Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Метод персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент 4 курсу, група КН-18-1* _____ *О.М. Щіпайло*
Курс, група виконавця — Підпис — Ініціали, прізвище

Керівник: *к.т.н., доцент кафедри КН* _____ *О.В. Мазурець*
Науковий ступінь, посада — Підпис — Ініціали, прізвище

Нормоконтроль: *к.т.н., доцент кафедри КН* _____ *Р.О. Багрій*
Науковий ступінь, посада — Підпис — Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор _____ *О.В. Бармак*
Підпис — Ініціали, прізвище

16 червня 2022 р.

Хмельницький 2022

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

_____
(підпис)
д.т.н., професор О.В. Бармак
«*25*» *березня* *2022* року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Метод персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів»*

2. Завдання видано студенту *Щіпайло Олександру Михайловичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *доцент кафедри КН Мазурець Олександр Вікторович*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «*01*» *березня* *2022* р. № *18*

5. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Мета роботи – розробка методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів за допомогою рекурсивного анкетування користувача для формування індивідуального вектору ознак за тегами, що створюються користувачем при публікації цифрового графічного твору мистецтва і подальшого передавання роботи в базу даних для подальшого представлення на веб-сервісі.

Виконавець: *студент 4 курсу, група КН-18-1* _____ *О.М. Щіпайло*
Курс, група виконавця — Підпис — Ініціали, прізвище

Керівник: *к.т.н., доцент кафедри КН* _____ *О.В. Мазурець*
Науковий ступінь, посада — Підпис — Ініціали, прізвище

Анотація

Тема кваліфікаційної роботи бакалавра: *«Метод персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів»*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-18-1 Щіпайло Олександр Михайлович*

Керівник кваліфікаційної роботи бакалавра: *к.т.н., доцент кафедри КН Мазурець Олександр Вікторович*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
74	45	24	20	3

В основі кваліфікаційної роботи бакалавра лежить розробка веб-сервісу цифрових графічних творів мистецтва з елементами персоналізованого підбору компонентів інтернет-контенту. Для розробки інформаційної системи було використано мову програмування JavaScript, бібліотека React, а також систему керування базами даних MS SQL Server.

Розроблена система призначена для формування користувачем власного портфоліо цифрових графічних творів мистецтва шляхом публікації своїх робіт на веб-сервісі та надання їм особливих міток (тегів) для кращого фільтрування контенту іншими користувачами по бажаним критеріям.

Напрямками практичного використання системи є багаторівнева фільтрація цифрових графічних творів мистецтва по заданим критеріям та рекурсивне анкетування користувачів для формування індивідуального вектору ознак за допомогою ключових слів (тегів).

Виконавець: *студент 4 курсу, група КН-18-1* (Курс, група виконавця) _____ (Підпис) *О.М. Щіпайло* (Ініціали, прізвище)

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
БД	База даних
ООП	Об’єктно орієнтоване програмування
КРБ	Кваліфікаційна робота бакалавра
КН	Комп’ютерні науки
3D	Тривимірна графіка
2D	Двовимірна графіка
VFX	Visual effects
SEO	Search engine optimization
VR	Virtual reality
MS	Microsoft
DOM	Document object model
API	Application programming interface
UI	User interface

Вступ

На сучасному етапі для публічної демонстрації у рекламних цілях своїх цифрових графічних творів мистецтва митці використовують як окремі персональні сайти-портфоліо, так і спеціальні веб-сервіси, які дозволяють створити власне портфоліо та додати його до загальної бази. У свою чергу, працедавці та потенційні покупці таких графічних творів мистецтва по результатах перегляду портфоліо зв’язуються з митцями для обговорення комерційних пропозицій.

Характерною рисою таких веб-сервісів для створення та перегляду портфоліо цифрових графічних творів мистецтва є велика кількість матеріалів (цифрових графічних творів мистецтва), які розподіляються по багатьох різних критеріях – тегах. Тому обрання потрібної комбінації тегів для більш зручного пошуку графічних матеріалів для працедавців та потенційних покупців складає серйозну проблему. Автоматизація цього процесу є актуальною задачею на сучасному етапі, яка дозволить полегшити процес пошуку цифрових графічних творів мистецтва у колективних електронних портфоліо.

Мета кваліфікаційної роботи бакалавра – створення й програмна реалізація методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів.

Об’єкт дослідження – процес підбору комбінацій тегів для пошуку цифрових графічних творів мистецтва у колективних електронних портфоліо.

Предмет дослідження – інформаційні технології, моделі, методи та засоби для автоматизованого підбору комбінацій тегів для полегшення процесу пошуку цифрових графічних творів мистецтва у колективних електронних портфоліо.

Розділ 1 Характеристика предметної області: аналіз моделей, методів та реалізацій

1.1 Аналіз інформаційних моделей

Тривимірна графіка на сьогоднішній день настільки проникла в наше життя, що інколи ми навіть не звертаємо уваги на її появу. Роздивляючись рекламні блоки в інтернеті і на вулиці з зображеннями інтер'єру кімнати, дивлячись гостросюжетний фільм або граючи комп'ютерні ігри, ми навіть не здогадуємось, що за цим всім стоїть важка робота графічного дизайнера.

3D графіка (тривимірна графіка) – це особливий вид комп'ютерної графіки, комплекс методів та інструментів, що мають застосування в створенні зображень 3D-об'єктів [1]. 3D графіка міцно увійшла в наше життя і вже має застосування в архітектурі, промисловості. Після появи 3D друку, тривимірне моделювання перейшло на новий рівень і стало ще популярнішим. Відповідно, з'явились такі професії: 3D-дизайнер, художники та арт-дизайнери, текстурщик тощо. Напрямки, у яких наразі застосовується 3D графіка [2]:

1. Мультиплікація: створення мультфільмів;

2. 3D візуалізація архітектури: застосовується в іграх, фільмах і будівництві;

3. Промисловість: складні моделі збирають по точних вимірюваннях. Є програми по створенню креслень, для розробки продукту по кресленням, для тестування продуктів тощо;

4. Створення комп'ютерних ігор – розробка 3D контенту зазвичай йде творчим шляхом, точні виміри моделей застосовуються не часто.

5. Медична сфера;

6. Реклама та маркетинг.

Швидкий розвиток 3D моделювання передбачає виникнення безлічі напрямків у яких можна розвиватись. Основними є [1]: анімація, моделювання ландшафтів, gameDev, VFX, gameDesign, візуалізація інтер'єрів, архітектура handPaint тощо. Відповідно, так як напрямків в графіці дуже багато, то до

кожного напрямку художники та дизайнери використовують свої інструменти. Самі відомі з них: Blender, Maya, Unreal Engine, Unity, Zbrush, Fushion 360, Unity, Substance Designer, Marmoset тощо.

Для того, щоб створити 3D модель потрібно певні етапи, багато з яких проходяться в різних програмах, а саме [3]:

1. Концепт та дизайн;
2. Скульптинг або моделювання High Poly моделі;
3. Репотологія;
4. Розгортка;
5. Запікання карт;
6. Текстурування;
7. Rigging і Skinning.

Етап 1 – Концепт та дизайн

Починати потрібно з ідеї, яку необхідно виразити в концептах. Це важливий етап, який є фундаментом всієї роботи. Тому що, концепт це план, дотримуючись якого можна зекономити багато часу. Маючи концепт, можна чітко представляти фінальний результат і кроки, які необхідні для його реалізації рисунок 1.1.

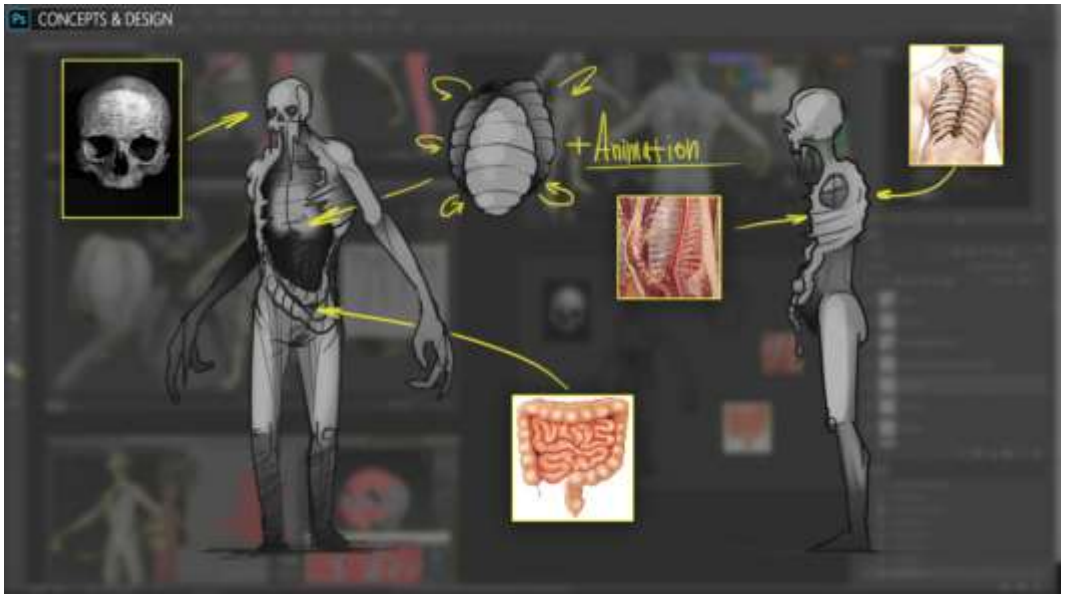


Рисунок 1.1 – Розробка базових форм персонажа перед моделюванням (Етап 1) [3]

Етап 2– Скульптинг або моделювання High Poly моделі

Головне завдання на цьому етапі створти максимально деталізовану модель. Так як на наступних етапах створити деталізовану модель буде проблематично. Тут треба застосувати всі свої знання та навички, щоб створити якісну роботу. Для цього ідеально підійдуть Zbrush. Maya або Blender, рисунок 1.2.

