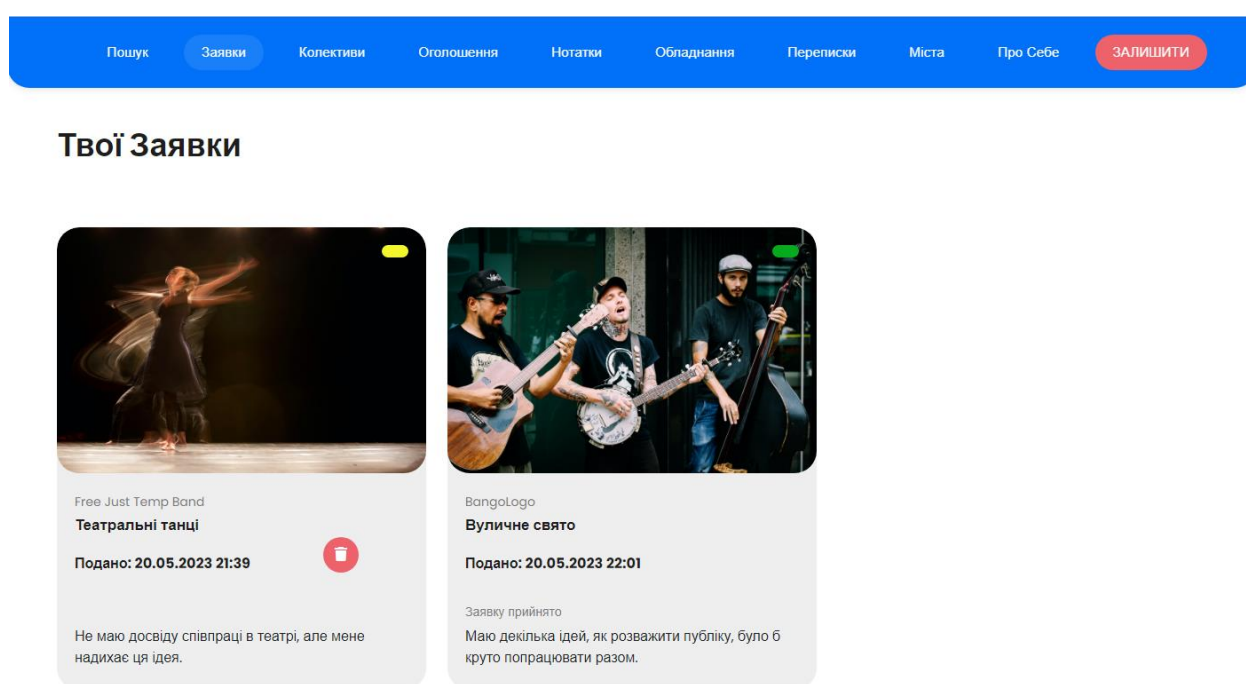


Рисунок 4.4 – Заявки музиканта в програмі

За власною заявкою відображається загальна інформація, опис від автора заявки, час подачі заявки та кнопка з зображенням корзини, за допомогою якої можна скасувати власну заявку. На фотографії у правому куті кольором позначається фактично стан заявки: жовтий – якщо заявка подана і не розглянута, червоний – якщо заявка скасована або відхилена, а зелений – якщо заявка підтримана автором (прийнята).

Розділ меню “Колективи” містить перелік колективів, даними якого може керувати авторизований музикант. Від кожного колективу можна

перейти до його сторінки, а звідки до сторінки музичного проєкту. Розділ “Оголошення” містить оголошення самого музиканта, а оголошення стосовно колективів для музиканта, який керує його даними, відображаються відповідно на сторінках колективу та музичного проєкту.

Розділ “Переписки” містить перелік розмов, які були створені цим музикантом. Перейти до створення розмови або додавання нового повідомлення, тобто фактично відкриття сторінки з повідомленнями, можна зі сторінки оголошення. На кожній такій сторінці відображається відповідна кнопка для написання повідомлення.

Розділ “Обладнання” дозволяє внести всі інструменти, на яких грає атворизований музикант. Ці інструменти потім відображаються на сторінці музичного колективу.

Розділ “Про себе” дозволяє відобразити власні дані музиканта та оновити їх за потреби.

4.3 Тестування програми

Тестування програми реалізовано у чіткій відповідності зі сценаріями, які були представлені на відповідних діаграмах варіантів використання в розділі 2. Там, де сценарії потребували певної деталізації, тобто фактично передбачали реалізацію деяких складових, вони були повністю перевірені, а позначка про результат встановлювалась тільки щодо повної відповідності або невідповідності сукупному результату. Якщо хоч якась невідповідність була в підсумковому тестуванні знайдена, то мала проставлятися позначка про відхилення від очікуваного результату. У табл. 4.1 представлені зведені підсумкові результати, за якими вдалося досягти успішного виконання всіх функцій, передбачених для реалізації.

Таблиця 4.1 – Результати тестування програми

Функція	Результат
1	2
Пошук музичного колективу	Успіх
Пошук музичного проєкту	Успіх
Пошук музиканта	Успіх
Перегляд музичних проєктів музичного колективу	Успіх
Перегляд оголошень за музичним колективом	Успіх
Перегляд оголошень за музикантом	Успіх
Перегляд музикантів і музичних колективів у місті	Успіх
Ознайомлення з інформацією про музичний колектив	Успіх
Ознайомлення з інформацією про музиканта	Успіх
Ознайомлення з оголошенням про пошук постійного учасника колективу	Успіх
Ознайомлення з оголошенням про пошук учасника музичного колективу	Успіх
Ознайомлення з оголошенням музиканта	Успіх
Відтворення музики	Успіх
Додавання власного профілю музиканта	Успіх
Авторизація облікового запису музиканта	Успіх
Додавання власного музичного колективу	Успіх
Модифікація даних власного музичного колективу	Успіх
Додавання учасників музичного колективу	Успіх
Вилучення учасників музичного колективу	Успіх
Визначення позиції учасника в музичному колективі	Успіх
Публікація оголошення про пошук музиканта для постійної участі в музичному колективі	Успіх
Модифікація оголошення про пошук музиканта для постійної участі в музичному колективі	Успіх

Продовження таблиці 4.1

1	2
Додавання нового музичного проєкту колективу	Успіх
Модифікація даних музичного проєкту	Успіх
Визначення керуючих учасників музичного колективу	Успіх
Публікація оголошення про пошук музиканта для участі в музичному проєкті	Успіх
Модифікація оголошення про пошук музиканта для участі в музичному проєкті	Успіх
Публікація оголошення музикантом про пошук музичного колективу	Успіх
Модифікація оголошення музикантом про пошук музичного колективу	Успіх
Додавання даних музичного обладнання музикантом	Успіх
Керування даними музичного обладнання музикантом	Успіх
Відправлення короткого запиту і відповіді між музичним гуртом і музикантом	Успіх
Перегляд переписки між музичним гуртом і музикантом	Успіх
Перегляд переліку переписок між гуртом і музикантом	Успіх
Додавання музики музичного колективу	Успіх
Керування переліком музики музичного колективу	Успіх
Додавання нотатки до оголошення музикантом	Успіх
Перегляд занотованих оголошень музикантом	Успіх
Перегляд занотованих оголошень усіма учасниками музичного колективу	Успіх
Керування нотаткою до оголошення	Успіх
Оновлення власного профілю музиканта	Успіх
Подання заявки на участь у музичному проєкті	Успіх
Подання заявки на постійну участь у музичному колективу	Успіх

Кінець таблиці 4.1

1	2
Подання заявки на співпрацю з музикантом у музичному колективі	Успіх
Подання заявки на співпрацю з музикантом у музичному проєкті	Успіх
Перегляд заявок за музичним проєктом	Успіх
Перегляд заявок за оголошенням за музичним проєктом	Успіх
Перегляд заявок за музичним колективом	Успіх
Перегляд заявок за оголошенням за музичним колективом	Успіх
Перегляд заявок за оголошенням музиканта	Успіх
Перегляд музичних колективів, оголошеннями яких може керувати музикант	Успіх
Перегляд власних поданих заявок	Успіх
Керування поданою заявкою	Успіх
Скасування власної заявки	Успіх

4.4 Висновки за розділом 4

Розділ 4 ПЗ містить опис роботи користувача з програмою щодо виконання основних функцій, які необхідні користувачам у програмі, результати тестування щодо виконання цих функцій.

