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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка веб-сайту “Artise” засобами React, Node.js та MongoDB

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

(підпис) Болож О. В. (прізвище та ініціали)

Керівник (підпис) Литвиненко Я. В. (прізвище та ініціали)

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

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

Рецензент (підпис) Осухівська Г.М. (прізвище та ініціали)

Тернопіль
2022

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

Факультет комп’ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп’ютерні науки
(шифр і назва спеціальності)

Студенту Боложу Олександру Васильовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Розробка веб-сайту “Artise” засобами React, Node.js та MongoDB

Керівник роботи Литвиненко Ярослав Володимирович д.т.н., професор кафедри КН
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 16 » березня 2022 року № 4/7-161

2. Термін подання студентом завершеної роботи 20 червня 2022р.

3. Вихідні дані до роботи Літературні та інтернет джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. РОЗДІЛ 1. Постановка задачі розробки веб-сайту “Artise”, 1.1 Аналіз предметної області, 1.2 Формування вимог розробки веб-сайту “Artise”, 1.3 Пошук актантів та варіантів використання веб-сайту “Artise”, 1.3.1 Пошук актантів, 1.3.2 Варіанти використання веб-сайту “Artise”, 1.4 Опис ключових варіантів використання веб-сайту “Artise”, 1.5 Вибір середовища розробки веб-сайту “Artise”, 1.5.1 Мова програмування, 1.5.2 База даних, 1.6 Обґрунтування використовуваних технологій розробки веб-сайту “Artise”, 1.7 Висновок до першого розділу, 2. Проєктування та реалізація веб-сайту “Artise”, 2.1 Архітектура веб-сайту “Artise”, 2.2 Вибір основних залежностей для розробки веб-сайту “Artise”, 2.3 Файлова структура веб-сайту “Artise”, 2.4 Структура та особливості використаної БД для розробки веб-сайту “Artise”, 2.5 Розгортання веб-сайту “Artise” на хостингових платформах, 2.6 Тестування та використання веб-сайту “Artise”, 2.6.1 Тестування веб-сайту “Artise”, 2.6.2 Використання веб-сайту “Artise”, 2.6 Висновки до другого розділу, 3. Безпека життєдіяльності, основи хорони праці, 3.1 Методи боротьби з монотонністю праці на виробництві, 3.2 Загальні вимоги безпеки з охорони праці для користувачів ПК, 3.3 Висновок до третього розділу, Висновки, Перелік використаних джерел, Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень, слайдів)

6. Консультанти розділів роботи

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

7. Дата видачі завдання 24 січня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи		*Виконано*
2.	Пошук джерел щодо розробки та створення веб-		
	сайтів		*Виконано*
3.	Опрацювання обраних джерел		*Виконано*
4.	Аналіз області розробки веб-сайту, підбір технологій		
	для розробки		*Виконано*
5.	Розробка веб-сайту “Artise”		*Виконано*
6.	Оформлення розділу «Постановка задачі розробки		
	веб-сайту “Artise”»		*Виконано*
7.	Оформлення розділу «Проєктування та реалізація		
	веб-сайту “Artise”»		*Виконано*
8.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності»		*Виконано*
9.	Виконання завдання до підрозділу «Основи хорони		
	праці»		*Виконано*
10.	Оформлення кваліфікаційної роботи		*Виконано*
11.	Нормоконтроль		*Виконано*
12.	Перевірка на плагіат		*Виконано*
13.	Попередній захист кваліфікаційної роботи		*Виконано*
14.	Захист кваліфікаційної роботи		*Виконано*

Студент _____ Болож О. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Литвиненко Я. В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка веб-сайту “Artise” засобами React, Node.js та MongoDB // Кваліфікаційна робота освітнього рівня «Бакалавр» // Болож Олександр Васильович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СН-41 // Тернопіль, 2022 // С. 44, рис. – 24, табл. – 2, кресл. – 0 , додат. – 5, бібліогр. – 16.

Ключові слова: react, nodejs, mongodb, express, mern, web, music platform, music, listen music, artise.

Кваліфікаційна робота присвячена розробці веб-сайту “Artise” засобами React, Node.js та MongoDB.

Метою даної роботи є розробка веб-сайту музичної платформи для прослуховування треків їх просування та обміну між користувачами.

В першому розділі кваліфікаційної роботи проведено аналіз предметної області, сформовано основні вимоги для розробки, вибравши при цьому необхідні інструменти, описано взаємодію акторів з системою.

В другому розділі кваліфікаційної роботи розглянуто архітектуру розроблюваного веб-сайту, описано основні залежності та структуру, розглянуто розгортання на хостингових платформах, проведено тестування.

В третьому розділі кваліфікаційної роботи розглянуто питання щодо, методів боротьби з монотонністю праці на виробництві та загальні вимоги безпеки з охорони праці для користувачів ПК.

ANNOTATION

Development of the “Artise” website using React, Node.js and MongoDB // Qualification work of Bachelor educational degree // Bolozh Oleksandr Vasylovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Computer Science Department, SN-41 group // Ternopil, 2022 // Pages – 44, figures – 24 , tables – 2 , sketches – 0, addendums – 5, references – 16.

Keywords: react, nodejs, mongodb, express, mern, web, music platform, music, listen music, artise.

The qualification work is concerned with the development of the “Artise” website using React, Node.js, and MongoDB.

The aim of this work is the development of a musical platform for listening, promoting, and sharing tracks

In the first section of the qualification work, an analysis of the subject area was carried out, and the main requirements for development were formed while selecting the necessary tools, and describing the interaction of actors with the system.

The second section of the qualification work considers the architecture of the developed website, describes the main dependencies and structure, considers the deployment on hosting platforms and conducted testing.

The third section of the qualification work deals with the issue of methods of combating monotony of labor in production and general safety requirements for labor protection for PC users.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CSS – (*від англ.* Cascading Style Sheets) – каскадні таблиці стилів.

HTML – (*від анг.* Hypertext Markup Language) – мова гіпертекстової розмітки.

JS – (JavaScript) – мова сценаріїв чи програмування, що дозволяє реалізувати функціонал веб-ресурсу.

DB – (*від англ.* Database) – база даних.

Стек (*від англ.* Stack) – набір технологій, для розробки веб–сайту.

Баг (*від англ.* Bug) – несправність, яку було виявлено на етапі розробки чи користування.

Трек (*від англ.* Track) – музичний запис.

Шарінг (*від англ.* Share) – поширення, обмін.

MERN – вид стеку, який розшифровується як MongoDB, Express.js, React, Node.js.

Use case – діаграма варіантів використання.

Крос-платформеність – здатність програмного забезпечення працювати більш ніж на одній апаратній платформі або під керівництвом більш ніж однієї операційної системи.

DOM – (*від англ.* Document Object Model) – це програмний інтерфейс для веб-документів. Він представляє сторінку так, що програми можуть змінювати структуру, стиль і зміст документа.

SPA (*від англ.* Single Page Application) – веб-сайт, що завантажує вміст на одну HTML сторінку, де завдяки за допомогою оновлення JavaScript, за час використання не потрібно перезаватнажувати сторінку.

UI (*від англ.* User Interface) – це серія екранів, сторінок і візуальних елементів, таких як кнопки та значки, які дозволяють людині взаємодіяти з продуктом або послугою.

ЗМІСТ

ВСТУП

Актуальність теми. В сучасних реаліях важко уявити своє життя без прослуховування музики, адже музика в свою чергу заряджає нас на емоції. Музика може змінити будь який поганий сценарій чи стан. З розвитком IT технологій змінився і сам підхід прослуховування та обміну треками. Раніше необхідно було завантажити трек на пристрій, і лише тоді його прослухати, а щоб ним поділитися треба було вдаватися до крайнощів обміну через Bluetooth, флеш носій, чи в гіршому випадку через ІЧ порт, що вже далеко в минулому. В сучасних реаліях, це можна реалізувати, як хмарне сховище у вигляді веб-сайту (платформи), що значно полегшить просування, популяризацію, гнучкість та зручність в першу чергу, особливо у випадку реалізації крос-платформеності.

Мета і задачі дослідження. Метою цієї кваліфікаційної роботи освітнього рівня “Бакалавр”, є реалізація веб-сайту музичної платформи для прослуховування треків їх просування та обміну, використовуючи при цьому сучасний інструментарій розробки програмного забезпечення.

Для успішної реалізації задачі, було сформовано ряд наступних завдань:

– вибрати сучасний інструментарій для розробки (стек);

– провести розробку як серверної частини, так і клієнтської;

– перевірити змістовність та сумісність вибраного інструментарію;

– провести тестування для виявлення багів та їх усунення;

– розгорнути веб-сайт на хостингу, для можливості функціонування.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ВЕБ-САЙТУ “ARTISE”

1.1 Аналіз предметної області

Музична платформа – це веб-сайт який повинен, насамперед, забезпечити якісне прослуховування треків та їх можливість шарінгу між користувачами. По перше важливими елементами є можливість користувача додавати треки, їх редагувати та прослуховувати, створювати плей листи, та отримувати фідбек від інших користувачів. Важливо реалізувати “мультиюзерність”, аби кожен користувач зміг авторизуватися, гнучко використати інструментарій сайту виходячи зі своїх інтересів, легко знайти, той чи інший трек, додати до улюблених, та зберегти попередню сесію.

Відповідно поставленого функціоналу, для повноцінної та коректної роботи всі записи важливо синхронізувати з базою даних.

Також розробка мобільного застосунку не передбачається, то для збільшення аудиторії, важливо, щоб веб-сайт був крос-платформенним, тобто коректно відображатися як на ПК чи ноутбуках, так і на мобільних пристроях.

1.2 Формування вимог розробки веб-сайту “Artise”

Перед тим, як починати проектування будь-якої системи, важливо в першу чергу визначити склад тих операцій, які будуть закладені до вмісту програмних модулів, проаналізувати необхідність і можливість реалізації функцій засобами обраної системи проектування.

Відповідно функціональних можливостей даного веб-сайту, повинен бути реалізований наступний функціонал, що має вміщати:

- можливість реєстрації нових та авторизації наявних користувачів;
- можливість додавати, видаляти, оновлювати, шукати треки;
- можливість надання швидкого відгуку від розроблюваної системи;
- можливість кросс-платформенності;

Структурні особливості веб-сайту можна охарактеризувати наступним чином:

– всі компоненти, які зустрічаються не одноразово, для уникнення повторюваності, повинні бути винесені в один компонент.

– всі компоненти повинні мати єдину структуру;

– клієнтська та серверна частина повинні бути зв’язними;

– повинна бути збережена єдина структура та стиль написання коду.

– Важливим аспектом при розробці, є врахування захисту даних, та їх обробка:

– всі форми повинні бути валідованими для уникнення розбіжності відповідно до типів;

– приватні дані повинні бути відомі та доступні тільки тому користувачеві, який ці дані вводив;

– можливість редагування, та оновлення створених треків повинні бути доступні тільки користувачеві, який їх створив;

– паролі повинні бути захешовані.

Вимоги щодо інтерфейсу розроблюваного веб-сайту повинні враховувати наступні пункти:

– всі розроблені компоненти повинні коректно відображатися на екранах різних розмірів;

– інтерфейс повинен бути максимально зрозумілий, та простий в освоєнні для різних категорій користувачів;

– веб-сайт повинен підтримуватися різними версіями браузерів.

Всі вище наведені вимоги, повинні бути враховані в процесі роботи на створенням веб-сайту.

1.3 Пошук актантів та варіантів використання веб-сайту “Artise”

1.3.1 Пошук актантів

Модель варіантів використання описує функціонал розроблюваної системи. Випадок використання являє собою дискретну одиницю взаємодії між користувачем (людиною або машиною) і системою. Ця взаємодія є єдиною одиницею значущої роботи, як-от створення облікового запису або перегляд даних облікового запису.

Кожен варіант використання описує функціональні можливості, які потрібно реалізувати в розроблювану систему, яка може включати різні функціональні можливості використання або розширювати наявний.