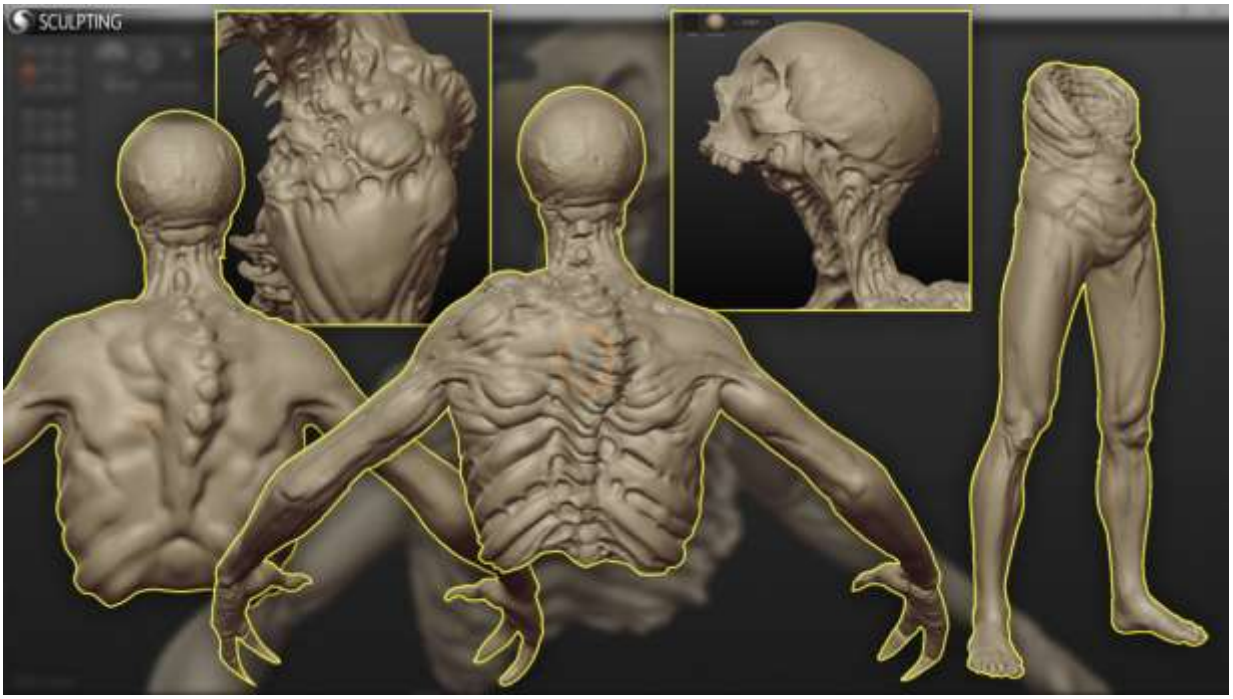


Рисунок 1.2 – Процес розробки деталізованої моделі персонажу (Етап 2) [3]

Етап 3 – Репотологія

Завершивши роботу над попереднім етапом, потрібно перейти до оптимізації моделі, тому що в такому вигляді, в якому вона знаходиться на другому етапі, використовувати в якомусь продукті не раціонально. Для цього придумали репотологію, що створена для зменшення кількості полігонів моделі та її оптимізації. В кожній програмі використання репотології відрізняється, рисунок 1.3.

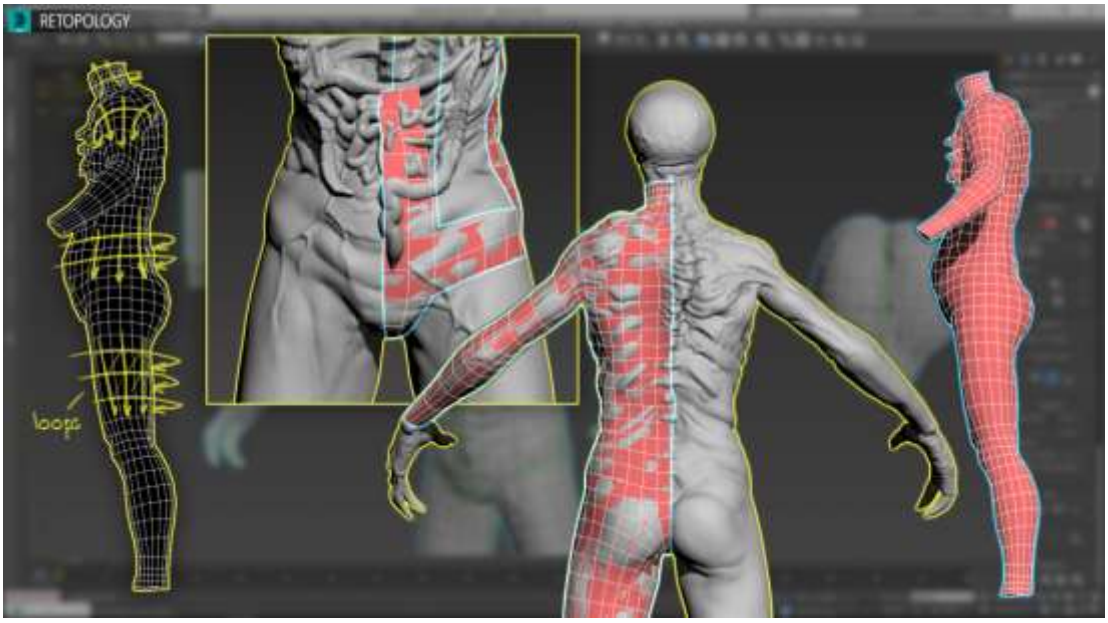


Рисунок 1.3 – Репотологія – оптимізація розміру моделі персонажу (Етап 3) [3]

Етап 4 – Розгортка

На цьому етапі розгортають всі частини меша, щоб текстура коректно лягла на модель. Якщо якась частина моделі потребує більшої деталізації, то їй виділяється більше місця на розгортці. При створенні розгортки треба ховати шви в менш помітних місцях, так як це портить вигляд моделі і текстури на ній, рисунок 1.4.

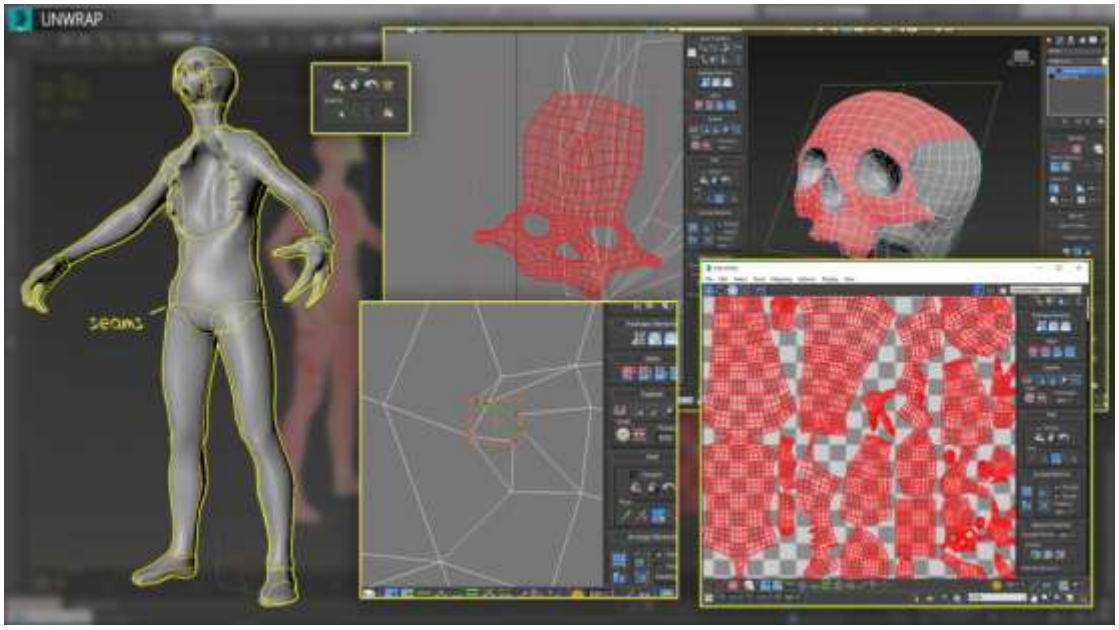


Рисунок 1.4 – Розгортання моделі для подальшого текстурування(Етап 4) [3]

Етап 5 – Запікання карт

Щоб перенести деталізацію з High Poly на Low Poly моделі, використовуються текстурні карти, такі як Normal Map. Ambient Occlusion, Diffuse Map тощо. Для запікання карт ідеально підійде Substance Painter, рисунок 1.5.

Рисунок 1.5 – «Запікання» карт в Substance Painter для пошуку помилок перед текстуруванням (Етап 5) [3]

Етап 6 – Текстурування

В Substance Painter є можливість красити прямо по моделі. Для текстурування можна використати як готові матеріали, так і створенні руками якісь текстури для специфічних потреб, рисунок 1.6.

Етап 7 – Rigging і Skinning

Для того, щоб оживити персонажа потрібно створити йому кістки і прикріпити до них модель. Skinning – це один з етапів сетапа 3D персонажа, коли готовий скелет прив’язується до самої моделі 3D персонажа. Це достатньо складний процес, оскільки потрібно вірно назначити вагу для кожної з вершин. Чим більше вага, тим більше впливає конкретна кістка на конкретну вершину моделі, рисунки 1.7, 1.8.

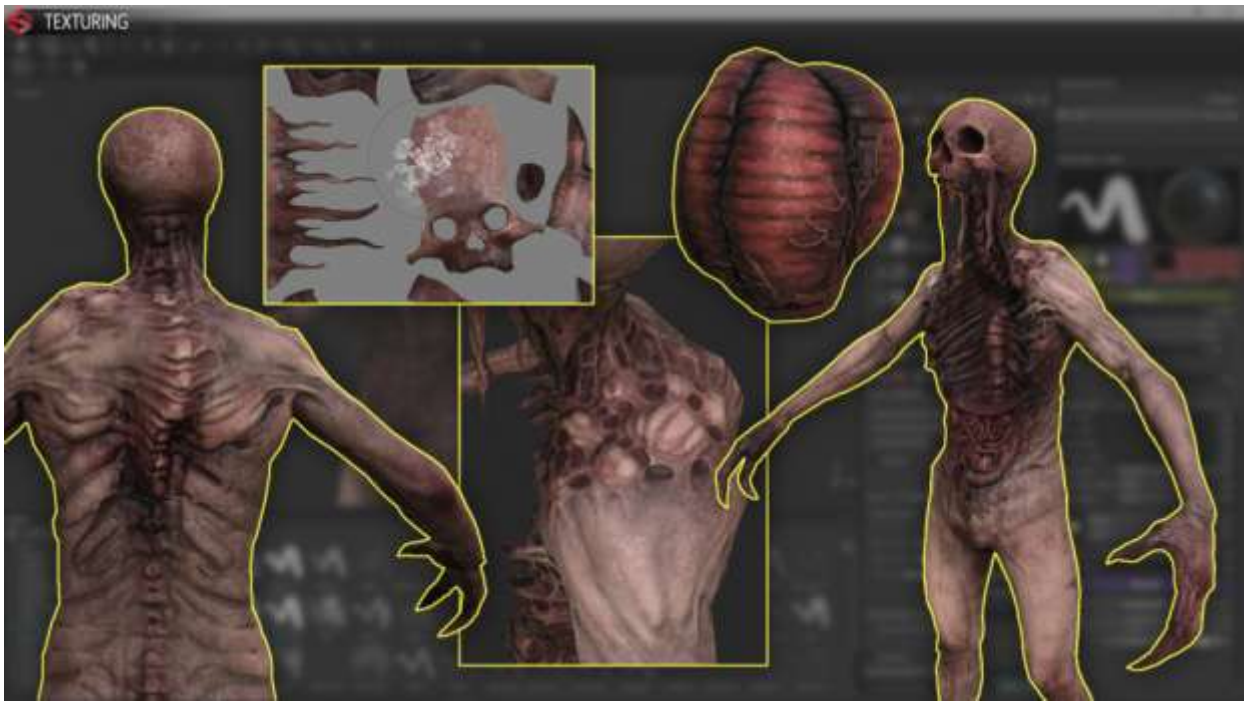


Рисунок 1.6 – Текстурування – надання моделі якось стилю (Етап 6) [3]