ВИСНОВКИ

Розробка програмного забезпечення пошуку учасників музичного колективу для спрощення взаємодії між музикантами і музичними колективами була визначена метою дипломної кваліфікаційної роботи і була разом з завданнями досягнута в роботі.

У ПЗ представлено основні результати, досягнуті під час виконання завдань:

- розділ 1 містить результати аналізу існуючих програм пошуку учасників музичного колективу, включаючи до нього Hendrix, Bandmix, BandFinder, Join-A-Band: за результатами аналізу визначені відмінності розроблюваного програмного забезпечення, що базуються в основному на можливості роботи щодо пошуку музикантів для участі в окремих проєктах музичного колективу, а також можливості поширення можливостей пошуку на користувачів з України;

- розділ 2 містить результати проєктування програми: вибір мови Python, системи керування БД MySQL, фреймворку Django, моделювання роботи з програмою на основі вимог, визначення структури БД;

- розділ 3 містить опис реалізації процесу пошуку оголошень щодо постійної участі в музичному колективі й щодо участі в музичному проєкті та порядок загальної роботи з програмою;

- розділ 4 містить опис роботи користувача з програмою щодо виконання основних функцій, які необхідні користувачам у програмі, результати тестування щодо виконання цих функцій.

Програмне забезпечення надає можливості пошуку не тільки музикантів для участі в колективі на певних позиціях і навпаки, але і пошуку музикантів саме під окремі музичні проєкти для реалізації тимчасових ідей колективів, надаючи для цього засоби створення нотаток з оголошень, публікації оголошень, подачі заявок, керування заявками і оголошеннями.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Keyes, D. 10 Ways To Find Band Members & Musicians [Electronic resource] / D. Keyes. – Access mode : <https://www.dk-mba.com/blog/10-ways-to-find-band-members-musicians>.
2. Proven Tactics to Find New Band Members [Electronic resource]. – Access mode : <https://pirate.com/en/blog/rehearsal-tips/how-to-find-band-members/>.
3. Hendrix – Meet Musicians. Make Music [Electronic resource]. – Access mode : <https://www.gohendrix.com/>.
4. Musicians Wanted – Search Local Musician Classifieds [Electronic resource]. – Access mode : <https://www.bandmix.com/>.
5. Get Your Jam On [Electronic resource]. – Access mode : <https://musicianfinder.com/>.
6. Musicians Wanted Sweden – Musicians Classifieds – Find Local Musicians Sweden. Bandfinder [Electronic resource]. – Access mode : <https://www.bandfinder.com/>.
7. Join A Band – Find band members online [Electronic resource]. – Access mode : <https://www.join-a-band.com/>.
8. Mahmood, S. H. The 8 Best Websites to Find Band Members and Bands [Electronic resource] / S. H. Mahmood. – Access mode : <https://www.makeuseof.com/websites-to-find-band-members-and-bands/>.
9. Musicians Wanted [Electronic resource] : A guide to finding bandmates. – Access mode : <https://mixdownmag.com.au/features/musicians-wanted-a-guide-to-finding-bandmates/>.
10. Davidson, M. 11 ways to find band members [Electronic resource] / M. Davidson. – Access mode : <https://blog.discmakers.com/2017/03/11-ways-to-find-band-members/>.
11. Mullen, M. 10 ways to find band members and musicians [Electronic resource] / M. Mullen. – Access mode : <https://www.musicradar.com/news/10->

ways-to-find-band-members-and-musicians.

12. Lim, G. Django 4 for the Impatient [Text] : Learn the core concepts of Python web development with Django in one weekend / Greg Lim, Daniel Correa. – Birmingham : Packt Publishing, 2022. – 190 p.

13. Lugh Gaming Template [Electronic resource]. – Access mode : <https://templatemo.com/tm-589-lugh-gaming>.

14. Django Settings. Django Documentation. Django [Electronic resource]. – Access mode : <https://docs.djangoproject.com/en/4.2/topics/settings/>.

ДОДАТОК А
Технічне завдання

Вступ

Пошук однодумців завжди є певною проблемою, яку будь-яка особа, що має власні ідеї для реалізації, намагається певним чином вирішити. Коли ці ідеї стосуються музики, дуже часто потрібно знайти для них деякий колектив людей. Колектив може створюватися, змінювати учасників з часом, розширювати або зменшувати склад учасників загалом. Усе це потребує пошуку музикантів або музичних колективів. Не завжди знайомі, персональні контакти цю проблему здатні вирішити. Саме для таких ситуацій, а також для ситуацій, коли потрібно розширити коло пошуку, має використовуватися програмне забезпечення, що повинно бути створено в роботі.

A.1 Підстави для розробки

Завдання на дипломну кваліфікаційну роботу сформульовано у відповідності з наказом № 87 від 4 квітня 2023 р. за Національним університетом «Запорізька політехніка», за яким було визначено тему «Програмне забезпечення пошуку учасників музичного колективу».

A.2 Призначення розробки

Програма повинна охоплювати і індивідуальних музикантів, і музичні колективи, і музикантів, які є учасниками колективів. Вона повинна використовувати для них загальну ідентифікацію щодо функцій, тобто на рівні функцій всі вони мають бути представлені однаково з забезпеченням необхідного рівня доступу. Окремо має бути виділений неавторизований користувач тільки. Для нього програма має бути призначеною для пошуку оголошень. Для загальної групи авторизованих музикантів програма призначена і для пошуку музичних колективів, до яких музикант може доєднатися, і для пошуку музикантів до музичного колективу.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Забезпечення призначення програми виконується через реалізацію сценаріїв використання програми на основі наступних функцій:

- пошук музичного колективу музикантом;
- пошук музичного проєкту музикантом;
- пошук музиканта для участі в музичному колективі або проєкті;
- додавання власного музичного колективу;
- модифікація даних власного музичного колективу;
- додавання і вилучення учасників музичного колективу;
- визначення позиції учасника в музичному колективі;
- публікація і модифікація оголошення про пошук музиканта для постійної участі в музичному колективі;
- додавання нового музичного проєкту колективу;
- модифікація даних музичного проєкту;
- перегляд музичних проєктів музичного колективу;
- визначення керуючих учасників музичного колективу;
- публікація і модифікація оголошення про пошук музиканта для участі в музичному проєкті;
- публікація і модифікація оголошення музикантом про пошук музичного колективу;
- перегляд оголошень за музичним колективом;
- перегляд оголошень за музикантом;
- деактивація оголошення;
- додавання, модифікація і вилучення даних музичного обладнання музикантом;
- перегляд музикантів і музичних колективів у місті;
- ознайомлення з інформацією про музичний колектив;
- ознайомлення з інформацією про музиканта;

- ознайомлення з оголошенням;
- відправлення короткого запиту і відповіді між музичним гуртом і музикантом;
- перегляд переписки між музичним гуртом і музикантом;
- перегляд переліку переписок між гуртом і музикантом;
- відтворення музики;
- додавання музики музичного колективу;
- керування переліком музики музичного колективу;
- додавання нотатки до оголошення музикантом;
- перегляд занотованих оголошень музикантом;
- перегляд занотованих оголошень усіма учасниками музичного колективу;
- подання заявки на участь у музичному проєкті;
- подання заявки на постійну участь у музичному колективі;
- подання заявки на співправцю з музикантом;
- перегляд заявок за музичним проєктом;
- перегляд заявок за оголошенням за музичним проєктом;
- перегляд заявок за музичним колективом;
- перегляд заявок за оголошенням за музичним колективом;
- перегляд заявок за оголошенням музиканта;
- перегляд музичних колективів, оголошеннями яких може керувати музикант;
- перегляд власних поданих заявок;
- керування поданою заявкою;
- скасування власної заявки;
- модифікація нотатки до оголошення музикантом;
- вилучення нотатки до оголошення музикантом;
- додавання і модифікація власного профілю музиканта;
- авторизація облікового запису музиканта.