Випадки використання, як правило, пов’язані з «акторами», тобто людьми або машинами, які використовують систему або взаємодіють з нею. Набір варіантів використання, до яких актор має доступ, визначає їх загальну роль у системі. [9]

Виходячи з цього діаграма варіантів використання веб-сайту “Artise”, матиме наступний вигляд (див. рис. 1.1):

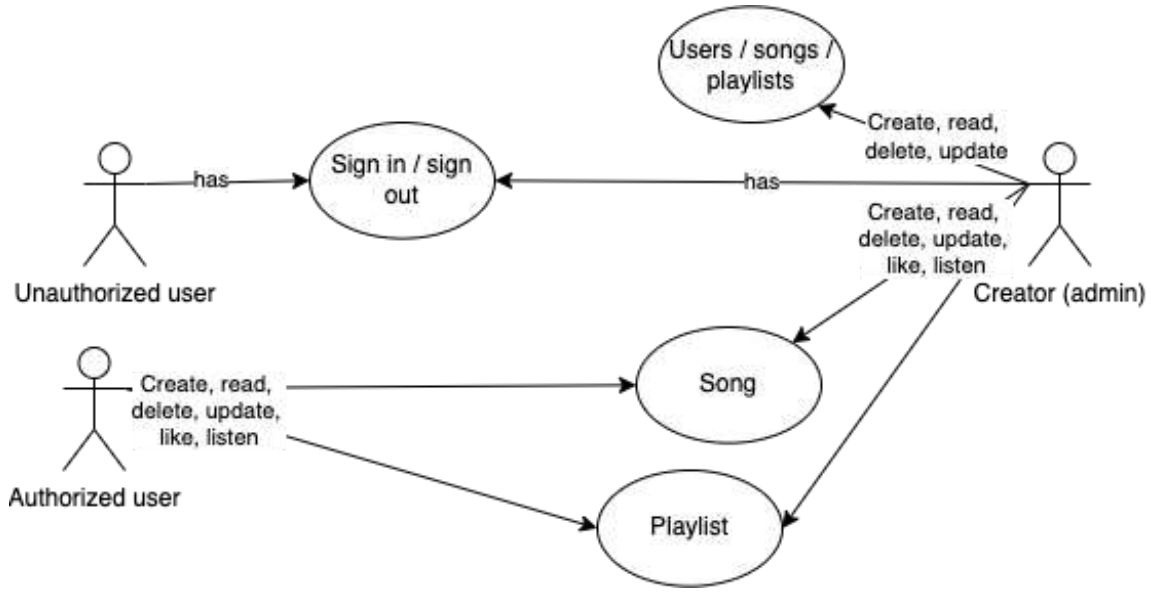

Рисунок 1.1 – діаграма варіантів використання (Use case) веб-сайту “Artise”

Як зрозуміло з рисунку, акторами в даному випадку будуть Creator (Admin), Unauthorized user.

1.3.2 Варіанти використання веб-сайту “Artise”

Виходячи з діаграми варіантів використання розроблювана музична платформа повинна у повній мірі забезпечувати вище зазначений функціонал, Тому з опису основних обов’язків актора можна сформувати ключові варіанти використання системи. Короткий опис варіантів використання наведено в таблиці 1.1.

Таблиця 1.1 – Варіантів використання веб-сайту “Artise”

Актор	Найменування	Формулювання
Unauthorized user	Sign in / sign up	Для можливість доступу використання функціоналу веб-сайту, автентифікації
Authorized user	Перегляд, додавання, редагування, видалення, прослуховування, вподобання треків	Перегляд, додавання, редагування, видалення, прослуховування, вподобання треків власних треків, та перегляд, вподобання, прослуховування, інших користувачів
Creator (admin)	Перегляд, додавання, редагування, видалення, прослуховування, вподобання треків	Перегляд, додавання, редагування, видалення, прослуховування, вподобання треків, в тому числі й інших користувачів

Таким чином, виходячи з табличних даних варіантів використання веб-сайту, можна зрозуміти як кожен з акторів повинен взаємодіяти з системою.

1.4 Опис ключових варіантів використання веб-сайту “Artise”

Кожен з наявних на веб-сайті акторів взаємодіє з системою по різному, за своєю власною схемою та очікує від системи певної поведінки й реакції. Таку поведінку акторів можна відобразити за допомогою UML діаграми.

Отже, для не авторизованого користувача UML діаграма виглядатиме наступним чином:

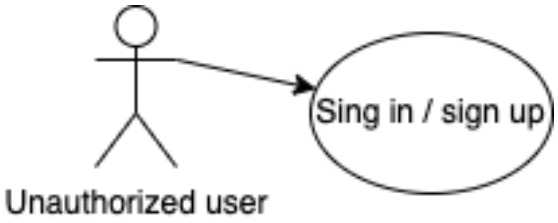

Рисунок 1.2 – діаграма варіантів використання не авторизованого користувача

Авторизувавшись, користувач має значно більше можливостей:

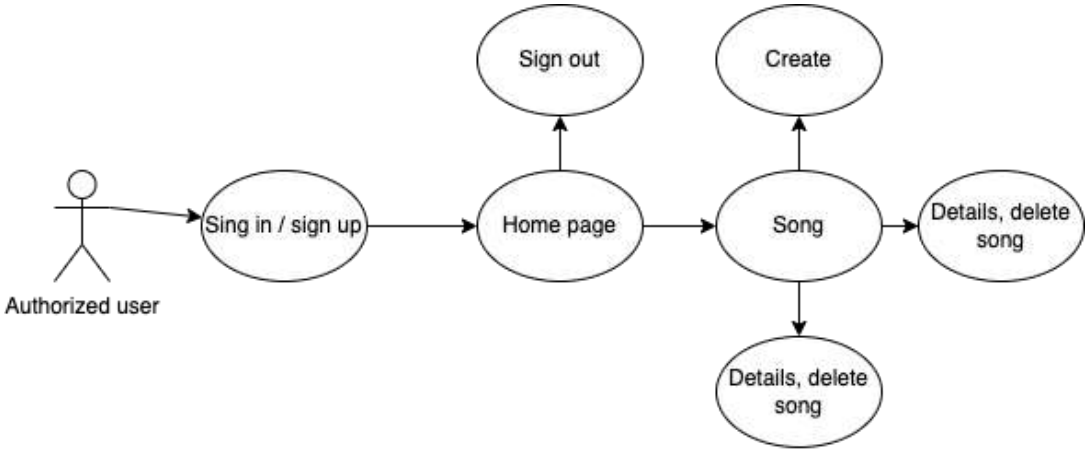

Рисунок 1.3 – діаграма варіантів використання авторизованого користувача

Адміністратор крім можливостей, які має користувач, додатково має можливість керування цими користувачами користувачами:

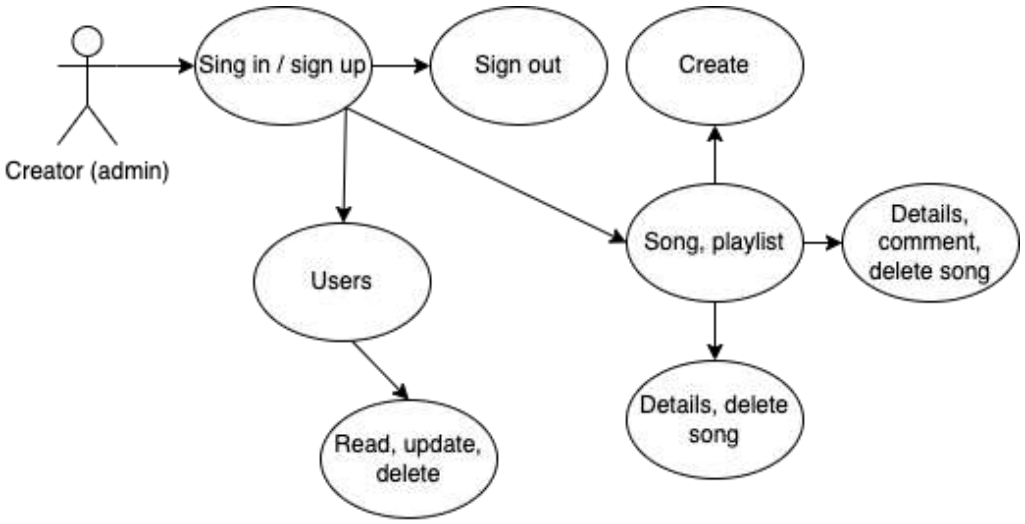

Рисунок 1.4 – діаграма варіантів використання адміністратора

Порівнявши діаграми можна зробити висновок, що не авторизований не має доступу до функціоналу сайту, а адміністратор має більші повноваження ніж користувач, і може керувати записами користувачів, редагувати, видаляти їх облікові записи.

1.5 Вибір середовища розробки веб-сайту “Artise”

Як вже зазначено в темі кваліфікаційної роботи було використано наступний стек технологій: React, Node.js та MongoDB, так званий MERN. Детальніше про кожний інструмент буде описано в підрозділах нижче.

1.5.1 Мова програмування

JavaScript – це крос-платформна, об’єктно-орієнтована скриптова мова, яка є невеликою і легкою. Всередині виконання JavaScript може бути пов’язаний з об'єктами даного середовища і надавати програмний контроль над ними.

JavaScript включає стандартну бібліотеку об'єктів, наприклад Array, Date і Math, а також базовий набір мовних елементів, наприклад, оператори та керуючі конструкції. Ядро JavaScript може бути розширене для різних цілей шляхом додавання нових об’єктів.

Java Script на стороні клієнта розширює ядро мови, надаючи об'єкти для контролю браузера та його Document Object Model (DOM). Наприклад, розширення клієнта дозволяють застосунку розміщувати елементи у формі HTML і обробляти події користувача, такі як клацання миші, введення даних у форму і навігація по сторінках.

JavaScript на стороні сервера розширює ядро мови, надаючи об’єкти для запуску JavaScript на сервері. Наприклад, розширення на стороні сервера дозволяє програмі з'єднуватися з базою даних, забезпечувати безперервність інформації між викликами програми або виконувати маніпуляції над файлами на сервері. [5]

Також варто сказати декілька слів про React та Node.js це по суті інструменти, що дозволяють працювати з JavaScript полегшивши роботу з DOM та розширивши JS у вигляді JSX розмітки зі сторони React та транслювати виклики мовою JavaScript в машинний код з боку Node.js.

1.5.2 База даних

MongoDB реалізує новий підхід до побудови баз даних, де немає таблиць, схем, запитів SQL, зовнішніх ключів та багатьох інших речей, які притаманні об'єктно-реляційним базам даних.

На відміну від реляційних баз даних MongoDB пропонує документо-орієнтовану модель даних, завдяки чому MongoDB працює швидше, має кращу масштабованість, її легше використовувати.

Але, навіть з огляду на всі недоліки традиційних баз даних та переваги MongoDB, важливо розуміти, що завдання бувають різні та методи їх вирішення бувають різні. У якійсь ситуації MongoDB дійсно покращить продуктивність вашої програми, наприклад, якщо треба зберігати складні структури дані. В іншій ситуації краще буде використовувати традиційні реляційні бази даних. Крім того, можна використовувати змішаний підхід: зберігати один тип даних у MongoDB, а інший тип даних – у традиційних БД.

Вся система MongoDB може представляти не лише одну базу даних, що знаходиться на одному фізичному сервері. Функціональність MongoDB дозволяє розташувати кілька баз даних на кількох фізичних серверах, і ці бази даних можуть легко обмінюватися даними та зберігати цілісність. [15]

1.6 Обґрунтування використовуваних технологій розробки веб-сайту “Artise”

MERN – це популярний стек технологій з відкритим вихідним кодом для створення одно сторінкових веб-сайтів так званих SPA додатків [16] комбінація MongoDB, Express, React та Node.js. За допомогою цього технологічного стеку можна створити високоефективну наскрізну структуру, розробляючи веб-сайт швидше та простіше.

MERN – це:

– бібліотеки з відкритим вихідним кодом;

– односпрямований потік даних;

– підтримка мобільних додатків;

– швидкий рендеринг компонентів;

– повна та зрозуміла документація.