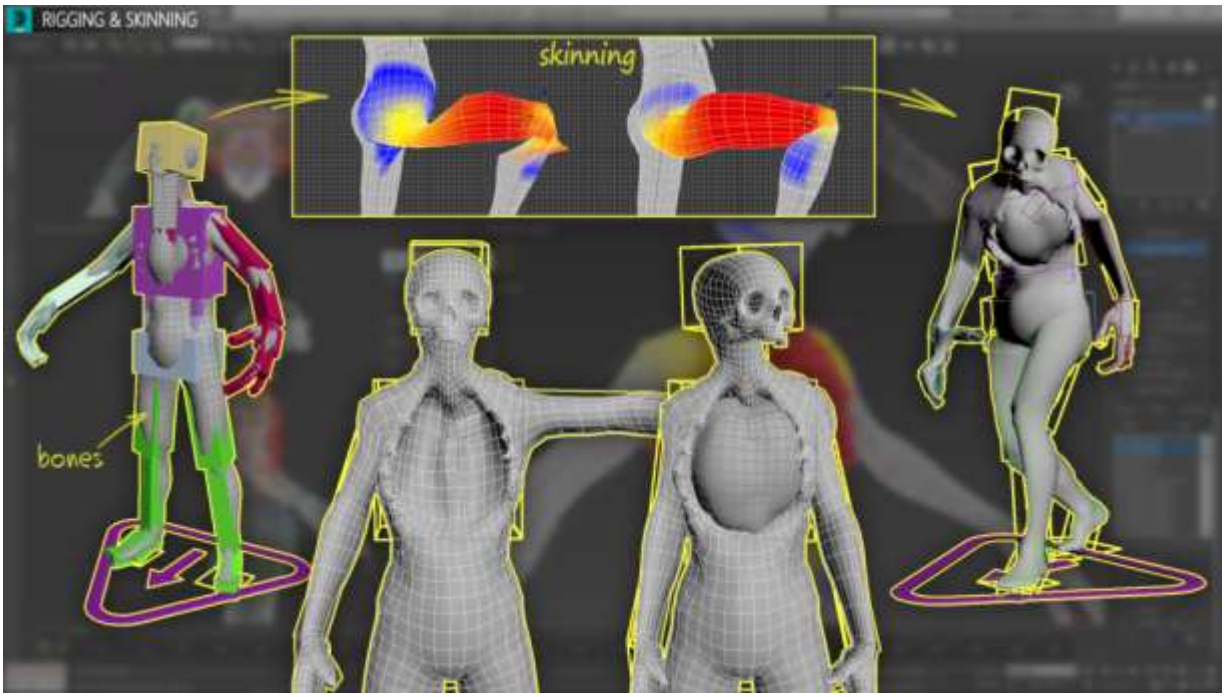


Рисунок 1.7 – Rigging, Skinning – надання моделі реалістичного вигляду за допомогою ваги кінечностей [3]

Створивши модель і пройшовши через всі етапи розробки, її можна опублікувати в інтернеті [3]. Для цього використовують сайти-портфоліо

(ArtStation, Pinterest, SketchFab), які дозволяють створити власне портфоліо та додати його до загальної бази [1]. Характерною рисою таких веб-сервісів для створення та перегляду портфоліо цифрових графічних творів мистецтва є велика кількість матеріалів які розподіляються по багатьох різних критеріях – тегах.

Рисунок 1.8 – Результат процесу моделювання

Обрання потрібної комбінації тегів для більш зручного пошуку графічних матеріалів для працедавців та потенційних покупців складає серйозну проблему. Тому про автоматизацію подібної завдання думають ще на етапі розробки. Характерною рисою такого процесу є пошук по певним критеріям, що неможливо зробити через звичайний пошук.

Результати пошуку виводяться в порядку найбільшої відповідності до назви цифрового твору мистецтва, ключових слів і характеристик по запиту користувача [4]. Щоб пропонувати користувачам найбільш підходящі результати пошуку, алгоритми в першу чергу орієнтуються на три фактора: релевантність, залучення, якість. При оцінці релевантності враховується наскільки назва, опис, теги і вміст цифрового твору мистецтва відповідають запиту користувача.

Залучення вимірюється по сигналам, які поступають від користувачів. Цей фактор напряму пов'язаний з релевантністю, Наприклад, кількість переглядів зображень, знайденому по певному запиту, допомагає зрозуміти, наскільки результат пошуку виправдав очікування користувачів і чи потрібно пропонувати його в результатах пошуку [5].

В додаток до цих трьох факторів система підбирає для користувачів актуальні роботи враховуючи історію перегляду користувача та пошуку. Тому результати пошуку можуть бути різними для різних користувачів. Автоматизація процесу тегування робить зручним пошук потрібних цифрових творів мистецтва для замовників та роботодавців, що робить також зручним пошук творців, що створюють роботи по конкретним тегам.

Для фрілансерів (творців цифрових творів мистецтва) на сайтах-портфоліо кожен тиждень буває з десяток хороших замовлень, де можуть випробувати долю і новачки. Крім цього, багато замовників шукають виконавців також через каталоги фрілансерів. Але новачку отримати замовлення через каталог проблематично: потрібні позитивні відгуки від попередніх замовників, хороше портфоліо де є вже мінімум 2 якісних роботи і непоганий рейтинг в каталозі. Тому для початку новачки влаштовуються на постійну роботу, де можуть набути досвід в команді [6].

Зазвичай в команді розробник спеціалізується на якомусь конкретному завданню. Сама популярна тема – архітектура. Сюда входить моделювання і візуалізація екстер’єрів будівель. Часто попадаються проекти по створенню об’єктів для поліграфії та реклами, зазвичай: упаковки, бутилки, електроника, деталі конструкцій, логотипи тощо. Наступне направлення – створення персонажів. Напрямок створення персонажів доволі спецефічний і потребує високих вмінь ті виконання, тому новачку отримати такий проєкт без прикладів хороших робіт буде важко. Самий популярний напрямок для новачка це створення пропсів органічних або неорганічних статичних моделей (Трава, забор, стіл, вода тощо). Також без уваги не залишаються і інші напрямки, яких налічується сотнями [6].

Для підсумку слід сказати, що на сьогоднішній день є сенс займатись вивченням 3D або 2D. Ринок у веб дизайні, розробці відео ігор, кіно, рекламі, архітектурі тощо, розвивається в геометричній прогресії, кожного року з'являється з десяток нових напрямків. Новачки в цій сфері без роботи не залишаються, що через велику конкурентність між роботодавцями в пошуку спеціалістів, дає змогу другим добре заробляти.

Етапи створення 3D моделі були розглянуті у зв'язку з тим, що кожен із них визначає ряд можливих тегів, за якими можна проводити тегування для подальшого формування ієрархії класів засобами анкетування зразками компонентів

1.2 Огляд теоретичних підходів до розв'язку подібних задач

Тегування – це повторювана кластеризація – це об'єднання схожих запитів в групи по певних критеріях і використання в оптимізації цих груп замість окремих ключів. За допомогою кластеризації ви очищуєте своє семантичне ядро, розділяючи його на зручні для роботи групи [7].

Розділення семантики на кластери – важлива частина початкових робіт над веб-сайтом. Важливим поштовхом стало оновлення Google під назвою «Колібрі» в 2013 році, яке позначило перехід до семантичних SEO. Пошуковик навчився краще розуміти запити і розпізнавати релевантні речення замість окремих ключових слів. Оновлення RankBrain 2015 року закріпило тенденцію, що цей алгоритм оцінює взаємодію користувачів в пошуку, він має змогу виявляти теми запитів і підбирати декілька схожих по сенсу.

Кластеризація пошукових запитів дасть уявлення про те, яким контентом потрібно наповнювати сторінки, на які фрази робити упор при фільтрації і як оптимізувати різні частини веб-платформи.

За допомогою групування ключових слів (тегів) можна [7]:

– Краще розуміти користувацький інтент. Аналіз тематичної релевантності робить фільтрацію зручною та швидкою для користувачів:

об’єднуючи схожі слова та вислови, можна цілитись на сенс пошукового запиту замість окремих запитів.

– Збирати максимум ключів для фільтрування контенту. За допомогою кластеризації можливо ранжувати по групах семантично близькі слова та фрази, а не по окремих запитах.

– Створити ефективну структуру веб-сайту. Кластеризація запитів допоможу проаналізувати семантичні зв’язки між сторінками і зробити структуру зручнішою.

– Заощадити час та уникнути великої кількості помилок. Якщо процес автоматизований кластеризацією запитів, то всі переваги кластеризації отримані під час фільтрування йдуть з швидше та з меншою кількістю помилок.

– Розуміти потенціал різних частин веб-ресурсу. Отримані запити з кластерів допоможуть зрозуміти, як різні частини сайту пов’язані між собою, якщо ви починаєте роботу, то на яких тематиках потрібно сконцентруватись. Треба розглянути свій сайт з точки зору пошукової системи і проаналізувати, на які категорії та розділи треба звернути увагу при розробці.

– Виключити лишні слова з семантики. Кластеризація запитів допомагає зробити великий перелік ключів більш зрозумілим при використанні. Так можливо відсіяти нерелевантні запити.

– Покращити пошукову видимість. За допомогою ключових слів можна краще розуміти свою семантику і з більшою уважністю підходити до створення контенту, що зробить контент більш авторитетним для користувачів, так і для систем фільтрування.