A.3.2 Умови експлуатації

Експлуатація програми на сервері потребує підтримки, що включає навички роботи з БД, зокрема MySQL, виконання міграцій для Django-додатків та внесення даних через адміністративний інтерфейс при необхідності розширення або зміни жанрів, інструментів, позицій або інших каталогів стандартних даних.

A.3.3 Вимоги до складу та параметрів технічних засобів

Набір мінімальних технічних засобів, що мають забезпечуватися на сервері для підтримки роботи вебдодатку, включає:

- процесор не менше ніж Xeon сімейства 3000 або Core 2 Duo;
- оперативна пам'ять обсягом у 4 Гб;
- простір на жорсткому диску з доступним вільним обсягом не менше ніж 3 Гб, але для можливості збереження файлів зображень бажано не менше 20 Гб;
- мережна карта і підключення до мереж Інтернет.

A.4 Порядок контролю та приймання

Для виконання дипломної кваліфікаційної роботи критичним є виконання визначеного завдання, що включає безпосередньо завдання на роботу і технічне завдання, за темою, яка зафіксована наказом, у строки, які визначені завданням на роботу.

У підсумку виконується захист роботи за оприлюдненим наперед графіком перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

Б.1 Текст файла `userviews.py`

```

from django.db.models import Q
from .models import MemberAdvert
from .models import Band
from .models import City
from .models import Genre
from .models import Influence
from .models import Instrument
from .models import Position
from .models import Capability
from .models import MusicianAdvert
from .models import ProjectAdvert
from .models import CapabilityInstrument
from .models import MemberAdvert
from .models import ProjectAdvertNote
from .models import MusicianAdvertNote
from .models import Equipment
from datetime import datetime

def seek_for_band_view(request):
    now_time=datetime.now()
    if request.user.is_active:

musician=Musician.objects.get(django_user=request.user)
        musadvert=
MusicianAdvert.objects.get(musician__django_user=request.user)
        band_city=musician.location_city
    else:
        band_city=self.request.GET.get('city', None)
        if band_city:
            band_city=City.objects.get(title=band_city, None)
        if band_city:

adverts=MemberAdvert.objects.filter(band__location_city=band_city)
    else:
        return render(request,
"musicianseeker/seek_band.html", {"adverts": []})
        adverts=adverts.filter(active=True)
        if request.user.is_active:

musician_positions=Capability.objects.filter(musician_advert=musadv
ert, active=True).distinct('position')
        condition=None
        for musician_position in musician_positions:
            if not condition:
                condition=Q(position=musician_position)
            else:

```

```

        condition=condition |
Q(position=musician_position)
    adverts=adverts.filter(condition)
else:
    position=self.request.GET.get('position', None)
    if position:
        adverts=adverts.filter(position=position)
    if request.user.is_active:
        practice=musadvert.practice_availability
    else:
        practice=self.request.GET.get('practice', None)
    if practice:

adverts=adverts.filter(practice_availability__lte=int(practice))
    if request.user.is_active:
        gig=musadvert.gig_availability
    else:
        gig=self.request.GET.get('gig', None)
    if gig:

adverts=adverts.filter(gig_availability__lte=int(gig))
    if request.user.is_active:
        start_year=musician.start_year
        experience=now_time.year-start_year
    else:
        experience=self.request.GET.get('experience', None)
    if experience:

adverts=adverts.filter(experience__lte=int(experience))
    if request.user.is_active:
        overall_level=musician.overall_level
    else:
        overall_level=self.request.GET.get('level', None)
    if overall_level:

adverts=adverts.filter(overall_level__lte=int(overall_level))
    if request.user.is_active:
        diff=musician.date_of_birth - now_time
        age=diff.years
    else:
        age=self.request.GET.get('age', None)
    if age:
        adverts=adverts.filter(age_from__lte=age,
age_to__gte=age)
    if request.user.is_active:
        music_theory=musician.music_theory
    else:
        music_theory=self.request.GET.get('theory', None)
    if music_theory:

```

```

adverts=adverts.filter(music_theory__lte=int(music_theory))
    if request.user.is_active:
        recording_level=musician.recording_level
    else:
        recording_level=self.request.GET.get('rec', None)
    if recording_level:

adverts=adverts.filter(recording_level__lte=int(recording_level))
    seek_text=self.request.GET.get('search', '')
    if seek_text:

adverts=adverts.filter(Q(band__title__icontains=seek_text) |
Q(band__description__icontains=seek_text))
        forbidden_text=self.request.GET.get('forbid', '')
        forbidden_text_list=forbidden_text.split(' ')
        if forbidden_text_list:
            for forbid in forbidden_text_list:

adverts=adverts.exclude(Q(band__title__icontains=forbid) |
Q(band__description__icontains=forbid))
        band_gigs=self.request.GET.get('gigs1', 0)
        if band_gigs:
            adverts=adverts.filter(band__gigs__gte =
int(band_gigs))
        band_gigs_max=self.request.GET.get('gigs2', 0)
        if band_gigs_max:
            adverts=adverts.filter(band__gigs__lte =
int(band_gigs_max))
        band_experience=self.request.GET.get('years1', 0)
        band_experience_end=now_time.year - int(band_experience)
        band_experience_max=self.request.GET.get('years2', 0)
        band_experience_start=now_time.year -
int(band_experience_max)
        if band_experience_start > band_experience_end:

adverts=adverts.filter(band__start_year__gte=band_experience_start,
band__start_year__lte=band_experience_end)
        elif band_experience and band_experience_max==0:

adverts=adverts.filter(band__start_year__lte=band_experience_end)
        elif band_experience_max and band_experience==0:

adverts=adverts.filter(band__start_year__gte=band_experience_start)
        genres=self.request.GET.get('genres', None)
        genres_list=genres.split(',')
        genres_condition=None
        genre_unite=int(self.request.GET.get('genreadd', 0))
        for genre in genres_list:
            genre=Genre.objects.get(title=genre)

```

```

        if not genres_condition:
            genres_condition=Q(band__genres=genre)
        elif genre_unite:
            genres_condition=genres_condition &
Q(band__genres=genre)
        else:
            genres_condition=genres_condition |
Q(band__genres=genre)
        if genres_condition:
            adverts=adverts.filter(genres_condition)
        if not request.user.is_active:
            influences=self.request.GET.get('influences', None)
            influences_list=influences.split(',')
            influence_condition = None

influence_unite=int(self.request.GET.get('influenceadd', 0))
        for influence in influences_list:
            influence=Influence.objects.get(title=influence)
            if not influence_condition:

influence_condition=Q(band__influences=influence)
            elif influence_unite:
                influence_condition=influence_condition &
Q(band__influences=influence)
            else:
                influence_condition=influence_condition |
Q(band__influences=influence)
            adverts=adverts.filter(influence_condition)
        else:

adverts=adverts.filter(band__influences__in=musician.influences)
            instruments=self.request.GET.get('instruments', None)
            instruments_list=instruments.split(',')
            instrument_condition = None

instrument_unite=int(self.request.GET.get('instrumentadd', 0))
        for instrument in instruments_list:
            instrument=Instrument.objects.get(title=instrument)
            if not instrument_condition:

instrument_condition=Q(band__instruments=instrument)
            elif instrument_unite:
                instrument_condition=instrument_condition &
Q(band__instruments=instrument)
            else:
                instrument_condition=instrument_condition |
Q(band__instruments=instrument)
            if instrument_condition:
                adverts=adverts.filter(instrument_condition)
            positions=self.request.GET.get('positions', None)