1.7 Висновок до першого розділу

Отже, в першому розділі кваліфікаційної роботи було проведено аналіз предметної області веб-сайту “Artise”, сформовано вимоги до: функціональних можливостей, структури веб-сайту, захисту та доступу. Було проаналізовано ключові варіанти використання веб-сайту, розроблено UML діаграми взаємодії кожного актора з системою. Також обґрунтовано вибір технологій, якими розроблятиметься веб-сайт, детальніше, щодо обраних технологій буде описано в другому розділі кваліфікаційної.

2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБ-САЙТУ “ARTISE”

2.1 Архітектура веб-сайту “Artise”

Як вже було зазначено вище, для розробки додатку було обрано MERN стек, тобто MongoDB, Express, React, Node.js. Основна перевага такого підходу розробки веб-сайту, полягає в тому, що кожен рядок коду написаний на JavaScript чи в кращому випадку на Typescript. Використовуючи єдину мову програмування, стек MERN усуває необхідність перемикання між мовами наприклад між JS та Python, що в певній мірі спрощує як і сам процес розробки, так і усуває необхідність пошуку залежностей, які б могли функціонал цих мов поєднати, чи якось логічно пов’язати.

За допомогою MERN стеку можна без зайвих зусиль побудувати три-рівневу архітектуру (front-end, back-end, database). Архітектура веб-сайту “Artise” показана на рисунку 2.1 нижче.

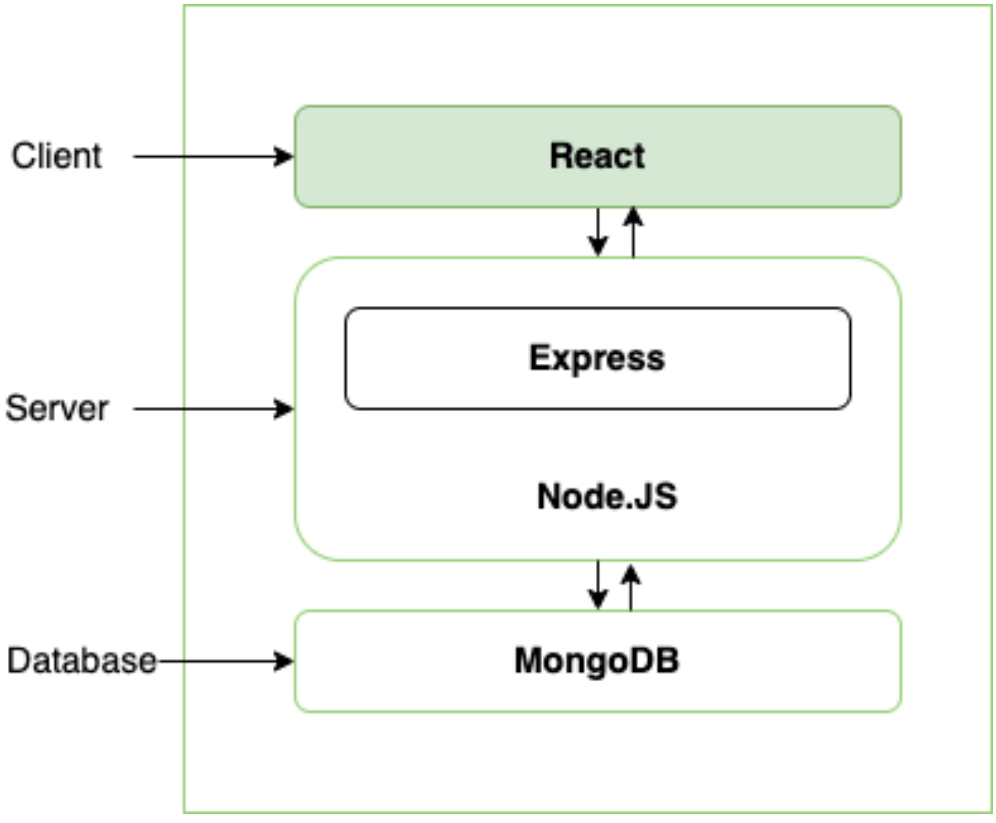

Рисунок 2.1 – Трирівнева архітектура веб-сайту “Artise”

Варто відзначити те, що також ще однією перевагою використання MERN, є те, що не має значення в якому напрямку цю архітектуру відтворювати, тобто можна спочатку повністю написати серверну частину, підключити її до БД [3], і вже тоді до клієнта, чи наприклад розробити повністю клієнтську, згенерувати псевдо дані, що відповідатимуть структурі даних, яку повертатиме сервер, а потім власне приступити вже до безпосередньої його реалізації та підключення до бази даних.

2.2 Вибір основних залежностей для розробки веб-сайту “Artise”

Після того як вже архітектура розроблюваної системи повністю спроектована та готова до реалізації, виникає питання вибору залежностей (dependencies) розроблюваної системи. Важливо, щоб цей список був максимально короткий, оскільки нативна реалізація, завжди швидше, краще, гнучкіше. Список залежностей будь якої React розроблюваної системи можна знайти в файлі проекту під назвою package.json. Тут залежності поділяються на два основних типи: devDependencies та dependencies. Перші слугують як інструментарій що допомагає під час розробки, наприклад підсвічує певний синтаксис, другий – для реалізації певної функціональності. Залежності в даному проекті присутні як при розробці серверної частини так і клієнтської.

Пакети (назва залежностей за межами package.json) можуть бути реалізовані як і звичайним розробником так і великими корпораціями по типу Microsoft. На це важливо звертати увагу оскільки, якщо пакет розроблений розробником, орієнтований на сумісність старішої версії React чи Node.js, і не користується попитом, то скоріше за все розробник не буде в подальшому підтримувати пакет і вже в новіших версіях може бути просто не сумісний з новим функціоналом, який був доданий до бібліотеки чи платформи.

Той чи інший пакет можна знайти на двох великих пакетних менеджерах, у випадку JS орієнтованих пакетів, це npm.js та yarnpkg. Також в ці самі пакети,

цими пакетниками і встановлюються, у випадку npm це npm install <package name>, yarn – yarn add <package name>.

Список залежностей клієнтської частини показаний на рисунку 2.2 нижче:

Рисунок 2.2 – Список залежностей front-end частини веб-сайту “Artise”

Як вже було сказано вище на back-end’і структура та встановлення ідентичні, оскільки ми використовуємо єдину мову програмування під капотом – JS.

Рисунок 2.3 – Список залежностей back-end частини веб-сайту “Artise”

Варто відзначити основні з залежностей, які були використані при розробці.

Front-end:

– chakra-ui: одним з важливих завдань є вибір дизайн системи (Design System). Звичайно можна розробити власну систему, але це досить трудомістко, оскільки існують вже готові рішення та очевидним лідером серед них на сьогоднішній день є Chakra UI, що прийшла на заміну вже відомому Material UI, який на даному етапі розвивається не дуже активно. Chakra UI пропонує хороший вибір базових безкоштовних елементів, а також складні платні темплейти. За допомогою цих темплейтів можна розробити дизайн E-Commerce сайту за лічені дні, не вдаючись до Figma або аналогічних сервісів [14];

– tailwind: CSS-бібліотека, яка спрощує стилізацію HTML, так само, як це робить Chakra, – додаючи величезну кількість різноманітних класів. Але, на відміну від Chakra, що додає стилі до вже стилізовані по замовчуванню компонентів, такі як кнопки, алерти та навбари, класи Tailwind CSS за замовчуванням цих стилів не має та забезпечує додатковий функціонал – директиви. [8];

– formik: для контролю валідності даних, які вводять користувачі в полях;

– apollo-client: бібліотека керування станом для JavaScript, яка дозволяє керувати локальними та віддаленими даними за допомогою GraphQL. Використовується для отримання, кешування та зміни даних, автоматично оновлюючи інтерфейс користувача. [4];

– axios: HTTP-клієнт на основі Promise для браузера та node.js;

– Faker: бібліотека, для генерації для “мокових” даних, для тестування, розробки тощо. [2].

Back-end:

– apollo-server-express: для розгортання Apollo graphql сервера та можливість його підключення до Express.

– bcrypt: для хешування паролів по принципу bcrypt;

– jsonwebtoken: для реалізації авторизації на основі JSON Web token;

– mongodb: для підключення до MongoDB, маніпуляцій з DB.

Також важливо дотримуватися єдиного стилю написання коду, адже у випадку, якщо розроблюваний додаток, веб-сайт чи платформа мають відкритий вихідний код, то тоді цей код повинен розуміти не лише розробник, який цей код писав, а й інші, за для внесення в майбутньому можливих коректив. При розробці веб-сайту “Artise”, з ціллю дотримання єдиної стилізації коду було використано Prettier.

Для виявлення можливих синтаксичних помилок – Eslint.

2.3 Файлова структура веб-сайту “Artise”

Як вже було сказано у підрозділі вище, про дотримання єдиного стилю написання коду, такому ж самому принципу варто слідувати під час структуризації компонентів. Це дозволяє забезпечити зрозумілість, масштабованість, не повторюваність.

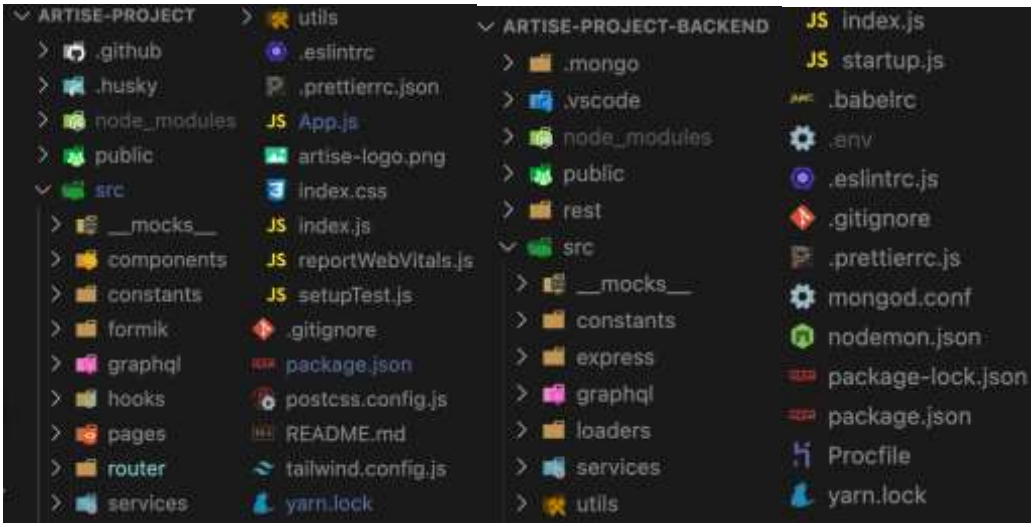

Рисунок 2.4 – Файлова структура веб-сайту “Artise”

Структура веб-сайту “Artise” це по суті дерево, в якій кореневі вузли залишаються незмінними. Кореневими вузлами є макет програми. Далі в кожній з папок відбувається розгалуження (вітки), в цих папках містяться інші папки чи файли (інші вітки) і т. д.

– artise-project / artise-project-backend – назва клієнтської / серверної частин проєкту;

– node_modules – містить список встановлених залежностей;

– public – основна інформація про проєкт, потрібна в основному для оптимізації пошукових двигунів (SEO) (у випадку front-end);

– src – для реалізації функціоналу.

– src.mocks__ – псевдо дані, які можуть бути використані у разі потреби даних, якщо на сервері вони ще не реалізовані чи просто для наповнення контентом;

– src.components – компоненти які можуть бути пере використані безліч разів;

– src.constants – константи, що не одноразово повторюються та можуть бути винесені;

– src.formik – компоненти для форми з валідацією вмісту;

– src.hooks – користувацькі хуки;

– pages – сторінки веб-сайту;

– services – сервіси, такі як наприклад авторизація чи API.

2.4 Структура та особливості використаної БД для розробки веб-сайту “Artise”

Якщо в реляційних БД вміст складають таблиці, то в MongoDB – колекції. Кожна колекція має своє унікальне ім’я – довільний ідентифікатор, що складається з не більше ніж 128 різних алфавітно-цифрових символів та символу нижнього підкреслення. На відміну від реляційних баз даних MongoDB не використовує табличний пристрій із чітко заданою кількістю стовпців та типів даних. MongoDB є документо-орієнтованою системою, в якій центральним поняттям є документ. Документ можна представити як об’єкт, який зберігає певну інформацію. Документ представляє набір пар ключ-значення.