Перед тим як зайнятись розподіленням ключів на тематичні групи, потрібно зібрати їх повний список для сайту. Збір семантики – фундаментальне завдання на початку роботи з веб-ресурсом [8], рисунок 1.9. Цей процес допоможе зрозуміти, що шукають користувачі на яких ви орієнтуєтесь. Після цього треба розбити теги на групи (Кластери) по частоті їх використання і кожну групу якось візуально розділити. Наприклад, кольором або розміром тексту тощо. Теги можна показати на картах з зображеннями при генерації каталогу.

Фільтрування тегів здійснювати при заданні певних параметрів або натисканні по конкретному зображенню, після цього контент відфільтрується по схожих тегах.

Після того як був зібраний список ключових слів, треба розділити їх на групи і відсіяти непотрібне.

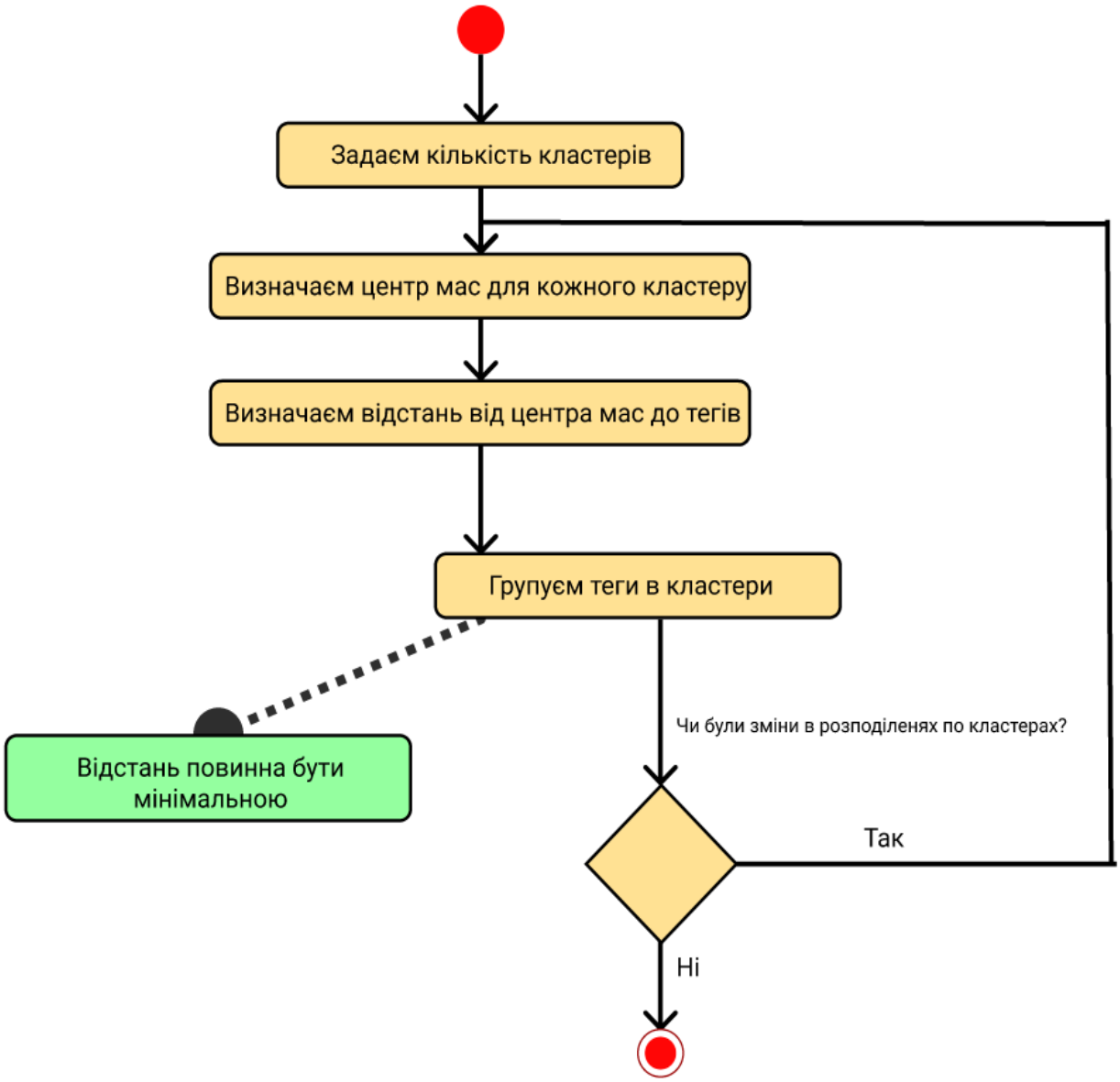


Рисунок 1.9 – Процес розподілення тегів по кластерах [8]

Алгоритм розподілення тегів по кластерах:

1. Задаєм кількість кластерів (Груп).

2. Для кожної групи задаєм центр мас. Це просте число, яке зв’язане з параметрами групи, по яких відбувається розбивання на групи.

3. Визначаєм відстань від кожного центру мас до кожного тегу.

4. Групуємо теги в кластери. Находим, який центр мас знаходиться ближче всього до заданого тегу. В цей кластер тег і добавляєм.

5. Перераховуємо значення центра мас. Для цього визначаєм середнє значення частоти використання всіх тегів, які увійшли в кластер на попередньому етапі.

6. Повторяєм пункти 3 і 5.

7. Алгоритм закінчується коли розподілення тегів по кластерах перестає змінюватись.

Групуючи запити вручну можна відфільтрувати їх по різних параметрах.

Можна відділити всі різні по формі і значенню фрази і об’єднати в групи ті, які мають однакові слова.

Розподіляти теги по тому, як будується запит, і фільтрувати різні інтенти. Можна добавляти теги для різних аспектів ключових слів по темі: наприклад, тип використання (онлайн або офлайн), цільова дія (знайти, купити, відкрити, скачати) тощо.

Після цього зрівняти теги і створити кластери на їх основі. Можна пройтись по кожному ключу окремо і розподілити їх між групами по будь-яких критеріях.

В цілому процес групування ключових слів багато в чому залежить від конкретних цілей і специфікації ніші контенту, незалежно від того автоматичний процес чи ні.

1.3 Аналіз існуючих програмних рішень

Створити сайт-портфоліо не саме легке завдання. Для цього потрібно мати навички у веб-розробці, продумати свій стиль подачі робіт і детально зупинитись на проєктах, якими художник пишається по особливому [9]. Але не

кожен дизайнер знається у веб-розробці та рекламі, тому для публічної демонстрації у рекламних цілях своїх цифрових графічних творів мистецтва митці використовують як окремі персональні сайти-портфоліо, так і спеціальні веб-сервіси, які дозволяють створити власне портфоліо та додати його до загальної бази цифрових графічних творів мистецтва, в такі як ArtStation, 3Dbaza, Shapeways, CGTrader тощо.

Сайти-портфоліо в своїй більшості схожі до «фото-стоків». Це своєрідні платформи для торгування зображеннями. Отож, в розробці графіки є подібні веб-ресурси, де ти можеш завантажити свої 3D моделі для продажу, а всі бажаючі зможуть купити їх в тебе за гроші [9]. Тому хочеться зразу згадати 3Dbaza, так як цей ресурс доволі цікавий. Цікавий тим, що на ринку 3D-стоків він недавно і конкуренція там невисока, тому роботи художників мають більші шанси бути поміченими. На платформі моделі продаються з заробітком від 50 до 75 відсотків [10].

Подібна платформа 3Dbaza є TurboSquid яка є однією з найкрупніших 3D-стоків. Величезна кількість купити високоякісні моделі зі всього світу. Платформа дозволяє авторам публікувати свої роботи і отримувати 40 відсотків при реєстрації і 80 при співробітництві, при заключенні контракту цей процент збільшується. Платформа популярна але має велику конкуренцію, тому без великих навичок заробити тут важко [11].

Без уваги неможливо залишити платформу CGTrader, що зображена на рисунку 1.10. Будь-який спеціаліст, який пов’язаний з дизайном знає, що це за ресурс. Платформа включає в себе величезну базу моделей на любий смак і будь-яку тему від простих моделей до VR. Однією із самих привабливих особливостей цієї платформи є нагорода авторам. Вважається, що воно саме велике в цій сфері – 90 відсотків в залежності від репутації автора на платформі [12].

Припустимо дизайнер вирішив зупинитись на якомусь вузькому напрямку в 3D, но зіткнувся з тим, що не знає як зробити свою роботу комфортною. На одному ресурсі мало авторських відрахувань, на іншому велика

конкуренція, а хочеться все одразу. Тоді Shapeways ідеальний варіант для 3D художника. Тут можна завантажувати свої моделі, купувати свої і печатати їх на 3D принтері для подальшого продажу. Це дуже цікавий функціонал, який може згодитися для власників своїх власних магазинів іграшок, різних аксесуарів, технічних виробів тощо [13].

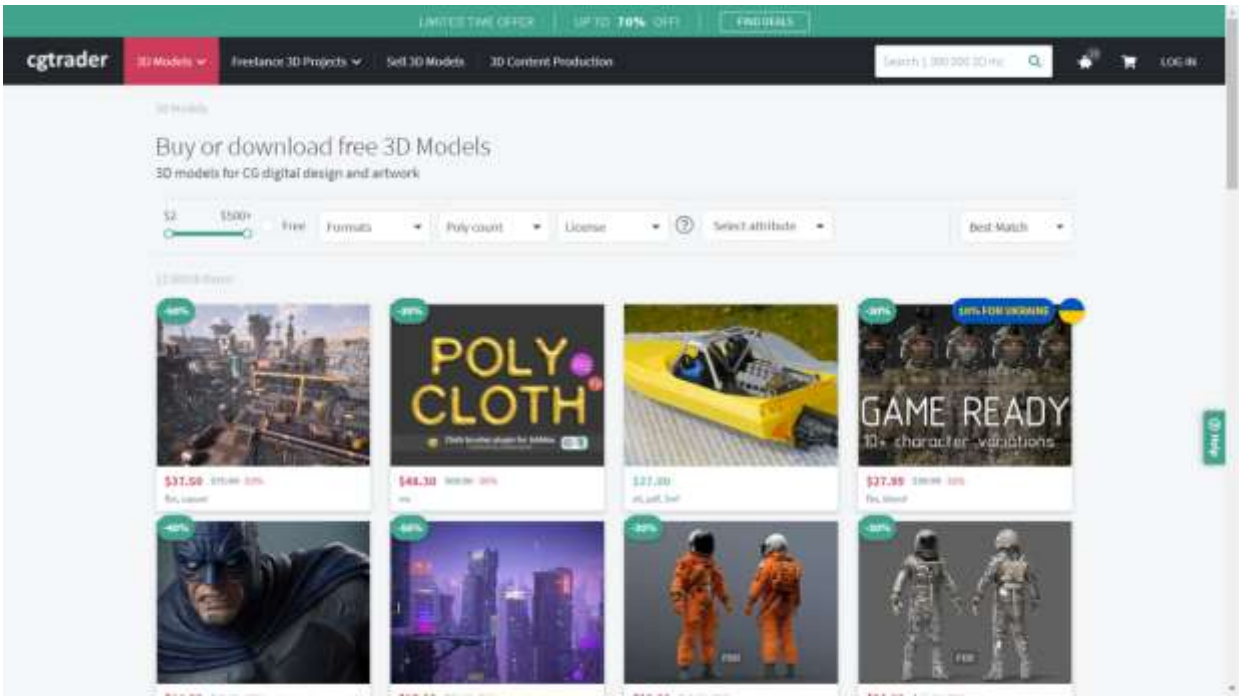


Рисунок 1.10 – CGTrader каталог [12]

Якщо дизайнер хоче не тільки займатись продажом моделей, але також хоче знайти нових друзів, досвід в команді, зв’язки в компаніях тощо. То для цього ідеально підходять дві самих відомих платформи для портфоліо: SketchFab та ArtStation. Вони є платформами не просто для продажу, це величезне ком’юніті дизайнерів, художників та програмістів.