```

```

        positions_list=positions.split(',')
        positions_condition=None
        positions_unite=int(self.request.GET.get('positionadd',
0))
        for position in positions_list:
            position=Position.objects.get(title=position)
            if not positions_condition:

positions_condition=Q(band__members__positions=position)
            elif positions_unite:
                positions_condition=positions_condition &
Q(band__members__positions=position)
            else:
                positions_condition=positions_condition |
Q(band__members__positions=position)
            if not positions_condition:
                adverts=adverts.filter(positions_condition)
                overall_level=self.request.GET.get('level1', 0)

adverts=adverts.exclude(band__members__musician__overall_level__lte
=int(overall_level))
                overall_level_max=self.request.GET.get('level2', 0)
                if overall_level_max:

adverts=adverts.exclude(band__members__musician__overall_level__gte
=int(overall_level_max))
                music_theory=self.request.GET.get('theory1', 0)

adverts=adverts.exclude(band__members__musician__music_theory__lte=
int(music_theory))

                music_theory_max=self.request.GET.get('theory2', 0)
                if music_theory_max:

adverts=adverts.exclude(band__members__musician__music_theory__gte=
int(music_theory_max))
                recording_level=self.request.GET.get('record1', 0)

adverts=adverts.exclude(band__members__musician__recording_level__l
te=int(recording_level))
                recording_level_max=self.request.GET.get('record2', 0)
                if recording_level_max:

adverts=adverts.exclude(band__members__musician__recording_level__g
te=int(recording_level_max))
                adverts=adverts.order_by('-create_time')
                if request.user.is_active:

notes=MemberAdvert.objects.filter(member_advert__in=adverts,
musician=musician)

```

```

        return render(request, "musicianseeker/seek_band.html",
{"adverts": adverts, "notes": notes})

def seek_for_project_view(request):
    now_time=datetime.now()
    if request.user.is_active:

musician=Musician.objects.get(django_user=request.user)
        musadvert=
MusicianAdvert.objects.get(musician__django_user=request.user)
        band_city=musician.location_city
    else:
        band_city=self.request.GET.get('city', None)
        if band_city:
            band_city=City.objects.get(title=band_city, None)
        if band_city:

adverts=ProjectAdvert.objects.filter(band__location_city=band_city)
    else:
        return render(request,
"musicianseeker/seek_project.html", {"adverts": []})
        adverts=adverts.filter(active=True)
        if request.user.is_active:
            musician_instruments=
CapabilityInstrument.objects.filter(capability__musician_advert=mus
advert).distinct('instrument')
            condition=None
            for musician_instrument in musician_instruments:
                if not condition:
                    condition=Q(instrument=musician_instrument)
                else:
                    condition=condition |
Q(instrument=musician_instrument)
            adverts=adverts.filter(condition)
        else:
            instruments=self.request.GET.get('musinstruments',
None)

            if instruments:
                instruments_list=genres.split(',')
                instruments_condition=None
                for instrument in instruments_list:

instrument=Instrument.objects.get(title=instrument)
                if not instruments_condition:

instruments_condition=Q(instrument=instrument)
                else:

instruments_condition=instruments_condition |
Q(instrument=instrument)

```

```

        adverts=adverts.filter(instruments_condition)
    if request.user.is_active:
        start_year=musician.start_year
        experience=now_time.year-start_year
    else:
        experience=self.request.GET.get('experience', None)
    if experience:

adverts=adverts.filter(experience__lte=int(experience))
    if request.user.is_active:
        overall_level=musician.overall_level
    else:
        overall_level=self.request.GET.get('level', None)
    if overall_level:

adverts=adverts.filter(overall_level__lte=int(overall_level))
    if request.user.is_active:
        music_theory=musician.music_theory
    else:
        music_theory=self.request.GET.get('theory', None)
    if music_theory:

adverts=adverts.filter(music_theory__lte=int(music_theory))
    if request.user.is_active:
        recording_level=musician.recording_level
    else:
        recording_level=self.request.GET.get('rec', None)
    if recording_level:

adverts=adverts.filter(recording_level__lte=int(recording_level))
    seek_text=self.request.GET.get('search', '')
    if seek_text:

adverts=adverts.filter(Q(project__title__icontains=seek_text) |
Q(project__idea__icontains=seek_text))
        forbidden_text=self.request.GET.get('forbid', '')
        forbidden_text_list=forbidden_text.split(' ')
        if forbidden_text_list:
            for forbid in forbidden_text_list:

adverts=adverts.exclude(Q(project__title__icontains=seek_text) |
Q(project__idea__icontains=seek_text))
        band_gigs=self.request.GET.get('gigs1', 0)
        if band_gigs:
            adverts=adverts.filter(project__band__gigs__gte =
int(band_gigs))
        band_gigs_max=self.request.GET.get('gigs2', 0)
        if band_gigs_max:
            adverts=adverts.filter(project__band__gigs__lte =
int(band_gigs_max))

```

```

        band_experience=self.request.GET.get('years1', 0)
        band_experience_end=now_time.year - int(band_experience)
        band_experience_max=self.request.GET.get('years2', 0)
        band_experience_start=now_time.year -
int(band_experience_max)
        if band_experience_start > band_experience_end:

adverts=adverts.filter(project__band__start_year__gte=band_experien
ce_start, band__start_year__lte=band_experience_end)
        elif band_experience and band_experience_max==0:

adverts=adverts.filter(project__band__start_year__lte=band_experien
ce_end)
        elif band_experience_max and band_experience==0:

adverts=adverts.filter(project__band__start_year__gte=band_experien
ce_start)
        genres=self.request.GET.get('genres', None)
        genres_list=genres.split(',')
        genres_condition=None
        genre_unite=int(self.request.GET.get('genreadd', 0))
        for genre in genres_list:
            genre=Genre.objects.get(title=genre)
            if not genres_condition:
                genres_condition=Q(project__genres=genre)
            elif genre_unite:
                genres_condition=genres_condition &
Q(project__genres=genre)
            else:
                genres_condition=genres_condition |
Q(project__genres=genre)
        if genres_condition:
            adverts=adverts.filter(genres_condition)
        if not request.user.is_active:
            influences=self.request.GET.get('influences', None)
            influences_list=influences.split(',')
            influence_condition = None

influence_unite=int(self.request.GET.get('influenceadd', 0))
        for influence in influences_list:
            influence=Influence.objects.get(title=influence)
            if not influence_condition:

influence_condition=Q(project__band__influences=influence)
            elif influence_unite:
                influence_condition=influence_condition &
Q(project__band__influences=influence)
            else:
                influence_condition=influence_condition |
Q(project__band__influences=influence)

```

```

        adverts=adverts.filter(influence_condition)
    else:

adverts=adverts.filter(project__band__influences__in=musician.influences)

        instruments=self.request.GET.get('instruments', None)
        instruments_list=instruments.split(',')
        instrument_condition = None

instrument_unite=int(self.request.GET.get('instrumentadd', 0))
    for instrument in instruments_list:
        instrument=Instrument.objects.get(title=instrument)
        if not instrument_condition:

instrument_condition=Q(project__instruments=instrument)
        elif instrument_unite:
            instrument_condition=instrument_condition &
Q(project__instruments=instrument)
        else:
            instrument_condition=instrument_condition |
Q(project__instruments=instrument)
        if instrument_condition:
            adverts=adverts.filter(instrument_condition)
            positions=self.request.GET.get('positions', None)
            positions_list=positions.split(',')
            positions_condition=None
            positions_unite=int(self.request.GET.get('positionadd',
0))

            for position in positions_list:
                position=Position.objects.get(title=position)
                if not positions_condition:

positions_condition=Q(project__band__members__positions=position)
                elif positions_unite:
                    positions_condition=positions_condition &
Q(project__band__members__positions=position)
                else:
                    positions_condition=positions_condition |
Q(project__band__members__positions=position)
                if positions_condition:
                    adverts=adverts.filter(positions_condition)
                    overall_level=self.request.GET.get('level1', 0)

adverts=adverts.exclude(project__band__members__musician__overall_level__lte=int(overall_level))
                    overall_level_max=self.request.GET.get('level2', 0)
                    if overall_level_max:

adverts=adverts.exclude(project__band__members__musician__overall_level__gte=int(overall_level_max))