СКБД використовують для зберігання подій у системі (логування), запису інформації з датчиків моніторингу на підприємстві, а також у сфері електронної комерції та мобільних додатків. Часто MongoDB застосовують як сховище у сфері машинного навчання та штучного інтелекту. [6]

MongoDB – це кроссплатфменний продукт, який легко впровадити у будь-яку операційну систему. Ряд унікальних особливостей дозволяє використовувати СКБД під певні завдання, у яких забезпечує максимальну продуктивність і надійність.

Схему бази даних веб-сайту “Artise” можна відобразити наступним чином (див рис. 2.4):

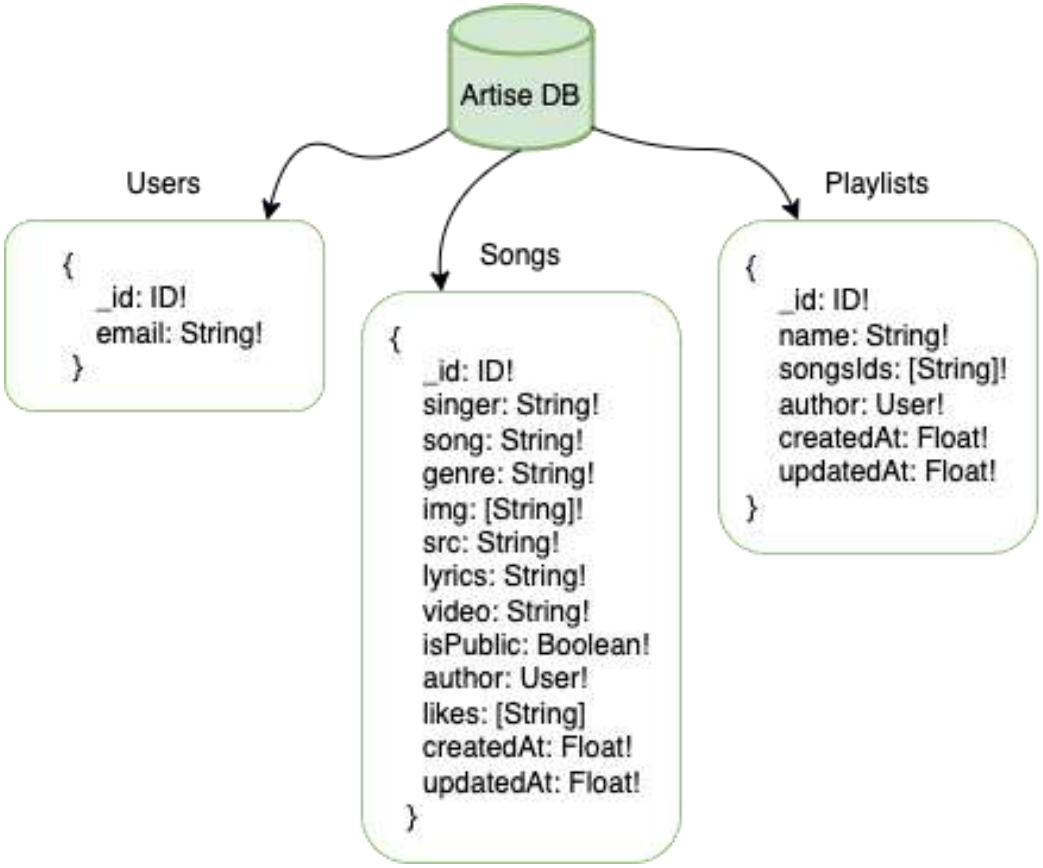

Рисунок 2.5 – Схема бази даних веб-сайту “Artise”

Документи в даній системі будуть: Users – містить інформацію про користувачів, Songs – про музичні записи (треки), Playlists – плейлисти. З таким

форматом (JSO) дуже зручно працювати, адже вибравши стек на основі Java Script, він зустрічається на всьому шляху розробки проєкту.

2.5 Розгортання веб-сайту “Artise” на хостингових платформах

Оскільки даний застосунок має розділену структуру, окремо серверну частину, окремо клієнтську то необхідно розгорнути даний стек відповідно роздільно. Проаналізувавши ринок рішень хостингових платформ, було обрано Heroku для серверної частини та Netlify для клієнтської. Саме в такий спосіб найкраще розгортати MERN.

Netlify та Heroku – хмарні платформи, розроблені для простого поширення локальних рішень в інтернеті. Netlify спеціалізується на статичних сайтах, тобто на яких гіпертекстова розмітка сайту, включаючи його вміст, попередньо згенерована, у той час як Heroku на динамічних, тобто такі, які генеруються під час запиту. [1] Детальніше про дані хостингові про кожну з платформ та їх відмінності буде описано нижче.

Netlify — це рішення, яке надає інструменти безперервної інтеграції, а також повністю вбудовану службу CDN. Сервіс повністю орієнтований на front-end.

Heroku — це більш загальна платформа, яка дозволяє використовувати кілька типів розгортання (включаючи розгортання веб-сайтів) під значною кількістю мов програмування (Java, Node.js, Scala, Clojure, Python, PHP і Go). Сервіс орієнтований на back-end.

Вибір між Netlify і Heroku залежить від кількох факторів, таких як:

– Тип проекту, що розробляється

– Потрібний рівень гнучкості

– Бюджет

Основні відмінності між платформами наведені в таблиці 2.1.

Таблиця 2.1 – Основні відмінності між Netlify та Heroku

	Netlify	**Heroku**
Опис	Розгорніть сучасний веб-сайт одним кліком	Створюйте, запускайте та керуйте додатками повністю в хмарі
Дата заснування	2014 рік	2007 рік
Материнська компанія	Відсутня	SalesForce
Кількість працівників	153	160
Гнучкість	Низька	Висока
Ціна	Фіксована ціна	Оплата за використання
Масштабованість	Присутня	Присутня
Ідеально підходить для	Сучасної автоматизації веб-сайтів	Створення хмарних рішень, в тому числі БД
Постійна інтеграція	З коробки	Можливо увімкнути
Фокус	Front-end	Back-end
Тип проекту	Web	Web + mobile
Зауваження	Немає режиму сну	Наявний режим сну

Базу даних веб-сайту було розміщено на MongoDB Atlas, який надається MongoDB.

Коротко кажучи MongoDB Atlas – це глобальний хмарний сервіс баз даних.

За допомогою MongoDB Atlas можна розмістити керовану базу даних MongoDB на таких гігантах як AWS, Azure або GCP. MongoDB Atlas дозволяє суттєво зменшити затрати на створення, налаштування, управління.

Налаштуванням БД та її розміщенням безпосередньо займається сам сервіс, від користувача потрібні тільки дані. В даному випадку користувачем є розроблений back-end, але це не виключає можливості вводу даних в ручну та маніпуляцію ними.

Реалізація. Для розгортання на Netlify можна просто закинути папку з проєктом, і якщо ж ніяких помилок при деплої не випливе, то по суті веб-сайт вже запрацює, але якщо говорити про маштабованість то в такому випадку вона близька нулю через те, що при випуску нової версії необхідно буде вручну завантажувати внесені зміни. Можна скористатися методом підключення з хмарних сховищ, таких як наприклад Github, в даному випадку було обрано саме цей метод. У разі успішного деплою сайт готовий до використання. Розміщений веб-сайт “Artise” показано на рисунку 2.6.

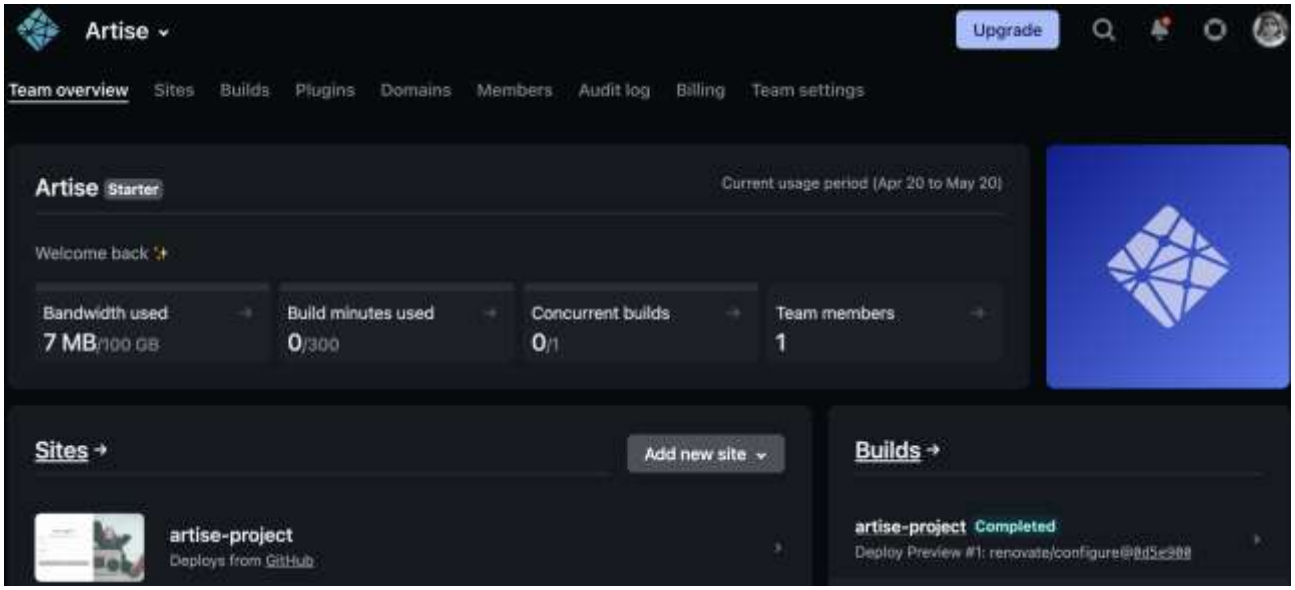

Рисунок 2.6 – Розміщення front-end на Netlify

Також крутим доповненням Netlify є те, що можна безкоштовно змінити доменне ім’я, що в даному випадку і було зроблено. [7]

Back-End також було розміщено на Github. Heroku, як і власне Netlify, дозволяє скористатися підключення проєкту з хмарних сервісів, тому вибір вочевидь зрозумілий. Підключивши та ініціалізувавши Heroku, в папці з проєктом налаштуємо redirect при переході на відповідні рути:

```
/api/*    https://artise.herokuapp.com/api/:splat   200
/graphql/*    https://artise.herokuapp.com/graphql    200
/*    /index.html   200
```

На цьому підключення back-end частини завершено. Розміщення серверної частини на хостингу наведено на рисунку 2.7.

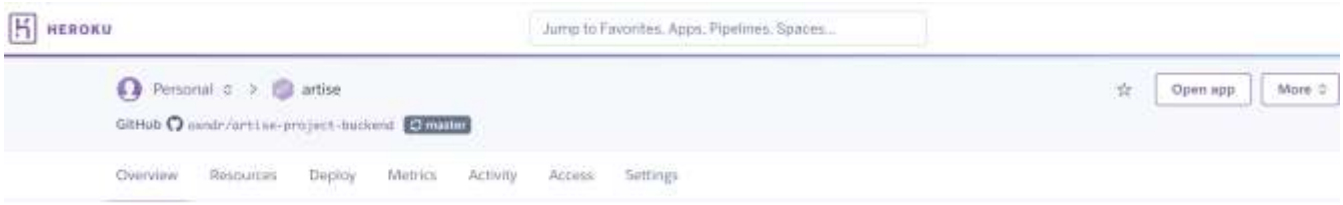

Рисунок 2.7 – Розміщення back-end на Heroku

Також підключимо MongoDB до MongoDB Atlas задеплоївши кластер та авторизувавшись через командну стрічку до Heroku. Розміщення бази даних на MongoDB Atlas наведено на рисунку 2.8.