SketchFab [14] виділяється своєю високо технологічністю – це веб-сервіс для публікації інтерактивних 3D-моделей в інтернет в режимі реального часу і їх послідуючому перегляду без використання як-небудь плагінів [15]. Можливістю маніпуляції об’єктами та взаємодії з ними, перегляд з різних ракурсів і налаштування моделі на свій смак при перегляді.

Рисунок 1.11 – Маніпуляція камерою в представленні моделей SketchFab [15]

Для того, щоб почати користуватися даним сервісом достатньо просто зареєструватись на ньому і завантажити свою модель. Сервіс підтримує 27 форматів розширень моделей. Після її завантаження, їй можна добавити мітки, опис або поділитись нею на інших сайтах. В любий момент можна видалити завантажену модель. Також можна добавляти SketchFab viewer на веб сторінку таким же чином, як відео на YouTube.

Одразу після завантаження буде доступно безліч налаштувань: назви, теги, опис моделі, зміна прозорості, освітлення, відображення, налаштування шейдерів тощо. Також присутній сервіс SkulptFab який дає змогу роботи моделі в браузері і завантажувати їх зразу на SketchFab.

Завершити огляд сучасних сайтів-портфоліо хочу ArtStation [16]. На сьогоднішній день ArtStation, рисунок 1.12, є самою великою платформою для розробників цифрових графічних творів мистецтва. В основному платформа орієнтується на GameDev і GameDesign. ArtStation включає в себе всі можливості у вище перелічених веб-ресурсах.

Artstation в першу чергу призначений для створення свого портфоліо і пошуку замовників. На платформі дуже велика конкуренція, значення має репутація. Конкурують в рейтингові між собою не тільки дизайнери, а також студії, розробники курсів, блогери, письменники, GameDev аналітики, програмісти, школи і навіть корпорації.

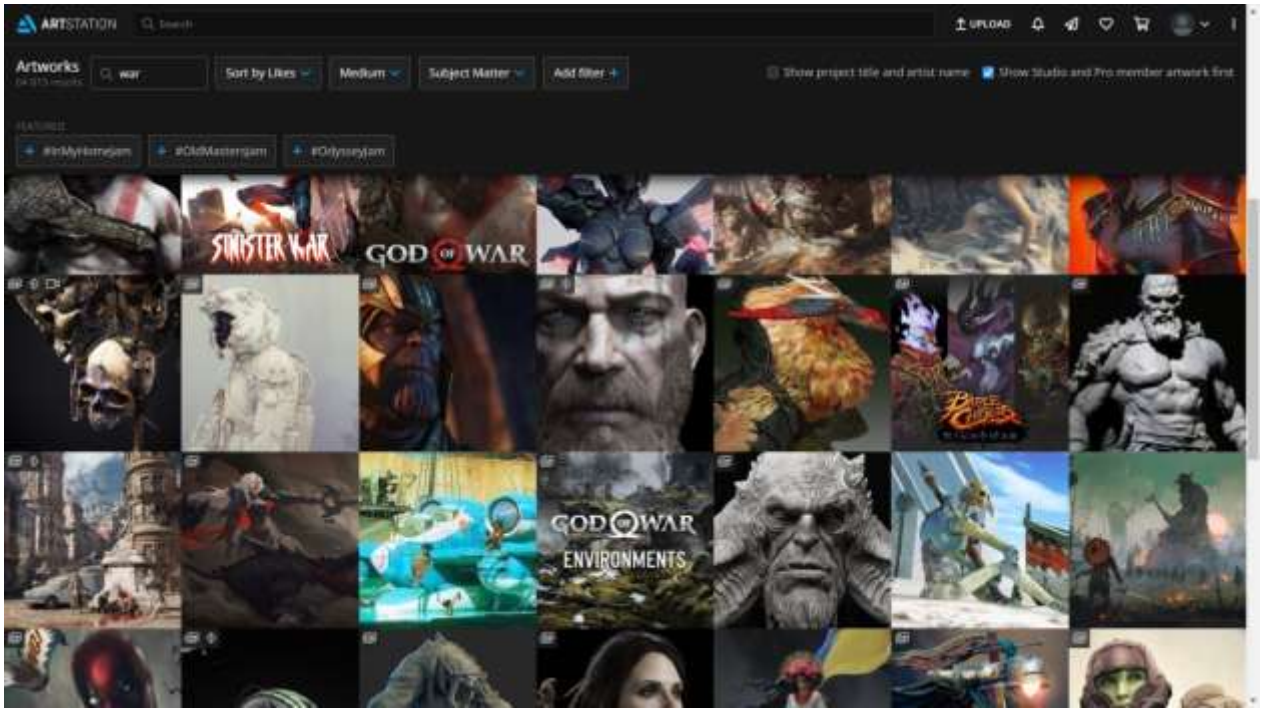


Рисунок 1.12 – Каталог ArtStation [16]

Особливістю платформи є багаторівнева фільтрація контенту. Через велику кількість контенту та вкладок приходиться робити пошук по кожній вкладці окремо. Присутній дуже детальний фільтр при пошуку графічних творів мистецтва, кожній роботі призначається окрема кількість тегів, що полегшує пошук по конкретним критеріям під час фільтрування контенту. Кожен дизайнер

при публікації роботи може призначити роботі вже існуючі теги або придумати свої, а кожен новий придуманий тег записується в базу даних фільтрування.

Теги відрізняються за своєю класифікацією. Є теги з типом розробки роботи, розширенням файлу та жанром, ці теги можуть на базовому рівні задати розробники. Також є тематичні теги, по них здійснюється основна фільтрація, ці теги задають автори при створенні цифрових графічних творів мистецтва.

Рисунок 1.13 – Перегляд моделі автора на ArtStation. На правій стороні можна побачити теги, що відносяться до цієї роботи [16]

Кожного дня на ArtStation з’являються тисячі нових робіт, а в світі нові комп’ютерні ігри, фільми, мультсеріали тощо. Тому розробникам ArtStation немає сенсу добавляти кожного дня по декілька тисяч нових тегів. Тому підхід з наданням авторам самим створювати теги допомагає полегшити пошук і фільтрацію, так як зазвичай в людей схожі смаки в чомусь конкретному, а адміністрацію платформи зберегти від зайвої роботи.

З вище зазначеного можна зробити висновок, що недивлячись на переваги сайтів-портфоліо, є значні недоліки.

По-перше, багато веб-ресурсів схожі між собою і зазвичай є копіями ArtStation або SketchFab. У платформ немає чогось особливого.

По-друге, на всіх платформах незручна фільтрація, окрім ArtStation. Тегування дуже обмежене, кожній роботі зазвичай прикріплюється лише один тематичний тег (Природа, меблі, одяг тощо). Немає тегів по типу та стилю роботи, неможна створювати власні теги, що дуже обмежує фільтрацію та пошук цифрових графічних творів мистецтва по конкретним критеріям.

По-третє, відсутнє будь-який автоматизований підбір тегів, візуальний або текстовий, що також обмежує фільтрацію, особливо коли користувач або замовник ще конкретно не визначився чого хоче і , можливо, не знає про існування тегів.

Вирішення вище згаданих проблем покращить швидкість та якість роботи платформи і фільтрування контенту.

1.4 Аналіз сучасних засобів створення програмного забезпечення

Підняття інтернету в основному пов’язане з його історією як візуального середовища. Для комп’ютерів і програм, потрібен був графічний інтерфейс, для розуміння повного потенціалу інтернету. Персональний комп’ютер не розповсюдився б по всьому світу і не став би широко використовуватись, поки електронні таблиці, текстові процесори та відеоігри не почали приваблювати користувачів [17].

Але на відміну від примітивних текстових редакторів з 90-тих, сьогодні є безліч способів розробки візуального інтерфейсу, такі як веб-сайти. Основна мова програмування при розробці сайтів є JavaScript.

Початково JavaScript був створений, щоб робити сайти живими. Програми на цій мові називаються скриптами. В старих методах розробки скрипти вставлялись в HTML сторінку і виконувались автоматично при завантаженні веб-сторінки. Скрипти розповсюджуються і виконуються як

простий текст. Їм не потрібна спеціальна підготовка до компіляції і запуску, так як JavaScript є інтерпретуємою мовою програмування.

Для того, щоб продемонструвати роботу JavaScript, використовують мову розмітки HTML і мову стилів CSS. HTML є структурою сторінки і задає блоки контенту на ній. CSS придає гарного вигляду сторінці і розставляє блоки структури у потрібних місцях за допомогою команд стилів, також за допомогою CSS можна робити прості анімації на сторінці. За допомогою JavaScript вже можна оживити сторінку і дати їй більшої інтерактивності, чого не можуть CSS і HTML.

На традиційному JavaScript теж можна створювати веб-сторінки за допомогою змінних, але такий код виходить більший ніж розмітка на HTML і такий код важко читати. Але JavaScript розвивався як і інші мови програмування і в ньому з’явилося ООП, а також велика кількість бібліотек, які полегшують створення веб-сайтів та веб-додатків.