```

```

        music_theory=self.request.GET.get('theory1', 0)

adverts=adverts.exclude(project__band__members__musician__music_theory__lte=int(music_theory))

        music_theory_max=self.request.GET.get('theory2', 0)
        if music_theory_max:

adverts=adverts.exclude(project__band__members__musician__music_theory__gte=int(music_theory_max))
        recording_level=self.request.GET.get('record1', 0)

adverts=adverts.exclude(project__band__members__musician__recording_level__lte=int(recording_level))
        recording_level_max=self.request.GET.get('record2', 0)
        if recording_level_max:

adverts=adverts.exclude(project__band__members__musician__recording_level__gte=int(recording_level_max))
        adverts=adverts.order_by('-project__create_time')

capinstruments=CapabilityInstrument.objects.filter(capability__musician_advert=musadvert)
        if request.user.is_active:
            diff=musician.date_of_birth - now_time
            age=diff.years
        else:
            age=self.request.GET.get('age', None)
        if age:
            adverts=adverts.filter(age_from__lte=age, age_to__gte=age)
            instruments_experience={}
            for capinstrument in capinstruments:

instruments_experience[capinstrument.instrument.title]=capinstrument.capability.experience
            for advert in adverts:
                diff=advert.experience -
instruments_experience[advert.instrument.title]
                if diff>0:
                    advert.experience_diff=diff
                else:
                    advert.experience_diff=0
            if request.user.is_active:

notes=ProjectAdvertNote.objects.filter(project_advert__in=adverts, musician=musician)
        return render(request, "musicianseeker/seek_project.html", {"adverts": adverts, "notes": notes})

```

```

def seek_for_musician_view(request):
    now_time=datetime.now()
    if request.user.is_active:

musician=Musician.objects.get(django_user=request.user)
        bands=Member.objects.filter(musician,
active=True).distinct('band')

memberadverts=MemberAdvert.objects.get(band__in=bands, active=True)

projadverts=ProjectAdvert.objects.get(project__band__in= bands,
active=True)
        band=self.request.GET.get('band', None)
        if band:
            band=bands.values()[int(band)]
            bands=[band]
            memberadverts=MemberAdvert.objects.get(band=band,
active=True)

            memberadvert=self.request.GET.get('memberad',
None)

            if memberadvert:

memberadverts=[memberadverts.values()[int(memberadvert)]]

projadverts=ProjectAdvert.objects.get(project__band= band,
active=True)
            projadvert=self.request.GET.get('projad', None)
            if projadvert:

projadverts=[projadverts.values()[int(projadvert)]]
                if memberadvert and not projadvert:
                    projadverts=[]
                elif projadvert and not memberadvert:
                    memberadverts=[]
                for memberadvert in memberadverts:
                    age_from=datetime.datetime(now_time.year -
memberadvert.age_to, now_time.month, now_time.day)
                    age_to=datetime.datetime(now_time.year -
memberadvert.age_from, now_time.month, now_time.day)

musadverts=Capability.objects.filter(position=memberadvert.position
,
            experience__gte=memberadvert.experience,
            active=True,

musician_advert__practice_availability__gte=memberadvert.practice_a
vailability,

musician_advert__gig_availability__gte=memberadvert.gig_availabilit
y)

```

```

musician_advert__musician__date_of_birth__gte=age_from,

musician_advert__musician__date_of_birth__lte=age_to

musician_advert__musician__overall_level__gte=memberadvert.overall_
level,

musician_advert__musician__music_theory__gte=memberadvert.music_the
ory,

musician_advert__musician__recording_level__gte=memberadvert.record
ing_level).distinct('musician_advert')
        band_cities=bands.distinct('location_city')

musadverts=musadverts.filter(location_city__in=band_cities)

notes=MusicianAdvertNote.objects.filter(musician_advert__in=musadve
rts, musician=musician)
        for projadvert in projadverts:
            age_from=datetime.datetime(now_time.year -
memberadvert.age_to, now_time.month, now_time.day)
            age_to=datetime.datetime(now_time.year -
memberadvert.age_from, now_time.month, now_time.day)

musadverts2=CapabilityInstrument.objects.filter(instrument=projadve
rt.instrument,

capability__experience__gte=projadvert.experience,
            active=True,
            musician_advert__project_opened=True,

capability__musician_advert__musician__date_of_birth__gte=age_from,

capability__musician_advert__musician__date_of_birth__lte=age_to

capability__musician_advert__musician__overall_level__gte=projadver
t.overall_level,

capability__musician_advert__musician__music_theory__gte=projadvert
.music_theory,

capability__musician_advert__musician__recording_level__gte=projadv
ert.recording_level).distinct('capability__musician_advert')
        band_cities=bands.distinct('location_city')

musadverts2=musadverts2.filter(location_city__in=band_cities)

notes2=MusicianAdvertNote.objects.filter(musician_advert__in=musadv
erts2, musician=musician)

```

```

        return render(request,
"musicianseeker/seek_musician.html", {"musadverts_for_band":
musadverts, "musadverts_for_project": musadverts2, "member_notes":
notes, "project_notes": notes2})
    else:
        band_city=self.request.GET.get('city', None)
        if band_city:
            band_city=City.objects.get(title=band_city, None)
        if band_city:
            adverts=
MusicianAdvert.objects.filter(location_city=band_city)
        else:
            return render(request,
"musicianseeker/seek_musician.html", {"adverts": []})
            overall_level=self.request.GET.get('level', None)
            if overall_level:

adverts=adverts.filter(overall_level__lte=int(overall_level))
            music_theory=self.request.GET.get('theory', None)
            if music_theory:

adverts=adverts.filter(music_theory__lte=int(music_theory))
            recording_level=self.request.GET.get('rec', None)
            if recording_level:

adverts=adverts.filter(recording_level__lte=int(recording_level))
            age_to=self.request.GET.get('age2', None)
            if age_to:
                age_from=datetime.datetime(now_time.year -
int(age_to), now_time.month, now_time.day)

adverts=adverts.filter(musician__date_of_birth__gte=age_from)
            age_from=self.request.GET.get('age1', None)
            if age_from:
                age_to=datetime.datetime(now_time.year -
int(age_from), now_time.month, now_time.day)

adverts=adverts.filter(musician__date_of_birth__lte=age_to)
            notes=[]
            notes2=[]
            return render(request,
"musicianseeker/seek_musician.html", {"adverts": adverts,
"member_notes": notes, "project_notes": notes2})

    def band_projects_view(request, id=None):
        if id:
            band_projects=Project.objects.filter(band__id=id)
        else:

musician=Musician.objects.get(django_user=request.user)

```

```

        bands=Member.objects.filter(musician,
active=True).distinct('band')
        band_projects=Project.filter(band__in=bands)
        return render(request,
"musicianseeker/band_projects.html", {"band_projects":
band_projects})

    def band_adverts_view(request, id=None):
        if id:
            band=Band.objects.filter(pk=id)
            member_adverts=MemberAdvert.objects.filter(band=band,
active=True).order_by('-create_time')

project_adverts=ProjectAdvert.objects.filter(project__band=band,
active=True).order_by('-create_time')
            else:

musician=Musician.objects.get(django_user=request.user)
            bands=Member.objects.filter(musician,
active=True).distinct('band')

member_adverts=MemberAdvert.objects.filter(band__in=bands).order_by
('-create_time')

project_adverts=ProjectAdvert.objects.filter(project__band__in=band
s).order_by('-create_time')
            if request.user.is_active:

member_notes=MemberAdvertNote.objects.filter(member_advert__in=
member_adverts, musician=musician)

project_notes=ProjectAdvertNote.objects.filter(project_advert__in=
project_adverts, musician=musician)
            return render(request,
"musicianseeker/band_adverts.html", {"band_project_adverts":
project_adverts, "band_memeber_adverts": member_adverts,
"project_notes": project_notes, "member_notes": member_notes})

    def musician_adverts_view(request, id=None):
        if id:

musician_adverts=MusicianAdvert.objects.filter(musician__id=id)
            else:

musician=Musician.objects.get(django_user=request.user)

musician_adverts=Capability.objetc.filter(musician_advert__musician
=musician, active=True).distinct('musician_advert')
            if request.user.is_active:

```

```

musician_notes=MusicianAdvertNote.objects.filter(musician_advert__i
n=member_adverts, musician=musician)
        return render(request,
"musicianseeker/musician_adverts.html", {"musician_adverts":
musician_adverts, "musician_notes": musician_notes})

    def city_music_view(request, id=None):
        if id:
            loc_city=City.objects.get(id=id)
            bands=Band.objects.filter(location_city=loc_city)
            musicians=
MusicianAdvert.objects.filter(location_city=loc_city).distinct('mus
ician')
        else:
            bands=[]
            musicians=[]
            if request.user.is_active:

musician_notes=MusicianAdvertNote.objects.filter(musician_advert__i
n= musicians, musician__django_user=request.user)
                return render(request, "musicianseeker/city_music.html",
{"musicians": musicians, "bands": bands, "musician_notes":
musician_notes})

    def band_view(request, id=None):
        if id:

band=Band.objects.get(pk=id).prefetch_related('members')

member_adverts=MemberAdvert.objects.filter(band=band).order_by('-
create_time')[:2]

project_adverts=ProjectAdvert.objects.filter(project__band=band).or
der_by('-project__create_time')[:2]

projects=Project.objects.filter(band=band).order_by('-
create_time')[:2]
        else:
            band=None
            member_adverts=[]
            project_adverts=[]
            projects=[]
            if request.user.is_active:

member_notes=MemberAdvertNote.objects.filter(member_advert__in=
member_adverts, musician=musician)

project_notes=ProjectAdvertNote.objects.filter(project_advert__in=
project_adverts, musician=musician)

```

```

        return render(request, "musicianseeker/band.html",
{"band": band, "member_adverts": member_adverts, "project_adverts":
project_adverts, "projects": projects, "project_notes":
project_notes, "member_notes": member_notes})

    def musician_view(request, id=None):
        noted=False
        if id:
            musician=Musician.objects.get(pk=id)

musician_advert=MusicianAdvert.objects.get(musician=musician)

equipments=Equipment.objects.filter(musician=musician)
        bands=Member.objects.filter(musician=musician,
active=True).distinct('band')

note=MusicianAdvertNote.objects.filter(musician_advert=musician_adv
ert, musician__django_user=request.user)
        if note:
            noted=True
        else:

musician=Musician.objects.get(django_user=request.user)

musician_advert=MusicianAdvert.objects.get(musician=musician)

equipments=Equipment.objects.filter(musician=musician)

bands=Member.objects.filter(musician=musician).distinct('band')
        return render(request, "musicianseeker/musician.html",
{"musician": musician, "musician_advert": musician_advert, "bands":
bands, "equipments": equipments, "noted": noted})

    def musician_advert_view(request, id=None):
        noted=False
        if id:
            musician=Musician.objects.get(pk=id)

musician_advert=MusicianAdvert.objects.get(musician=musician)

capabilities=Capability.objects.filter(musician_advert=musician_adv
ert).prefetch_related('instruments')

equipments=Equipment.objects.filter(musician=musician)
        bands=Member.objects.filter(musician=musician,
active=True).distinct('band')

note=MusicianAdvertNote.objects.filter(musician_advert=musician_adv
ert, musician__django_user=request.user)
        if note:

```

```

        noted=True
        return render(request, "musicianseeker/musician.html",
{"musician": musician, "musician_advert": musician_advert, "bands":
bands, "equipments": equipments, "noted": noted, "capabilities":
capabilities})

    def member_advert_view(request, id=None):
        noted=False
        if id:

member_advert=MemberAdvert.objects.filter(pk=id).prefetch_related('
band').prefetch_related('instruments')

note=MemberAdvertNote.objects.filter(member_advert=member_advert,
musician__django_user=request.user)
        if note:
            noted=True
            return render(request,
"musicianseeker/member_advert.html", {"member_advert":
member_advert, "noted": noted})

    def project_advert_view(request, id=None):
        noted=False
        if id:

project_advert=ProjectAdvert.objects.filter(pk=id).prefetch_related
('project')

note=ProjectAdvertNote.objects.filter(project_advert=project_advert
, musician__django_user=request.user)
        if note:
            noted=True
            return render(request,
"musicianseeker/project_advert.html", {"project_advert":
project_advert, "noted": noted})

```

Б.2 Текст файла admin.py

```

from django.contrib import admin
from .models import MemberAdvert
from .models import Band
from .models import Bandlink
from .models import Music
from .models import Project
from .models import ProjectAdvert
from .models import City
from .models import Genre
from .models import ProjectApplication

```

```

from .models import ProjectAdvertNote
from .models import Country
from .models import MemberApplication
from .models import MemberAdvertNote
from .models import Instrument
from .models import Equipment
from .models import Influence
from .models import EquipmentBrand
from .models import Position
from .models import Member
from .models import Musician
from .models import Notice
from .models import MusicianAdvertNote
from .models import MusicianAdvert
from .models import Capability

class MemberAdvertAdmin(admin.ModelAdmin):
    exclude = ["create_time"]

admin.site.register(MemberAdvert, MemberAdvertAdmin)
admin.site.register(Band)
admin.site.register(Bandlink)
admin.site.register(Music)

class MusicProjectAdmin(admin.ModelAdmin):
    exclude = ["create_time"]

admin.site.register(Project, MusicProjectAdmin)
admin.site.register(ProjectAdvert)
admin.site.register(City)
admin.site.register(Genre)

class ProjectApplicationAdmin(admin.ModelAdmin):
    exclude = ["create_time"]

admin.site.register(ProjectApplication, ProjectApplicationAdmin)
admin.site.register(ProjectAdvertNote)
admin.site.register(Country)

class MemberApplicationAdmin(admin.ModelAdmin):
    exclude = ["create_time"]

admin.site.register(MemberApplication, MemberApplicationAdmin)
admin.site.register(MemberAdvertNote)
admin.site.register(Instrument)
admin.site.register(Equipment)
admin.site.register(Influence)
admin.site.register(EquipmentBrand)
admin.site.register(Position)

```

```
admin.site.register(Member)

class MusicianAdmin(admin.ModelAdmin):
    exclude = ["django_user"]

admin.site.register(Musician, MusicianAdmin)

class NoticeAdmin(admin.ModelAdmin):
    exclude = ["time_mark"]

admin.site.register(Notice, NoticeAdmin)
admin.site.register(MusicianAdvertNote)
admin.site.register(MusicianAdvert)
admin.site.register(Capability)
```

ДОДАТОК В
Слайди презентації

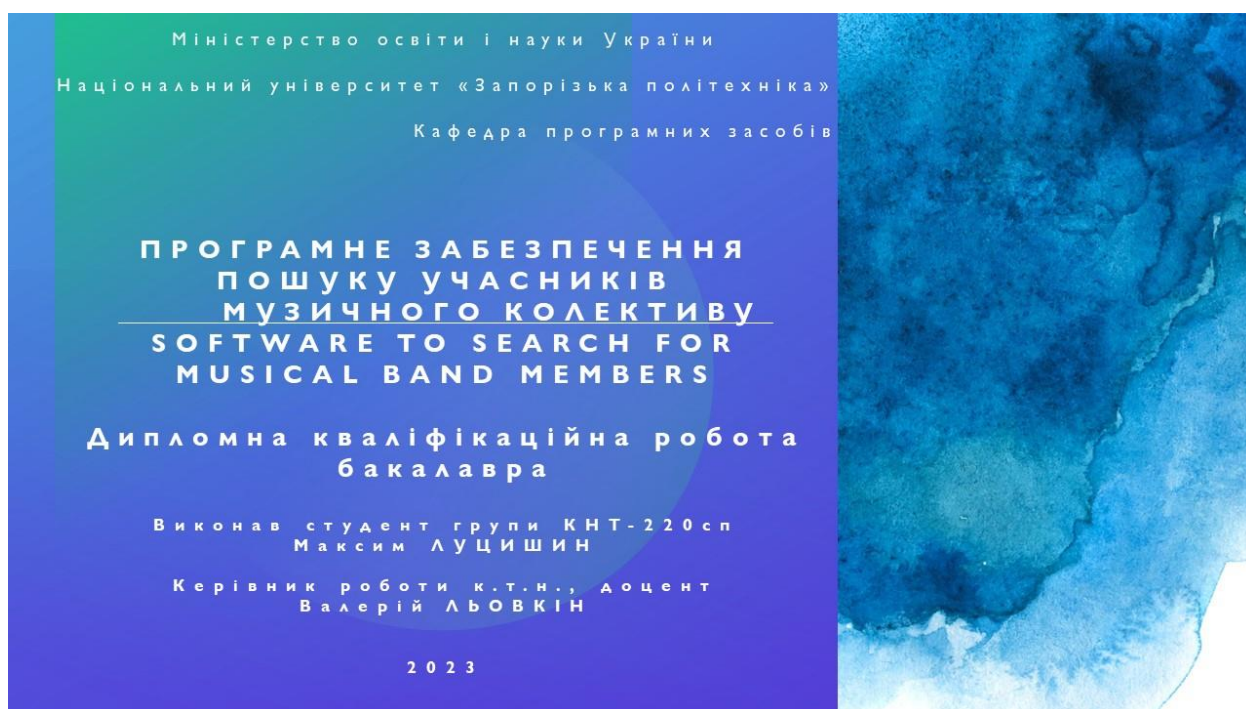


Рисунок В.1 – Початковий слайд

ОБ'ЄКТ, ПРЕДМЕТ І МЕТА ДИПЛОМНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

- об'єкт роботи – процес пошуку учасників музичного колективу;
- предмет роботи – програмне забезпечення пошуку учасників музичного колективу;
- мета роботи – розробити програмне забезпечення пошуку учасників музичного колективу для спрощення взаємодії між музикантами і музичними колективами.