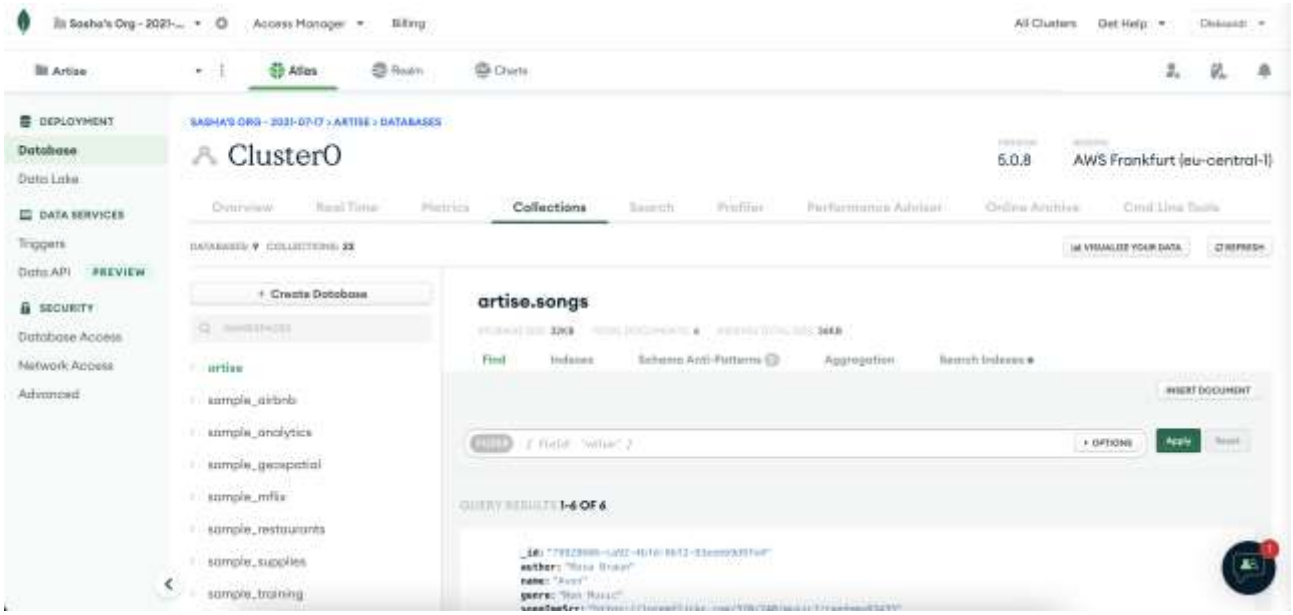

Рисунок 2.7 – Розміщення бази даних на MongoDB Atlas

В результаті отримуємо розміщений в мережі інтернет веб-сайт, який повністю функціонує не залежно від локального хостингу, отримує дані отримані від бази даних з хмари, які успішно зберігаються та оновлюються.

2.6 Тестування та використання веб-сайту “Artise”

2.6.1 Тестування веб-сайту “Artise”

Тестування веб-сайту це дуже обширне поняття оскільки можна проводити тестування функціоналу з написанням різних тестів, таких як наприклад Jest, що стане у нагоді великим проєктам, оскільки над такими проєктами працює ціла команда розрбників, а не одна людина, що може призвести до не очікуваної поведінки після того як один розробник, змінив логіку тієї чи іншої частини коду, яку задумав інший розробник і т. д.

В даному випадку було проведено ручне тестування, тобто перевірка коректності роботи задуманого функціоналу, за допомогою Lighthouse – вбудований в Chrome інструмент для покращення якості веб-сайту. Результати тесту показані на рисунку 2.8.

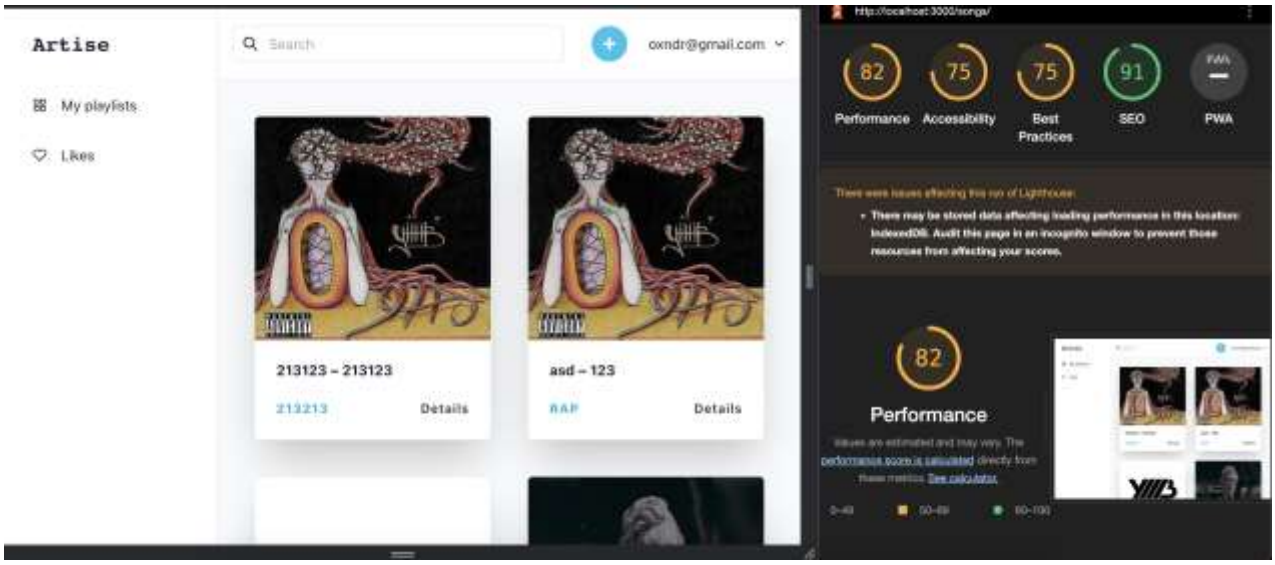

Рисунок 2.8 – Результати тестування Lighthouse

Тут метрики показують досить не поганий результат для SEO і Performance як для локального хостингу, Accessibility та Best Practises потребують уваги.

Також було протестовано коректність верстки на HTML валідаторі, в результаті якого не було виявлено жодної помилки. Результат тесту верстки показаний на рисунку 2.9.

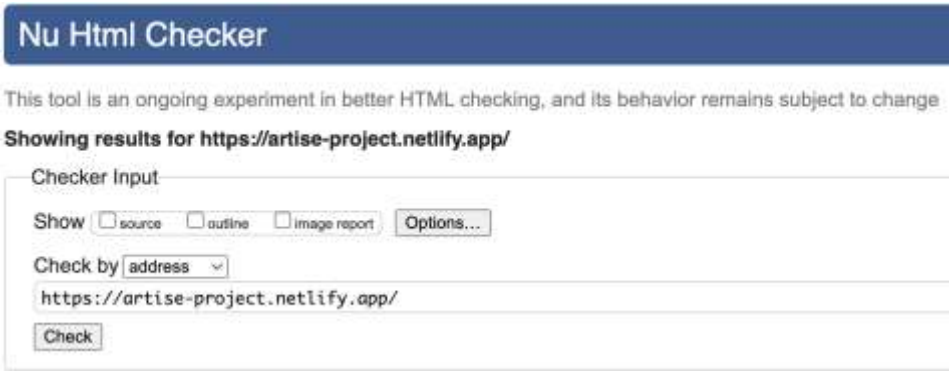

Рисунок 2.9 – Результати тестування верстки валідатором

Також в процесі тестування було виявлено, некоректне відображення на мобільних пристроях малого розміру, цю помилку було успішно усунено. На етапі розробки було виявлено не коректну роботу плеєра, довелося замінити бібліотеку на іншу та змінити логіку роботи плеєра.

2.6.2 Використання веб-сайту “Artise”

Перейшовши на сайт перше, що можна побачити – це сторінка логіну, тут маючи власний акаунт можна авторизуватися, і увійти до веб-сайту, чи натиснути на SignUp і зареєструватися. Сторінка логіну наведена на рисунку 3.

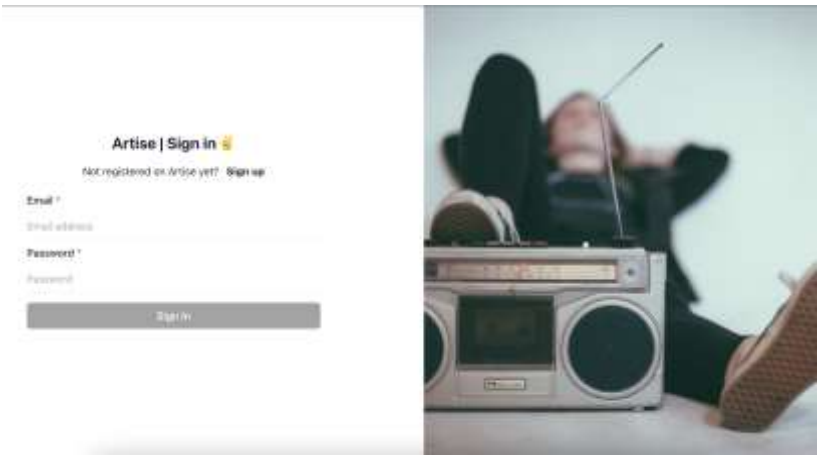

Рисунок 3 – Сторінка логіну

Після успішної авторизації, відбудеться перенаправлення на домашню сторінку. Тут в списку можна побачити власні треки, та треки інших користувачів. Власні треки можна редагувати, видаляти, вподобати, оновлювати, треки інших користувачів видаляти, оновлювати, та редагувати не можна. На рисунку 3.1 можна побачити домашню сторінку веб-сайту “Artise”.

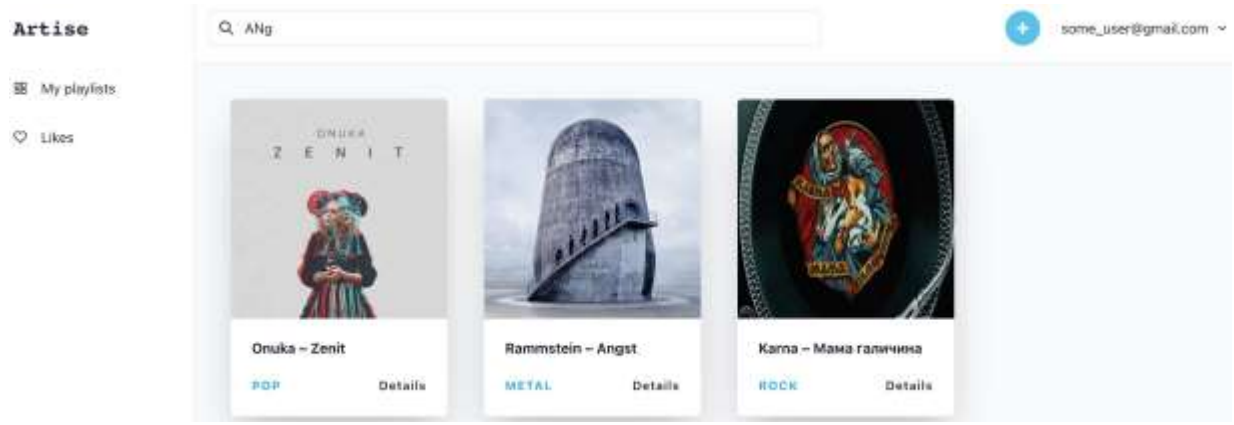

Рисунок 3.1 – Домашня сторінка

Натиснувши на деталі можна побачити більше інформації по треку, таку як більше фото, відео до треку чи текст треку, варто відзначити, що відео і текст це не обов’язкові поля тому ці дані можуть бути відсутні. На рисунку 3.2 наведено деталі певного треку.

Рисунок 3.2 – Деталі треку

Якщо навести курсор миші на аватарку треку (чи у випадку мобільної версії натиснути), то можна скористатися функціями редагування (якщо автором цього треку є авторизований користувач), вподобання, програвання треку. Можливі функції певного треку наведені на рисунку 3.3.