Однією з таких бібліотек є React, де HTML, CSS та JavaScript можна поєднати в класах (які з’явились в JavaScript відносно недавно), розбити кожен клас і блок сторінки по компонентам і викликати код за допомогою методів класів. React використовується для створення користувацьких інтерфейсів. Він виконує єдине завдання: показує на сторінці компонент інтерфейсу, синхронізуючи його з даними додатка. Але тільки цієї бібліотеки в загальному випадку недостатньо для того, щоб повністю реалізувати проект.

Скоро після появи React і подібних йому рішень (Vue.js. Svelte), вони практично захопили світ веб-розробки, тому що вони допомагають вирішувати проблеми, базуючись на ідеї декларативного програмування, а не на імперативному підході. Виявилось, що декларативний підхід добре підходить для створення інтерфейсів, і він прижився в суспільстві. Декларативний підхід – це опис кінцевого результату, те що хочемо в кінці отримати. Імперативний підхід – описує конкретні кроки для досягнення кінцевого результату.

React – це не універсальний інструмент, який підійде для будь-якого проєкту. І в неї є свої переваги і недоліки, але ця бібліотека допоможе спростити життя розробникам. За її допомогою можна:

1. Побудувати інтерфейс з окремих компонентів, які легко підтримувати.

2. Вона добавляє зручний шар абстракції, позбавляючи від необхідності працювати з DOM.

3. React – це не нова бібліотека, за нею стоїть компанія Facebook і велике ком'юніті розробників. Тому вона прекрасно протестована, регулярно підтримується і стабільно оновлюється, а перехід до нових версій відбувається максимально безпечно.

4. На GitHub можна знайти готові React-компоненти на всі випадки життя. А якщо таких немає, але є потрібні незалежні бібліоеки, то можна пошукати інтеграцію або зробити її самостійно.

5. React – це проект з відкритим початковим кодом. За допомогою цього його можна безпечно використати навіть в комерційних додатках.

JavaScript пішов ще далі і окрім Frontend бібліотек як React, Vue.js тощо, тепер можна розробляти 3D контент. Для цього використовується такий міцний інструмент, як бібліотека Three.js. Вона допомагає використовувати в браузері 3D з хорошою продуктивністю, вона включає в себе набір готових класів для створення і відображення інтерактивності 3D графіки в WebGL. Подібна бібліотека може використовуватись на тих самих сайтах-портфоліо для графічних цифрових творів мистецтва, наприклад, щоб продивитись 3d модель під різними картами запікання (див. розділ 1.1), щоб потенційний покупець міг оцінити якість розробки або роботодавець вміння дизайнера.

Three.js розроблений на основі бібліотеки WebGL. WebGL – це програмна бібліотека JavaScript, яка дозволяє створювати 3D графіку, яка функціонує в браузері. Ця бібліотека заснована на архітектурі бібліотеки OpenGL. WebGL використовує мову програмування шейдерів GLSL, яка має C-подібний синтаксис. Бібліотека цікава тим, що код моделюється в самому браузері. Для цього WebGL використовує об'єкт canvas, який був введений в HTML5.

Робота з WebGL, і з шейдерами зокрема, це досить трудомісткий процес. У процесі розробки необхідно описати кожну точку, лінію, грань тощо. Щоб все це візуалізувати, потрібно прописати досить об'ємний шматок коду. Для підвищення швидкості розробки було розроблено бібліотеку Three.js.

Three.js можна використовувати з різними бібліотеками, такі як React. На даний момент є хороший пакет розробки react-three-fiber. Для новачків в 3d розробці на основі JavaScript рекомендується використовувати бібліотеку Vanila JS, тому що більшість онлайн матеріалів, написаних про Three.js вілносяться до Vanila JS.

З цього можна зробити висновок, що світ веб-розробки розвивається і для кращої оптимізації продукту нативні мови програмування без додаткових бібліотек майже не використовують. Це є великий плюс в розробці нових програмних рішень, так як велике ком'юніті розробників дає змогу швидше розробляти нові технології і покращувати старі.

1.5 Мета, задачі та вимоги до реалізації інформаційної системи

Метою кваліфікаційної роботи бакалавра є створення й програмна реалізація методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів.

Для прикладного тестування методу персоналізації підбору компонентів інтернет-контенту слід виконати його програмну реалізацію у вигляді веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва, що виконує наступні основні функції:

1. Робота користувачів з базою цифрових графічних творів мистецтва:

– навігація в системі (реєстрація, авторизація, кабінет);

– формування власного портфоліо цифрових графічних творів мистецтва;

– привласнення тегів роботам з власного портфоліо цифрових графічних творів мистецтва;

– перегляд портфоліо інших користувачів;

– фільтрування робіт інших користувачів за критеріями автора, тегів, року;

– проходження анкетування для формування індивідуального вектору ознак за тегами;

– фільтрування робіт інших користувачів за індивідуальним вектором ознак.

2. Забезпечення анкетування користувачів та автоматизоване формування індивідуального вектору ознак за тегами:

– рекурсивний вибір випадкових зображень в процесі тестування;

– рекурсивна корекція індивідуального вектору ознак в процесі тестування;

– ведення «чорного списку» опрацьованих цифрових графічних творів мистецтва;

– формування індивідуального вектору ознак за результатом тестування.

Для досягнення мети створення й програмної реалізації методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів слід вирішити задачі:

– Провести аналіз предметної області та проблеми персоналізованого підбору компонентів інтернет-контенту.

– Розробити метод персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів.

– Виконати проєктування веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва на базі методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів.

– Здійснити вибір засобів розробки веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва.

– Розробити програмну реалізацію методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами

анкетування зразками компонентів у вигляді веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва.

– Провести тестування створеного веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва.

Розділ 2 Проєктування інформаційної системи

2.1 Метод персоналізованого підбору компонентів інтернет-контенту

Для реалізації веб-сервісу для перегляду цифрових графічних творів мистецтва була створена схема методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів, що зображена на рисунку 2.1 і виконує наступні функції:

– Роботу користувачів з базою цифрових графічних творів мистецтва і базою користувачів та їх даних.

– Рекурсивне анкетування користувачів та автоматизоване формування індивідуального вектору ознак за тегами.

Програма приймає як вхідні дані базу цифрових графічних творів мистецтва авторів, базу користувачів та їх даних, перелік тегів для формування індивідуального підбору цифрових графічних творів мистецтва для користувачів і вкінці повертає ставлення користувача до категорій у вигляді індивідуального вектору ознак за тегами, яким є новосформований список цифрових графічних творів мистецтва.

Робота користувачів з базою цифрових графічних творів мистецтва передбачає навігацію користувача в системі (вхід, реєстрація, фільтрація, редагування кабінету та робіт), формування власного портфоліо цифрових графічних творів мистецтва шляхом редагування свого портфоліо в кабінеті користувача та привласнення тегів своїм роботам перед публікацією, перегляд та фільтрування робіт інших користувачів за критеріями (автор, назва роботи, рік, тег), проходження анкетування для формування індивідуального вектору ознак за тегами для подальшого фільтрування цифрових графічних творів мистецтва інших авторів за цим вектором ознак та редагування індивідуального вектору ознак шляхом ігнорування нецікавого контенту під час рекурсивного анкетування користувача та скидання налаштувань до початкового вигляду.

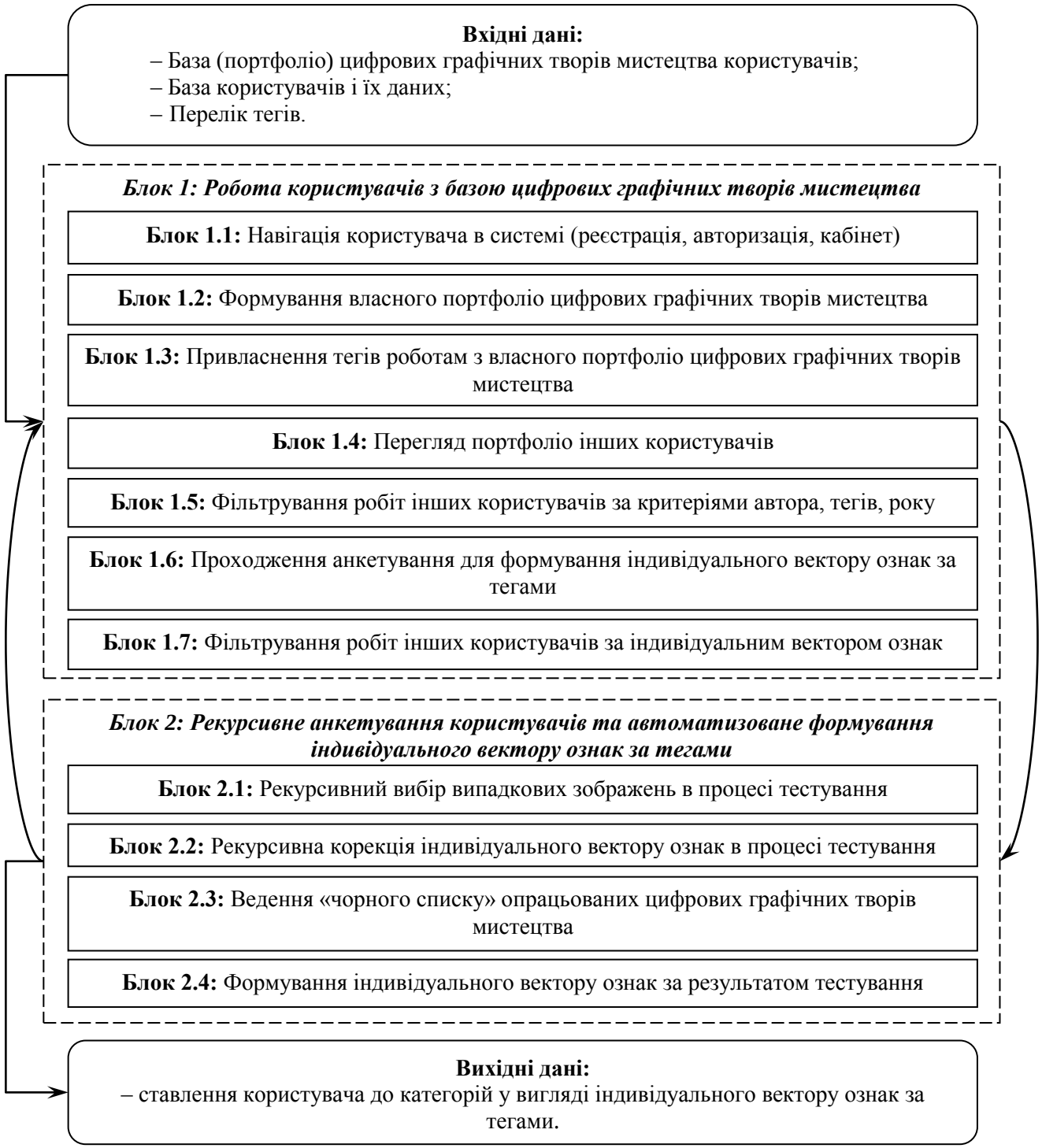


Рисунок 2.1 – Схема методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів

Взаємодія користувачів з базою цифрових графічних творів мистецтва та базою користувачів та їх даних, передбачає збільшення кількості можливих тегів в базі цифрових графічних творів мистецтва шляхом публікацій робіт користувачами, що поліпшує роботу персоналізованого підбору компонентів

інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів (тегів) для точнішого формування індивідуального вектору ознак за тегами, що виконує наступні функції під час рекурсивного анкетування користувача: рекурсивний вибір випадкових зображень в процесі тестування, рекурсивна корекція індивідуального вектору ознак в процесі тестування шляхом відкидання нецікавого контенту або скидання всіх тегів, ведення “Чорного списку” опрацьованих цифрових графічних творів мистецтва для відкидання нецікавого для користувача контенту та формування індивідуально вектору ознак за результатами тестування.