Рисунок В.2 – Мета дипломної кваліфікаційної роботи

ЗАВДАННЯ ДИПЛОМНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

- аналіз існуючих програм пошуку учасників музичного колективу;
- визначення функціональних вимог до програми;
- проектування програмного забезпечення пошуку учасників музичного колективу;
- реалізація і тестування програми.

3

Рисунок В.3 – Завдання дипломної кваліфікаційної роботи

ПОРІВНЯННЯ АНАЛОГІВ З ДИПЛОМНОЮ ПРОГРАМОЮ ЗА РЕЗУЛЬТАТАМИ ОГЛЯДУ

Характеристика	Hendrix	Bandmix	BandFinder	Join-A-Band	Дипломна програма
Країни	США	Великобританія	США, Великобританія, Канада, Швеція	Усі, в тому числі Україна	Без обмежень
Пошук музикантами колективів	+	+	+	+	+
Пошук постійних учасників колективів	+	+	+	+	+
Пошук музикантів для проєктів	–	–	–	–	+

4

Рисунок В.4 – Порівняння аналогів з дипломною програмою за результатами огляду

ПОРІВНЯННЯ ВЕБФРЕЙМВОРКІВ BOTTLE І DJANGO

Показник	Bottle	Django
Мова програмування	Python	Python
Тип вебфреймворку	Мікрофреймворк	Фреймворк високого рівня
Доступність об'єктно-реляційного відображення	Доступний для підключення	Вбудований
Вбудоване керування даними	Відсутнє	Наявне
Підтримка на рівні документації	Гірша	Краща
Вбудована підтримка вебформ	Відсутня	Наявна
Вбудована аутентифікація	Відсутня	Наявна
Вбудована підтримка безпеки	Відсутня	Наявна