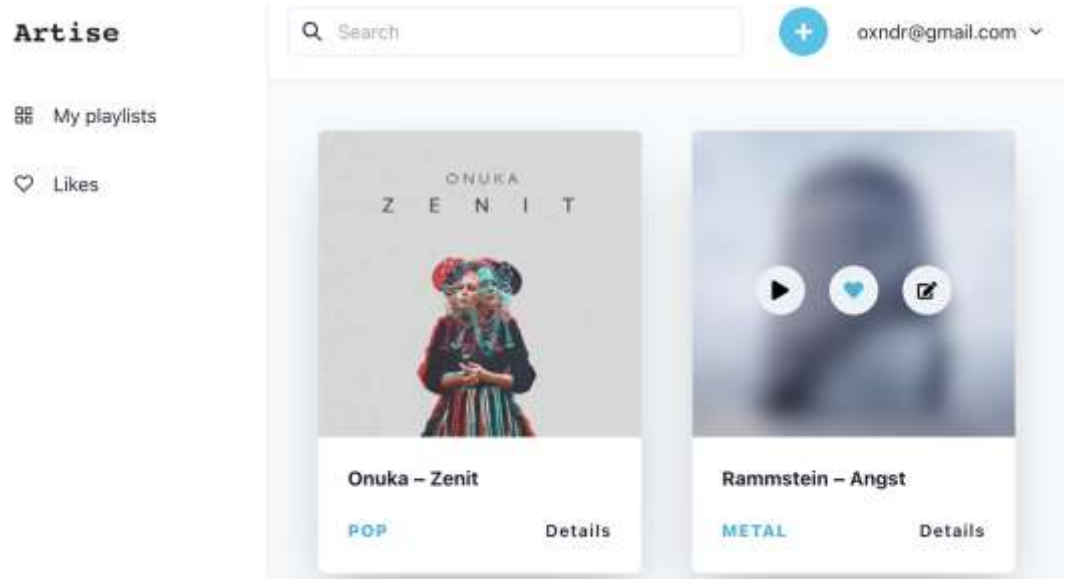

Рисунок 3.3 – Функції треку

За необхідності швидко знайти трек, необхідно в пошуку ввести ім’я треку, виконавця чи жанр. Результати роботи пошуку наведені на рисунку 3.4. Якщо натиснути на певний знайдений результат пошуку, можна перейти на деталі цього треку.

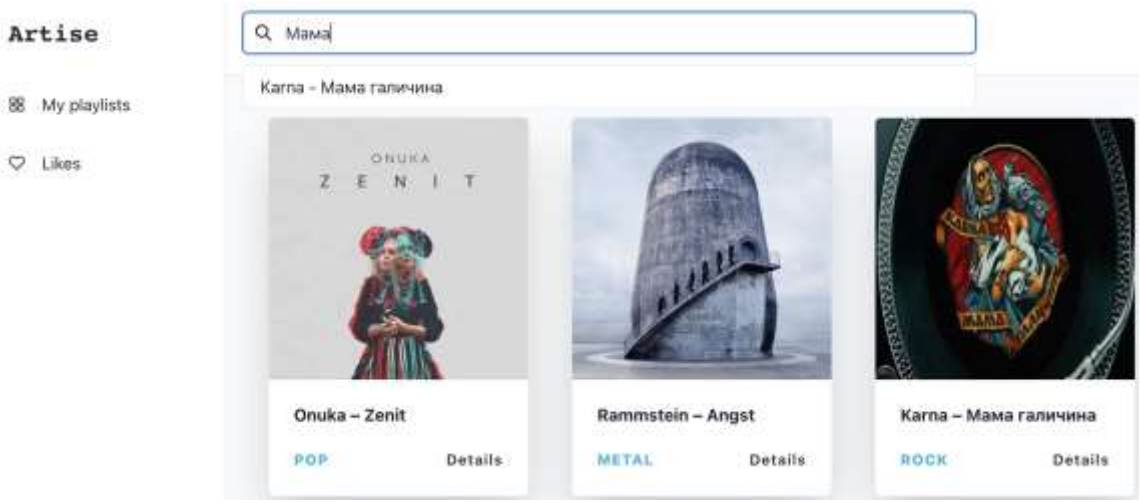

Рисунок 3.4 – Результати роботи пошуку

Якщо натиснути на кнопку “+”, то можна скористатися функціями додавання треку чи плейлиста. Наявні функції при натисканні на кнопку “+” наведені на рисунку 3.5.

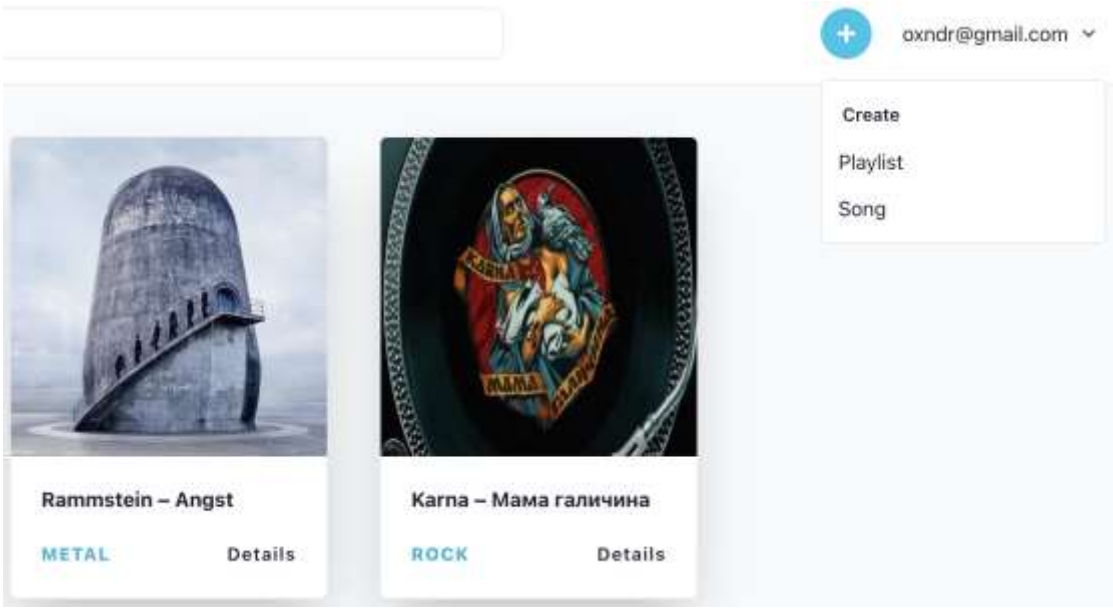

Рисунок 3.5 – Функції при натисканні на кнопку “+”

Список вподобаних треків, можна побачити на сторінці “Likes”, перейшовши на неї по натисканню на “Likes”, в боковій панелі розміщеній зліва. Список вподобаних треків можна побачити на рисунку 3.6.

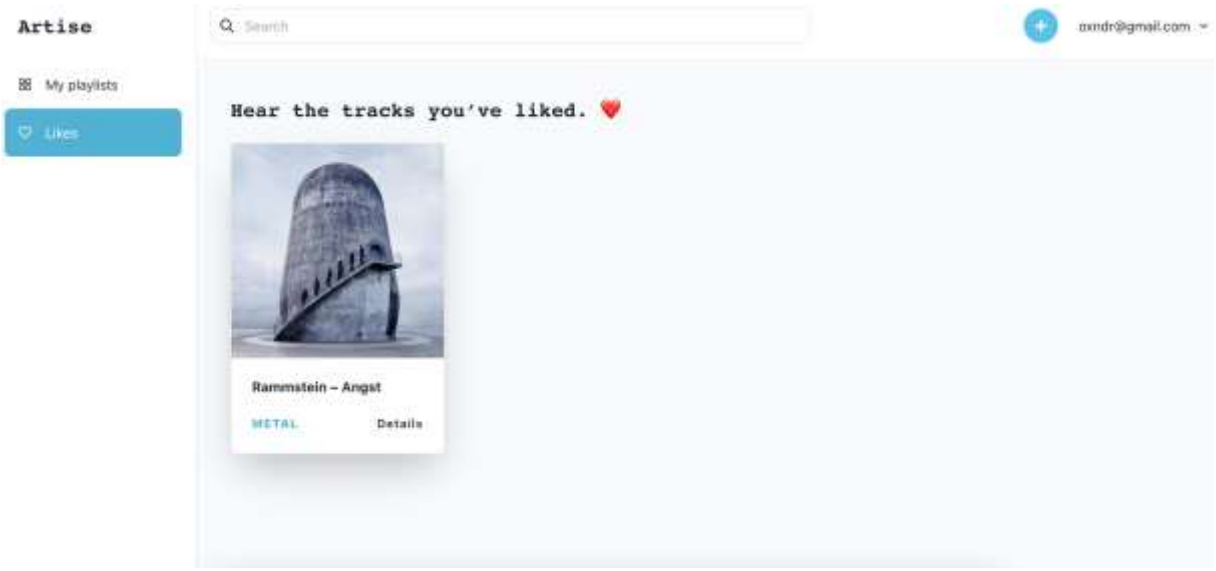

Рисунок 3.6 – Список вподобаних треків

Після створення плейлист(а/ів) їх можна знайти на сторінці “My playlists”. А натиснувши на заголовок плейлиста з’являться функціональні кнопки для прослуховування, редагування чи видалення плейлиста. Сторінка “My playlists” показана на рисунку 3.7.

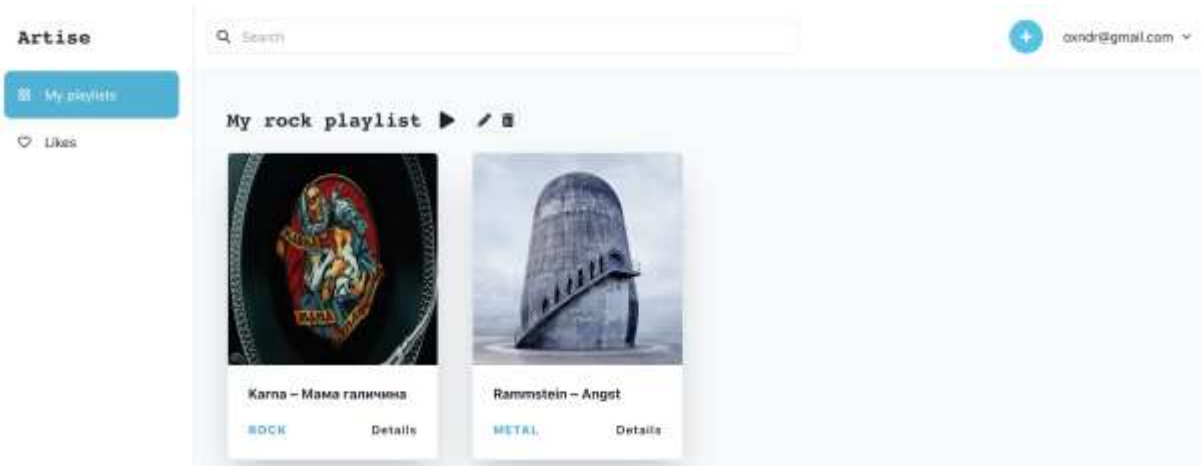

Рисунок 3.7 – Сторінка “My playlists”

Також на сайті всі компоненти було зверстано з можливістю підтримки темної теми, для зручного перегляду веб сайту ввечері, чи вночі, або ж взагалі на постійній основі. Результати використання темної теми наведені на рисунку 3.8.