Результатом рекурсивного анкетування користувачів та автоматизованого формування індивідуального вектору ознак є новостворений список цифрових графічних творів мистецтва в якому є тільки ті теги, які пройшли процес анкетування.

2.2 Інформаційна структура системи

2.2.1 Проектна архітектура системи та взаємозв’язок компонентів

Перед програмним тестуванням методу персоналізації підбору компонентів інтернет-контенту слід створити проектну архітектуру інформаційної системи у вигляді схеми редагування бази цифрових графічних творів мистецтва, що зображена на рисунку 2.2.

Проектна архітектура інформаційної системи цифрових графічних творів мистецтва поділяється на дві підсистеми, що виконують наступні функції:

– Рекурсивне анкетування користувачів та автоматизоване формування індивідуального вектору ознак за тегами.

– Роботу користувачів з базою цифрових графічних творів мистецтва і базою користувачів та їх даних.

Підсистема роботи користувачів з базою цифрових графічних творів мистецтва і базою користувачів та їх даних призначена для опрацювання нових даних, що з’являються після реєстрації нового користувача або публікації нової

роботи. З публікацією нового контенту користувачами збільшується кількість можливих тегів для рекурсивного анкетування користувачів та автоматизованого формування індивідуального вектору ознак, за що відповідає друга підсистема.

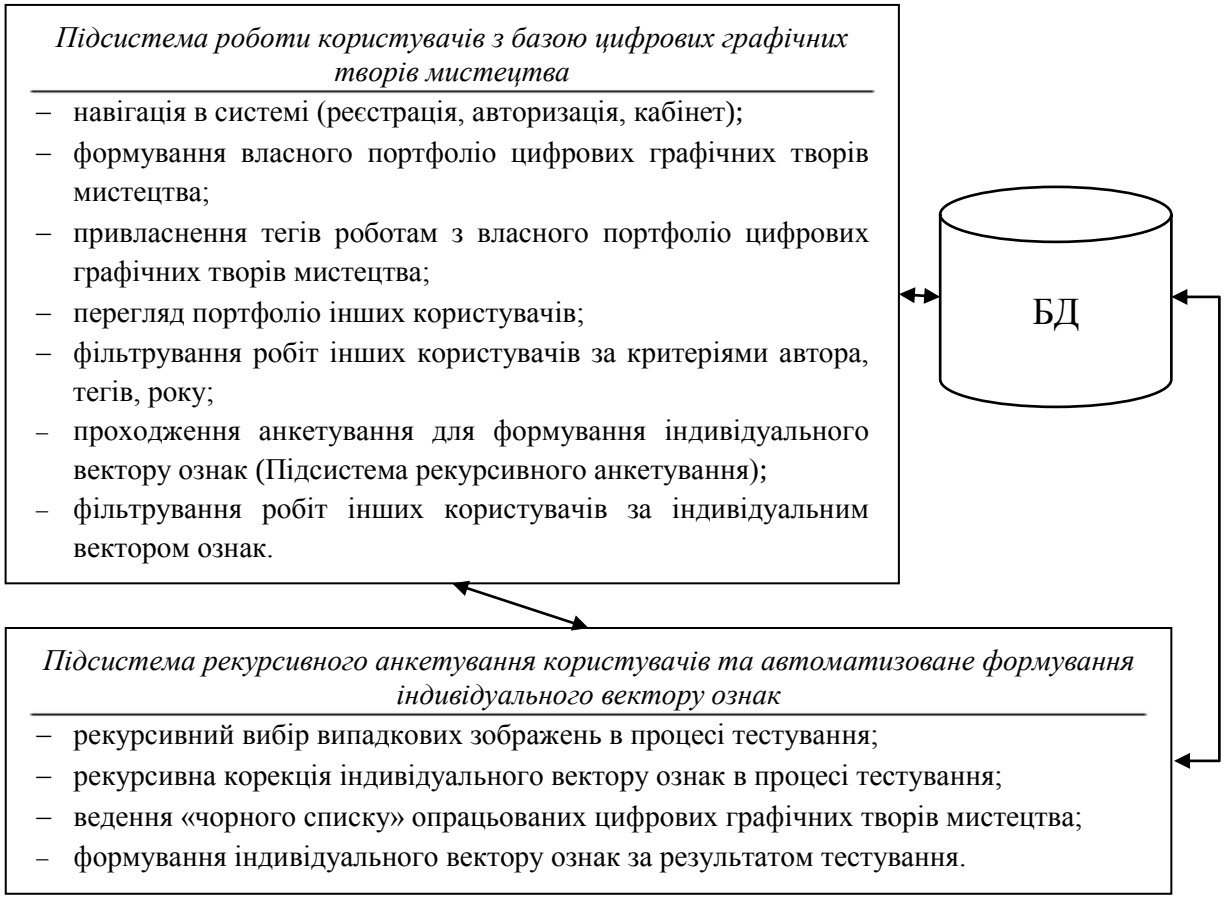


Рисунок 2.2 – Схема інформаційної системи цифрових графічних творів мистецтва

З рисунку 2.2 видно, що з підсистеми роботи користувачів з базою цифрових графічних творів мистецтва передаються дані для анкетування по тегам в підсистему рекурсивного анкетування користувачів. В свою чергу для роботи рекурсивного анкетування потрібна заповнена контентом база даних. Як видно з схеми при реєстрації і публікації роботи користувача, дані передаються в БД і далі можуть бути використані при рекурсивному анкетуванню користувачів. Звідки формується новий список вподобань користувача після анкетування і передається в БД для збереження при іншому вході на веб-сервіс.

2.2.2 Інформаційна модель

Для забезпечення коректної роботи веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва, необхідно створити базу даних, даталогічна модель якої зображена на рисунку 2.3.

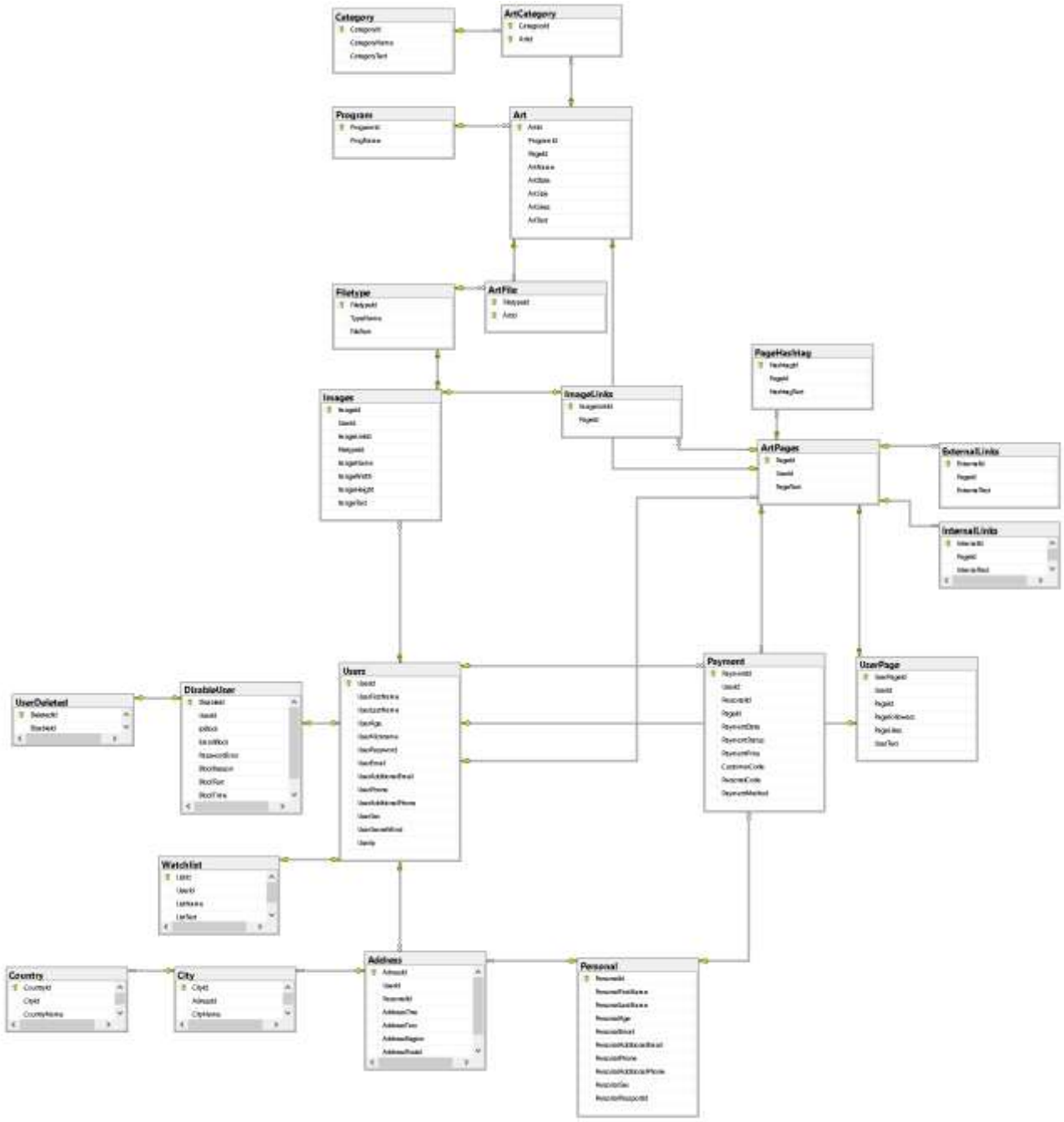


Рисунок 2.3 – Даталогічна модель веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва

Таблиця «Users» (таблиця 2.1) призначена для збереження в базі даних інформацію про користувача. Вся інформація відображається в портфоліо користувача і частина (рік, назва роботи, автор) при відображені в каталозі.