Рисунок В.5 – Порівняння вебфреймворків Bottle і Django

ДІАГРАМИ ВАРІАНТІВ ВИКОРИСТАННЯ ПРОГРАМИ

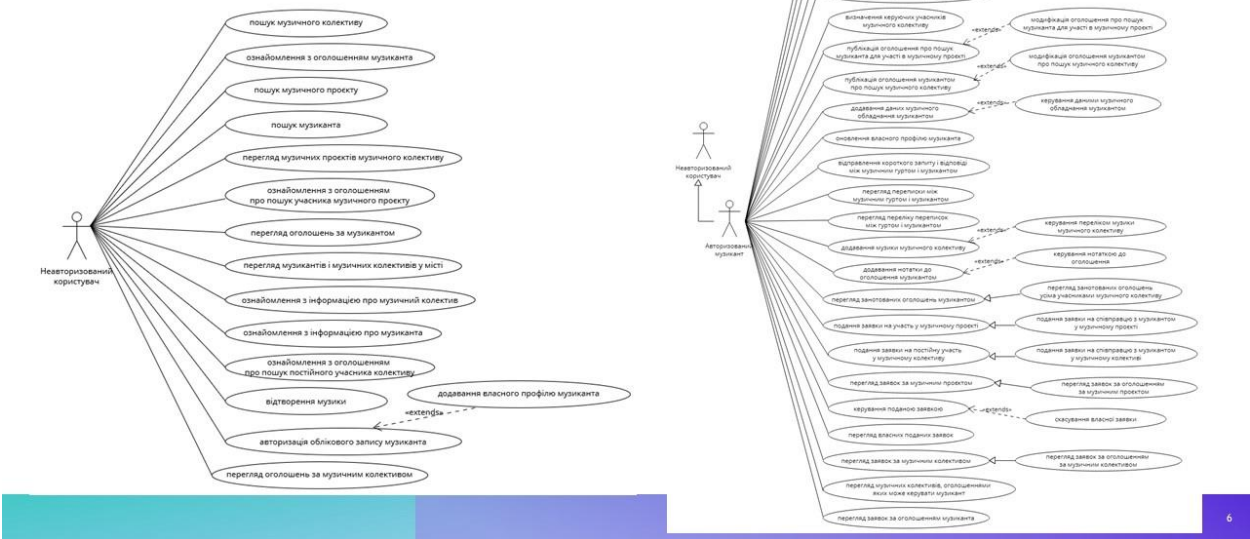


Рисунок В.6 – Діаграми варіантів використання програми

project	sheet	level	type	safety



project	sheet	level	type	safety

8



8

ПОШУК ОГОЛОШЕНЬ У ПРОГРАМІ

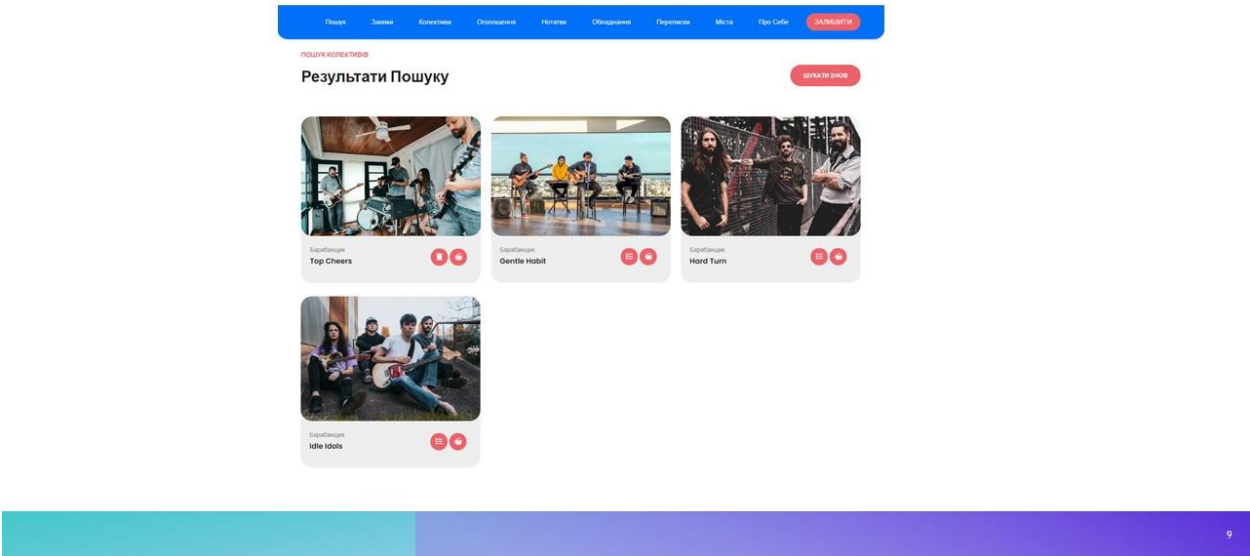


Рисунок В.9 – Пошук оголошень у програмі

МУЗИЧНИЙ ПРОЄКТ У ПРОГРАМІ

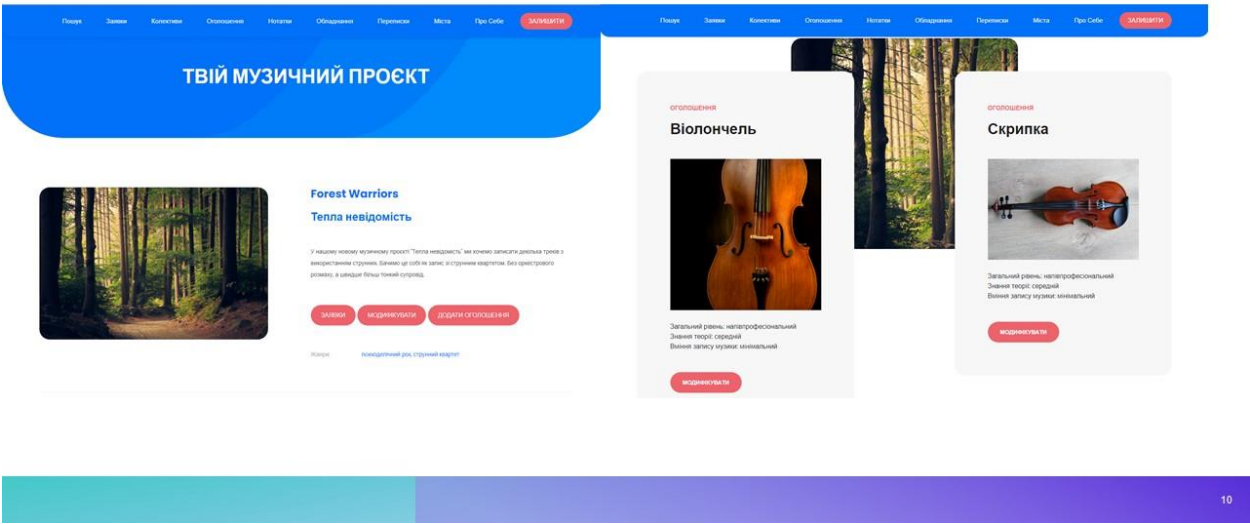


Рисунок В.10 – Музичний проєкт у програмі

ЗАЯВКИ МУЗИКАНТА У ПРОГРАМІ

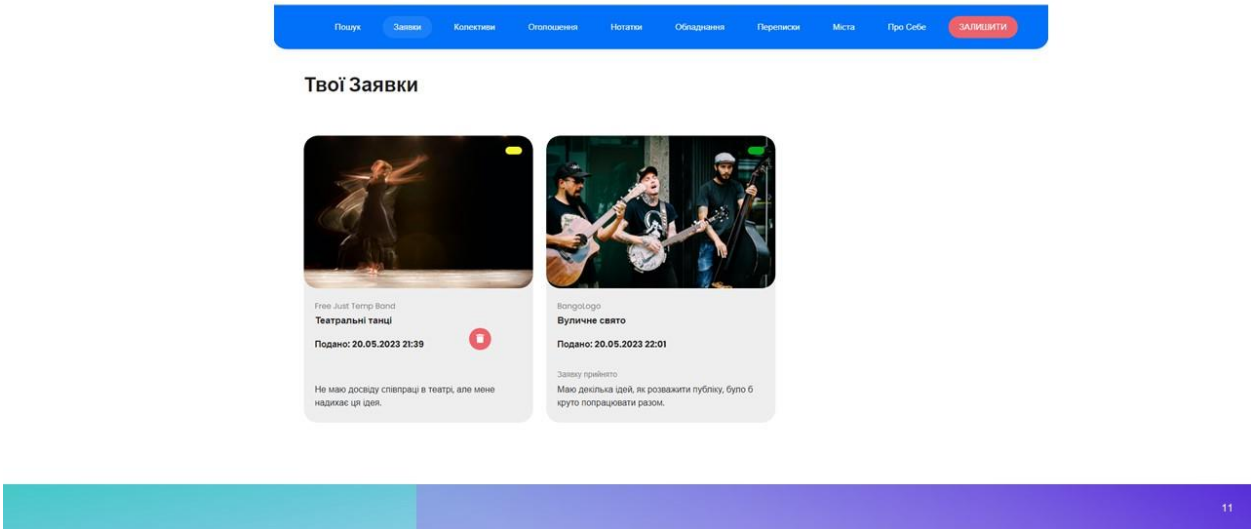


Рисунок В.11 – Заявки музиканта у програмі

РЕЗУЛЬТАТИ ТЕСТУВАННЯ ПРОГРАМИ

Функція	Результат
Пошук музичного колективу	Успіх
Пошук музичного проекту	Успіх
Пошук музиканта	Успіх
Перегляд музичних проєктів музичного колективу	Успіх
Перегляд оголошень за музичним колективом	Успіх
Перегляд оголошень за музикантом	Успіх
Перегляд музикантів і музичних колективів у місті	Успіх
Ознайомлення з інформацією про музичний колектив	Успіх
Ознайомлення з інформацією про музиканта	Успіх
Ознайомлення з оголошенням про пошук постійного учасника колективу	Успіх
Ознайомлення з оголошенням про пошук учасника музичного колективу	Успіх
Ознайомлення з оголошенням музиканта	Успіх
Відтворення музики	Успіх
Додавання власного профілю музиканта	Успіх
Авторизація облікового запису музиканта	Успіх
Додавання власного музичного колективу	Успіх
Модифікація даних власного музичного колективу	Успіх
Додавання учасників музичного колективу	Успіх
Вилучення учасників музичного колективу	Успіх
Визначення позиції учасника в музичному колективі	Успіх
Публікація оголошення про пошук музиканта для постійної участі в музичному колективі	Успіх
Подання заявки на співпрацю з музикантом у музичному колективі	Успіх
Подання заявки на співпрацю з музикантом у музичному проєкті	Успіх
Перегляд заявок за музичним проєктом	Успіх
Перегляд заявок за оголошенням за музичним проєктом	Успіх
Перегляд заявок за музичним колективом	Успіх
Модифікація оголошення про пошук музиканта для постійної участі в музичному колективі	Успіх

Рисунок В.12 – Результати тестування програми

РЕЗУЛЬТАТИ ТЕСТУВАННЯ ПРОГРАМИ

Функція	Результат
Додавання нового музичного проекту колективу	Успіх
Модифікація даних музичного проекту	Успіх
Визначення керуючих учасників музичного колективу	Успіх
Публікація оголошення про пошук музиканта для участі в музичному проекті	Успіх
Модифікація оголошення про пошук музиканта для участі в музичному проекті	Успіх
Публікація оголошення музикантом про пошук музичного колективу	Успіх
Модифікація оголошення музикантом про пошук музичного колективу	Успіх
Додавання даних музичного обладнання музикантом	Успіх
Керування даними музичного обладнання музикантом	Успіх
Відправлення короткого запиту і відповіді між музичним гуртом і музикантом	Успіх
Перегляд переписки між музичним гуртом і музикантом	Успіх
Перегляд переліку переписок між гуртом і музикантом	Успіх
Додавання музики музичного колективу	Успіх
Керування переліком музики музичного колективу	Успіх
Додавання нотатки до оголошення музикантом	Успіх
Перегляд занотованих оголошень музикантом	Успіх
Перегляд занотованих оголошень усіма учасниками музичного колективу	Успіх
Керування нотаткою до оголошення	Успіх
Оновлення власного профілю музиканта	Успіх
Подання заявки на участь у музичному проекті	Успіх
Перегляд заявок за оголошенням за музичним колективом	Успіх
Перегляд заявок за оголошенням музиканта	Успіх
Перегляд музичних колективів, оголошеннями яких може керувати музикант	Успіх
Перегляд власних поданих заявок	Успіх
Керування поданою заявкою	Успіх
Скасування власної заявки	Успіх
Подання заявки на постійну участь у музичному колективу	Успіх

13

Рисунок В.13 – Результати тестування програми

ВИСНОВКИ

- розробка програмного забезпечення пошуку учасників музичного колективу для спрощення взаємодії між музикантами і музичними колективами була визначена метою дипломної кваліфікаційної роботи і була разом з завданнями досягнута в роботі;
- програмне забезпечення надає можливості пошуку не тільки музикантів для участі в колективі на певних позиціях і навпаки, але і пошуку музикантів саме під окремі музичні проекти для реалізації тимчасових ідей колективів, надаючи для цього засоби створення нотаток з оголошень, публікації оголошень, подачі заявок, керування заявками і оголошеннями.

14

Рисунок В.14 – Висновки за дипломною кваліфікаційною роботою