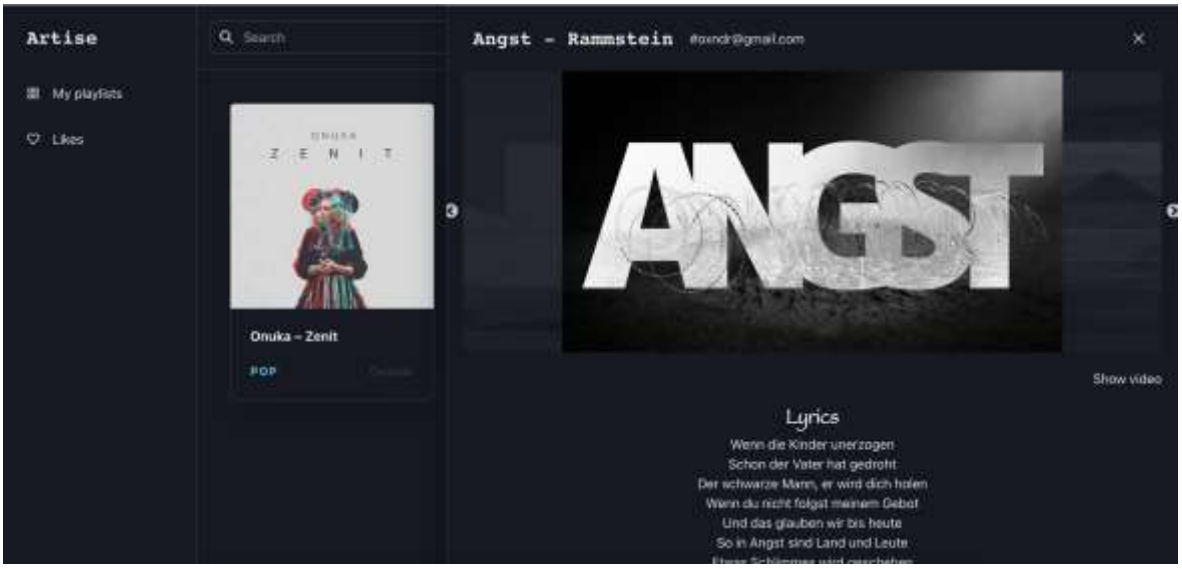

Рисунок 3.8 – Темна тема веб-сайту

Ця функція індивідуальна для кожного користувача. І не залежить від включення, виключення інших користувачів.

Також, запланована крос-платформенність, тобто підтримка на пристроях різного типу, таких як мобільні пристрої, планшети, ноутбуки, тощо, була успішно реалізована, впродовж імплементації верстки сайту. Результат відображення головної сторінки веб-сайту “Artise” на мобільному пристрої показано на рисунку 3.9.

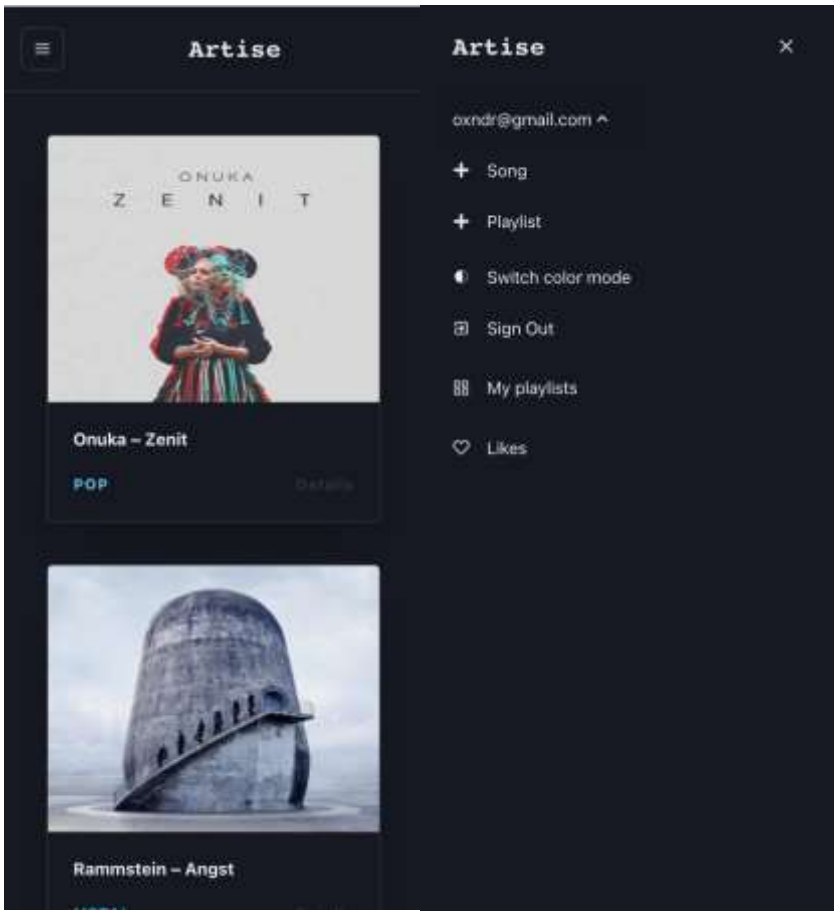

Рисунок 3.9 – Відображення головної сторінки веб-сайту “Artise” на мобільному пристрої Google Pixel 5

Як вже було сказано в першому розділі, це дозволить, підвищити цільову аудиторію, оскільки мобільними пристроями користуються значно більше користувачів. Також зі зростанням попиту можливо повністю нативно розробити веб-сайт за допомогою React Native чи інших технологій.

2.6 Висновки до другого розділу

В другому розділі кваліфікаційної роботи, було описано та обґрунтовано архітектуру розроблюваного веб-сайту “Artise”. Після того як вже архітектура системи була спроектована та готова до реалізації, було описано вибір основних залежностей для розробки. Вказано файлову структуру веб-сайту “Artise”, та важливість дотримання єдиного стилю написання коду, на етапі розробки. Детальніше описано структуру та особливості використаної бази даних, її переваги перед SQL базам, проведено аналіз між обраними хостинговими платформами, виділено переваги та недоліки кожної, і в результаті розміщено веб-сайт на хостингу. В кінці проведено тестування розробленого веб-сайту, та описано його використання.

3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ХОРОНИ ПРАЦІ

3.1 Методи боротьби з монотонністю праці на виробництві

Реалізація творчих здібностей особистості, підвищення мотивації до праці за рахунок так званого «збагачення» праці набувають дедалі більшого значення в розвитку виробництва на сучасному етапі.

Обґрунтування системи заходів щодо запобігання монотонності і її негативних наслідків базується на вченні І. П. Павлова і І. М. Сєченова про необхідність розширення поля коркової активності в процесі праці і виключення «довбання» в одну клітину.

Найрадикальнішим заходом є проектування раціональних трудових процесів і операцій на основі оптимального поділу праці. Завдання полягає в тому, щоб кожну операцію зробити змістовною, яка сприяла б розвитку у працівника творчого мислення. Основним принципом проектування раціонального трудового процесу (операції) є принцип збереження певної логічної завершеності і структурної цілісності виконуваної операції. Навіть в умовах глибокої диференціації технологічного процесу необхідно встановлювати таку кількість елементів операції і послідовність їх виконання, яка сприймалася б працівником як логічно завершена одиниця.

Другим важливим принципом проектування трудового процесу є забезпечення достатнього енергетичного рівня операції. Спеціальними дослідженнями встановлено, що негативні психічні стани більшою мірою виявляються при виконанні тих робіт, які через незначну енергетичну вартість не стимулюють функціональної активності організму. Якщо монотонна робота досить інтенсивна за затратами енергії, то нудьга, сонливість, психічне перенасичення можуть не виникати. Доведено, що при фізичній роботі для підтримання активного тонусу кори затрати енергії не повинні бути меншими за 2,5 ккал/хв (150 ккал/год).

Запобіганню монотонності і підвищенню змістовності праці сприяє укрупнення трудових операцій. Завдяки укрупненню операцій у працівника формується більш складний стереотип трудових дій, що позитивно позначається на стані психофізіологічних функцій. Досвід показує, що операція повинна складатися не менш як з 5—6 елементів за умови збереження цільового змісту.

Важливим засобом боротьби з монотонністю є чергування операцій, кожна з яких є монотонною. Науковою основою чергування операцій є ефект Сєченова, суть якого в тому, що при зміні діяльності активізується інша група нервових центрів, а в раніше працюючих ефективно відбувається «заправка» енергією. Отже, принцип чергування операцій полягає в заміщенні і компенсації психофізіологічних функцій, активізації інших м'язових груп, нервових центрів, зменшенні надмірного напруження працюючих м'язів. Значення чергування операцій, таким чином, полягає в ліквідації негативного впливу однобоких навантажень. На практиці застосовується декілька варіантів чергування операцій: через кожну годину, через 2,5 год, один раз протягом зміни, через день. Відносно зняття фактора монотонності найбільш ефективне чергування операцій один раз протягом зміни, хоча в конкретних виробничих умовах це питання вирішується по-різному. Враховуються умови праці, структура операцій, майстерність працівників.

Чергування операцій пов'язане із суміщенням професій і трудових функцій. Зазначимо, що оволодіння працівником другими і суміжними професіями, крім подолання монотонності і підвищення привабливості праці, підвищує конкурентоспроможність працівника на ринку праці і мобільність на самому підприємстві.

Для зняття монотонності необхідно, щоб операції відрізнялися за характером навантажень, але в той же час були позбавлені інтерферентних елементів.

Основні умови суміщення професій і трудових функцій, які забезпечують зменшення монотонності:

– суміщувані професії повинні змінювати рівень завантаженості різних органів і систем;

– суміщувана операція повинна бути легшою, ніж основна. При легкій монотонній роботі ефективна зміна на більш важку;

– більш монотонну роботу необхідно суміщувати з менш монотонною;

– суміщувані трудові комплекси повинні забезпечувати роботу за участю м’язів-антагоністів, а також зміну робочих поз;

– статичні навантаження повинні компенсуватися помірними динамічними навантаженнями.

При організації монотонних робіт важливе значення має вибір темпу роботи. Темп може бути вільним або примусовим. Кожний з них має переваги і недоліки. Тому при виборі темпу роботи слід виходити зі специфіки конкретного виробництва. В одних випадках доцільним є оптимальний заданий темп з регулюванням швидкості конвеєра відповідно до кривої працездатності. Варіація швидкості не повинна перевищувати 10—15 %. В інших випадках ефективне самостійне регулювання робочого темпу. Останнє застосовується на автономних конвеєрах, що забезпечує не лише свободу ритму, а й регулювання змісту роботи.

Ефективним засобом боротьби з монотонністю є бригадно-групова форма організації потоку. Суть її в тому, що бригада виконує операції всього циклу по виготовленню більш-менш закінченого продукту (вузла). Процеси виготовлення кожного вузла виділяються в самостійні виробничі секції. Робітники працюють у вільному ритмі, а вузли з’єднуються в монтажній секції. В цьому випадку трудовий процес менше розчленований і тісніше кооперований.

Зменшенню негативного впливу монотонних робіт на психічний стан працівників і показники їхньої праці сприяють такі заходи:

– раціоналізація режимів праці і відпочинку;

– естетизація виробничого середовища;

– застосування функціональної музики.

До факторів зменшення монотонності відносяться також психологічні заходи, покликані посилити внутрішні мотиви діяльності. Це, зокрема, психологічна стимуляція трудової діяльності за рахунок постановки проміжних виробничих цілей, забезпечення працівників поточною інформацією щодо виконання роботи. Особливе значення мають залучення робітників до управління і розв’язання виробничих проблем, а також сприятливий соціально-психологічний клімат, створення умов для спілкування в процесі праці, якщо це можливо. Усе це формує позитивні емоційні стани у працівників, посилює їх монотоностійкість. [13]

3.2 Загальні вимоги безпеки з охорони праці для користувачів ПК

В сьогоднішніх реаліях важко уявити своє життя без використання персонального комп’ютера, особливо з приходом у світ пандемії COVID-19, все більше і більше підприємств вже перейшли, і надалі продовжують переходити в віддалений режим трудових відносин. Для деяких сфер, таких як наприклад IT, це вже незворотній процес, адже це дуже зручно, і гнучко, оскільки можна надавати послуги з будь якої точки земного шару. Важливо при цьому не забувати про безпеку з охорони праці для користувачів ПК. В основному вимоги безпеки можна поділити на три пункти, а саме: вимоги безпеки перед початком роботи за ПК, вимоги безпеки під час роботи за ПК, і вимоги безпеки після закінчення роботи за ПК.

Перед початком роботи за ПК слід:

– привести в порядок робоче місце і впевнитися, що на ньому відсутні сторонні предмети та всі пристрої і блоки ПК під'єднані до системного блоку.

– перевірити наявність та надійність захисного заземлення устаткування;

– перевірити справність вимикачів та інших органів управління ПК;

– перевірити справність роз’ємів кабелів електроживлення;

– перевірити відсутність пошкоджень ізоляції проводів живлення;

– перевірити відсутність відкритих струмопровідних частин у пристроях ПК.

– відрегулювати сидіння робочого стільця (крісла);

– кут нахилу спинки стільця в межах 90—220° до площини сидіння;

– кут зору на екрані монітора - складав 25°

– відстань до екрана — 600—800 мм;

– вжити заходів, щоб пряме світло не потрапляло на екрани пристроїв;

– протерти трохи зволоженою серветкою клавіатуру (для зниження рівня статичної електрики), зовнішню поверхню екрана монітора;

– освітленість у приміщеннях з ПК регулювати сонцезахисними пристроями;

– перед вмиканням штепсельної вилки – упевнитися в тому, що вимикачі мережі на всіх пристроях ПК знаходяться в положені «вимкнено»;

– при виявленні будь-яких несправностей ПК не вмикати і негайно відповідальну за це людину. [11]

Під час роботи за ПК важливо:

– не залишати працюючі ПК і їхні пристрої без нагляду;

– підключати і відключати роз'єм кабелів пристроїв ПК тільки при відключеній напрузі;

– подавати напругу на пристрої і окремі блоки ПК тільки після ретельної перевірки надійності кріплення провідників заземлення, справності кабелів і роз’ємів мережі електромережі;

– при виявленні запаху горілого в пристроях ПК, необхідно вимкнути пристрій, і звернутися до спеціаліста з технічного обслуговування ПК;

– для профілактики порушень і підтримання працездатності оператора (користувача) ПК власником повинні бути введені додаткові регламентовані перерви для відпочинку;

– у період роботи за дисплеєм необхідно передбачити через кожні 40-45 хв три-п’ятихвилинні перерви для відпочинку. Середня сумарна

тривалість роботи за монітором за день не повинна перевищувати 4-5 год, а за тиждень 20-25 год. [10]

Після завершення роботи за ПК необхідно:

– зберегти інформацію;

– вимкнути пристрій;

– вимкнути стабілізатор, якщо комп’ютер під’єднаний до мережі через нього;

– прибрати робоче місце. [12]

3.3 Висновок до третього розділу

В третьому розділі кваліфікаційної роботи, було розглянуто методи боротьби з монотонністю праці на виробництві, що є досить актуальною темою для розробників програмного забезпечення, які постійно стикаються з так званим “вигоранням”. Висвітлено чому в таких випадках важливо правильно розбивати монотонні задачі та чергувати їх між собою зміщуючи увагу з проблемних монотонних задач, важливість перерв та відпочинку. Також описано загальні вимоги безпеки з охорони праці для користувачів перед початком виконання роботи, під час виконання роботи та після завершення виконання роботи за ПК.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи освітнього рівня “бакалавр”, розробки веб-сайту “Artise” засобами React, Node.js та MongoDB, було виконано всі раніше поставлені завдання при дослідженні задачі.

Результати дослідження теми було інтерпретовано у вигляді написання трьох розділів кваліфікаційної роботи, два з яких технічні що мають на меті реалізацію веб-сайту.

В першому розділі було проаналізовано предметну область, сформовано вимоги до розробки веб-сайту, проведено пошук актантів та можливі варіанти використання веб-сайту, детальніше описано ключові. Вибрано стек для реалізації.

В другому розділі описано архітектуру веб-сайту “Artise”, детальніше про вибір технологій відповідно до стеку, файлову структуру веб-сайту, структурні особливості вибраної бази даних, розгортання на хостигових платформах, про тестування та використання розробленої системи.

В третьому розділі, було розглянуто методи боротьби з монотонністю праці на виробництві та сформовано загальні вимоги безпеки з охорони праці для користувачів ПК.

В процесі роботи над кваліфікаційною роботою:

– освоєно новий теоретичний матеріал досліджуваної теми, заповнено пробіли у наявних знаннях;

– розглянуто сучасний інструментарій розробки веб-сайтів та вибрано підходящий;

– проведено порівняння між функціональними можливостями веб-сайту “Artise” та функціональними можливостями конкурентів;

– на основі отриманої інформації веб-сайт було: розроблено, протестовано та розміщено на хостингових платформах;

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Deploying Static Sites with Netlify [Електронний ресурс]. – Режим доступу до ресурсу: https://www.codecademy.com/learn/deploying-with-netlify-and-heroku/modules/deploying-static-sites-with-netlify.

2. Faker documentaion [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://fakerjs.dev/about/announcements/2022-01-14.html.

3. Guide to develop MERN application, step by step. Part 1 [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://dev.to/osiroski/guide-to-develop-mern-application-step-by-step-part-1-2h3l/.

4. Introduction to Apollo Client [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.apollographql.com/docs/react/.

5. JavaScript [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

6. MongoDB [Електронний ресурс]. – Режим доступу до ресурсу: https://itglobal.com/ru-ru/company/glossary/mongodb/.

7. Netlify docs [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://docs.netlify.com/

8. TailwindCSS – очередной фреймворк или новый шаг эволюции? [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://habr.com/ru/post/508844/.

9. The Use Case Model [Електронний ресурс] – Режим доступу до ресурсу: https://sparxsystems.com/resources/tutorials/uml/use-case-model.html.

10. Вимоги безпеки під час роботи на ПК [Електронний ресурс]. – Режим доступу до ресурсу: https://sites.google.com/site/ohoronapraci44/33-vimogi-bezpeki-pid-cas-roboti-na-pk.

11. Вимоги охорони праці перед початком роботи на комп'ютері [Електронний ресурс]. – Режим доступу до ресурсу: https://sites.google.com/site/ohoronapraci44/01-vimogi-ohoroni-praci-pered-pocatkom-roboti-na-komp-uteri.

12. Інструкція з охорони праці при роботі на персональному комп’ютері [Електронний ресурс]. – Режим доступу до ресурсу: https://www.sop.com.ua/article/485-nstruktsya-z-ohoroni-prats-pri-robot-na-personalnomu-kompyuter.

13. Основні заходи, спрямовані на запобігання монотонності і підвищення змістовності праці [Електронний ресурс]. – Режим доступу до ресурсу: https://studentbooks.com.ua/content/view/951/76/1/4.

14. Профессиональный React стек для создания сложных приложений в 2022 году [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://habr.com/ru/post/646887/.

15. Руководство по MongoDB [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://metanit.com/nosql/mongodb/1.1.php

16. Стек MERN. Что, как и почему? [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://habr.com/ru/post/653981/.

ДОДАТКИ

Додаток А

Лістинг 1 – Код головної сторінки веб-сайту (Home.js)

```
import React from 'react';
import SongListGrid from '..componentslayoutssongSongListGrid';
import useSongs from '..hooksuseSongs';

const Home = () = {
  const { data songs } = useSongs();

  return SongListGrid songs={songs} ;
};

    export default Home;
```

Лістинг 2 – Код компоненту багаторазового використання SongListGrid

```
import {
  Box,
  Fade,
  Flex,
  Icon,
  Image,
  SimpleGrid,
  Text,
} from '@chakra-ui/react';
import { FaPencilAlt, FaTrashAlt } from 'react-icons/fa';
import SongListCard from './SongListCard';

const SongListGrid = ({
  songs,
  title,
  emptyTitle,
  emptyImageSrc,
  listType = 'songs',
  actionButtonsVisisble,
  onPressTitle,
  onPressEdit,
  onPressRemove,
}) => {
  const ActionButtons = () => {
    return (
      <Fade in={actionButtonsVisisble}>
        <Flex
          border="1px"
          borderColor="gray.200"
          borderRadius="md"
          p="5px"
          gap={3}
          ml={3}
```

```
        align="center"
        mb={5}
      >
        <Icon
          cursor="pointer"
          height={4}
          width={4}
          _hover={{
            color: 'cyan.500',
          }}
          as={FaPencilAlt}
          onClick={onPressEdit}
        />
        <Icon
          cursor="pointer"
          height={4}
          width={4}
          _hover={{
            color: 'red.500',
          }}
          as={FaTrashAlt}
          onClick={onPressRemove}
        />
      </Flex>
    </Fade>
  );
};

if (songs.length === 0 && listType === 'songs') {
  return (
    <Flex justify="center" align="center">
      <Box>
        <Text
          p={[5, 5, 5, 10]}
          position="absolute"
          fontSize="2xl"
          fontFamily="monospace"
          fontWeight="bold"
          color="white"
          maxW="50vw"
        >
          {emptyTitle || 'Your song list is empty try to add one. 🎸'}
        </Text>

        <Image
          h={['50vw', '60vw', '35vw']}
          w="100%"
          src={
            emptyImageSrc ||
            'https://images.pexels.com/photos/5157803/pexels-photo-
5157803.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=2'
          }
        />
```

```
      </Box>
    </Flex>
  );
}

if (songs.length === 0 && listType === 'playlists') {
  return (
    <Box mb={5}>
      <Flex alignItems="center">
        <Text
          fontSize="2xl"
          maxW="70vw"
          fontFamily="monospace"
          fontWeight="bold"
          pb={5}
          cursor="pointer"
          onClick={onPressTitle}
        >
          {title}
        </Text>

        {listType === 'playlists' && <ActionButtons />}
      </Flex>

      <Text fontSize="xl">
        {emptyTitle || 'No songs in this playlist try to add one. 🎸'}
      </Text>
    </Box>
  );
}

return (
  <Box mb={5}>
    <Flex alignItems="center">
      {title && (
        <Text
          fontSize="2xl"
          maxW="70vw"
          fontFamily="monospace"
          fontWeight="bold"
          cursor="pointer"
          pb={5}
          onClick={onPressTitle}
        >
          {title}
        </Text>
      )}

      {listType === 'playlists' && <ActionButtons />}
    </Flex>

    <SimpleGrid columns={[1, 2, 2, 3, 4]} spacing={10}>
      {songs.map((song) => (
```

```
          <SongListCard key={song.id} song={song} />
        ))}
      </SimpleGrid>
    </Box>
  );
};

    export default SongListGrid;
```

Лістинг 3 – Код сервісу музичних треків (серверна частина веб-сайту)

```
import mongo from 'mongodb';
import MongoClientProvider from './MongoClientProvider';

class SongService {
  collectionName = 'songs';

  canViewSong = (song, userId) => {
    const authorObjectId = new mongo.ObjectID(song.authorId);
    const userObjectId = new mongo.ObjectID(userId);
    if (authorObjectId.equals(userObjectId)) return true;
    return false;
  };

  getCollection() {
    return MongoClientProvider.db.collection(this.collectionName);
  }

  async getUserSongs(query = {}, context) {
    return this.getCollection()
      .find({ authorId: new mongo.ObjectID(context.userId), ...query })
      .sort({ createdAt: -1 })
      .toArray();
  }

  async getPublicSongs(isPublic) {
    return   this.getCollection().find({   isPublic:   true   }).sort({
    createdAt: -1 }).toArray();
  }

  async findSongById(_id) {
    const objectId = new mongo.ObjectID(_id);
    return this.getCollection().findOne({ _id: objectId });
  }

  async findSongByFields(song, singer, genre) {
    if (song)
      return this.getCollection()
        .find({ song: { $regex: `${song}`, $options: 'i' } })
        .toArray();
    if (singer)
      return this.getCollection()
```

```
        .find({ singer: { $regex: `${singer}`, $options: 'i' } })
        .toArray();
    if (genre)
      return this.getCollection()
        .find({ genre: { $regex: `${genre}`, $options: 'i' } })
        .toArray();
  }

  async createSong(song, context) {
    const doc = { ...song, authorId: new mongo.ObjectID(context.userId),
    createdAt: new Date(), updatedAt: new Date() };
    const res = await this.getCollection().insertOne(doc);
    const _id = res.insertedId;
    return { ...song, _id };
  }

  async updateSong(_id, data, context) {
    await this.getCollection().updateOne(
      { _id: new mongo.ObjectID(_id) },
      { $set: { ...data, updatedAt: new Date() } },
    );
  }

  async resetSongs() {
    await this.getCollection().remove({});
  }

  async deleteSongById(songId) {
    await        this.getCollection().deleteOne({        _id:        new
    mongo.ObjectID(songId) });
  }
}
    export default new SongService();
```