Таблиця 2.1 – Атрибути таблиці «Users»

Назва атрибуту	**Тип даних**	**Опис**
UserId	int	Ідентифікатор користувача
UserFirstName	nvarchar(128)	Ім’я
UserLastName	nvarchar(128)	Прізвище
UserAge	int	Вік
UserNickname	varchar(64)	Ім’я профілю
UserPassword	varchar(64)	Пароль
UserEmail	nvarchar(128)	Пошта
YserAdditionalEmail	nvarchar(128)	Додаткова пошта
UserPhone	nvarchar(128)	Телефон
UserAdditionalPhone	nvarchar(128)	Додатковий телефон
UserSex	nvarchar(16)	Стать
UserSecretWord	varchar(32)	Таємне слово
UserIp	varchar(64)	Айпі пристрою

Таблиця «WatchList» (таблиця 2.2) призначена для збереження тем після анкетування користувача. Користувач може вибрати одну з відсортованих тем з тегами для відображення необхідного контенту в каталозі.

Таблиця 2.2 – Атрибути таблиці «Watchlist»

Назва атрибуту	**Тип даних**	**Опис**
ListId	int	Ідентифікатор теми
UserId	int	Ідентифікатор користувача
ListName	varchar(64)	Назва теми
ListText	nvarchar(128)	Опис теми

Таблиця «UserPage» (таблиця 2.3) призначена для відображення сторінок з портфоліо користувача.

Таблиця 2.3 – Атрибути таблиці «UserPage»

Назва атрибуту	**Тип даних**	**Опис**
UserPageId	int	Ідентифікатор сторінки користувача
UserId	int	Ідентифікатор користувача
PageId	int	Ідентифікатор сторінки
PageFollowers	bigint	Кількість підписників
PageLikes	bigint	Вподобання роботи на одній сторінці
UserText	varchar(5000)	Опис сторінки

Таблиця «UserDeleted» (таблиця 2.4) показує інформацію про видалених користувачів

Таблиця 2.4 – Атрибути таблиці «UserDeleted»

Назва атрибуту	**Тип даних**	**Опис**
DeletedId	int	Ідентифікатор користувача
DisableId	int	Ідентифікатор видаленого або заблокованого користувача

Таблиця «Program» (таблиця 2.5) показує назви програм за допомогою яких була виконана робота.

Таблиця 2.5 – Атрибути таблиці «Program»

Назва атрибуту	**Тип даних**	**Опис**
ProgramId	int	Ідентифікатор програми
ProgName	varchar(64)	Назва програми

Таблиця «Personal» (таблиця 2.6) зберігає інформацію про робочий персонал на сайті. Таблиця «Payment» (таблиця 2.7) зберігає інформацію про всі транзакції на веб-сервісі.

Таблиця 2.6 – Атрибути таблиці «Personal »

Назва атрибуту	**Тип даних**	**Опис**
PersonalId	int	Ідентифікатор працівника
PersonalFirstName	nvarchar(128)	Ім’я
PersonalLastName	nvarchar(128)	Прізвище
PersonalAge	int	вік
PersonalEmail	nvarchar(128)	пошта
PersonalAdditionalEmail	nvarchar(128)	Дод. пошта
PersonalPhone	nvarchar(128)	телефон
PersonalAdditionalPhone	nvarchar(128)	Дод. телефон
PersonalSex	nvarchar(16)	стать
PersonalPassportId	nvarchar(32)	Ідентифікатор паспорта

Таблиця 2.7 – Атрибути таблиці «Payment»

Назва атрибуту	**Тип даних**	**Опис**
PaymentId	int	Ідентифікатор номеру оплати
UserId	int	Ідентифікатор користувача
PersonalId	int	Ідентифікатор працівника
PageId	int	Ідентифікатор сторінки покупки
PaymentDate	nvarchar(64)	дата
PaymentStatus	nvarchar(64)	Статус оплати
PaymentPrice	decimal(18, 0)	Ціна
CustomerCode	nvarchar(64)	Код користувача, що купував
PersonalCode	nvarchar(64)	Код відповідального працівника
PaymentMethod	nvarchar(64)	Метод оплати

Таблиця «PageHashtag» (таблиця 2.8) зберігає всі описи загальних тегів. Це тип тегів, що ділять роботи по жанрах і не створюються користувачами.

Таблиця 2.8 – Атрибути таблиці «PageHashtag»

Назва атрибуту	**Тип даних**	**Опис**
HashtagId	int	Ідентифікатор сторінки
PageId	int	Ідентифікатор сторінки пов’язаної з хештегом
HashtagText	nvarchar(128)	Опис хештегу

Таблиця «Image» (таблиця 2.9) призначена для зберігання інформації про цифрові графічні твори мистецтва.

Таблиця 2.9 – Атрибути таблиці «Image»

Назва атрибуту	**Тип даних**	**Опис**
ImageId	int	Ідентифікатор картинки
UserId	int	Ідентифікатор корисувача
ImageLinkId	int	Посилання на картинку
FileTypeId	int	Тип файлу
ImageName	varchar(64)	назва
ImageWidth	varchar(64)	ширина
ImageHeight	varchar(64)	висота
ImageText	varchar(128)	Опис

Таблиця «Image» (таблиця 2.10) призначена для посилань на роботи авторів

Таблиця 2.10 – Атрибути таблиці «ImageLink»

Назва атрибуту	**Тип даних**	**Опис**
ImageLinkId	int	Ідентифікатор посилання
PageId	int	Ідентифікатор сторінки

Таблиця «InternalLinks» (таблиця 2.11) зберігає посилання на внутрішні сторінки веб-сервісу. Використовується адмінами при публікації нової інформації на сайті і всі внутрішні посилання з публікації записуються в БД.

Таблиця 2.11 – Атрибути таблиці «InternalLinks»

Назва атрибуту	**Тип даних**	**Опис**
InternalId	int	Ідентифікатор посилання
PageId	int	Ідентифікатор сторінки
InternalText	nvarchar(128)	посилання

Таблиця «FileType» (таблиця 2.12) описує тип файла роботи або зображення

Таблиця 2.12 – Атрибути таблиці «FileType»

Назва атрибуту	**Тип даних**	**Опис**
FileTypeId	int	Ідентифікатор файлу
TypeName	nvarchar(128)	Тип файлу
FileText	nvarchar(128)	Опис

Таблиця «ExternalLinks» (таблиця 2.13) призначена для зберігання посилань на зовнішні джерела

Таблиця 2.13 – Атрибути таблиці «ExternalLinks»

Назва атрибуту	**Тип даних**	**Опис**
ExternalId	int	Ідентифікатор посилання
PageId	int	Ідентифікатор сторінки
ExternalText	nvarchar(128)	Опис

Таблиця «DisableUser» (таблиця 2.14) зберігає інформацію про заблокованих користувачів.

Таблиця 2.14 – Атрибути таблиці «DisableUser»

Назва атрибуту	Тип даних	Опис
DisableId	int	Ідентифікатор неактивного користувача
UserId	int	Ідентифікатор користувача
IpBlock	int	Айпі
EmailBlock	nvarchar(128)	Заблокована пошта
PasswordError	int	Використані спроби введення паролю
BlockReason	nvarchar(128)	Причина блокування
BlockText	nvarchar(128)	Опис причини
BlockTime	date/time	Час блокування

Таблиця «Country» (таблиця 2.15) зберігає інформацію про країну користувачів та працівників.

Таблиця 2.15 – Атрибути таблиці «Country»

Назва атрибуту	Тип даних	Опис
CountryId	int	Ідентифікатор країни
CityId	int	Ідентифікатор міста
CountryName	nvarchar(128)	Назва країни

Таблиця «Category» (таблиця 2.16) зберігає інформацію категорії в якій знаходиться тег.

Таблиця 2.16 – Атрибути таблиці «Country»

Назва атрибуту	Тип даних	Опис
CategoryId	int	Ідентифікатор категорії
CategoryName	varchar(64)	Назва категорії
CategoryText	nvarchar(128)	Опис категорії

Таблиця «City» (таблиця 2.17) зберігає інформацію про місто користувачів та працівників.

Таблиця 2.17 – Атрибути таблиці «City»

Назва атрибуту	**Тип даних**	**Опис**
CityId	Int	Ідентифікатор міста
AddressId	int	Ідентифікатор адреси проживання
CityName	nvarchar(128)	Назва міста

Таблиця «ArtPages» призначена для представлення контенту на сторінці роботи. (таблиця 2.18)

Таблиця 2.18 – Атрибути таблиці «ArtPages»

Назва атрибуту	**Тип даних**	**Опис**
PageId	int	Сторінка посилання
UserId	int	Ідентифікатор користувача
PageText	varchar(5000)	Опис сторінки

Таблиця «ArtFile» (таблиця 2.19) зберігає тип файлу роботи.

Таблиця 2.19 – Атрибути таблиці «ArtFile»

Назва атрибуту	**Тип даних**	**Опис**
FiletypeId	int	Ідентифікатор файлу
ArtId	int	Ідентифікатор роботи

Таблиця «ArtCategory» (таблиця 2.20) зберігає категорію роботи до якої належить.

Таблиця «Art» (таблиця 2.21) зберігає інформацію про саму роботу і відображається на сторінці цифрового графічного твору мистецтва.

Таблиця 2.20 – Атрибути таблиці «ArtCategory»

Назва атрибуту	**Тип даних**	**Опис**
CategoryId	int	Ідентифікатор категорії
ArtId	int	Ідентифікатор роботи

Таблиця 2.21 – Атрибути таблиці «Art»

Назва атрибуту	**Тип даних**	**Опис**
ArtId	int	Ідентифікатор роботи
ProgramId	int	Ідентифікатор програми
PageId	int	Ідентифікатор сторінки
ArtName	varchar(64)	Назва роботи
ArtDate	varchar(64)	Дата завантаження
ArtSize	varchar(64)	Розмір в пам’яті
ArtLikes	bigint	Вподобання
ArtText	varchar(5000)	Опис

Таблиця «Address» (таблиця 2.21) зберігає інформацію про домашню адресу користувачів та працівників.

Таблиця 2.22 – Атрибути таблиці «Address»

Назва атрибуту	**Тип даних**	**Опис**
AdressId	int	Ідентифікатор адреси
UserId	int	Ідентифікатор користувача
PersonalId	int	Ідентифікатор працівника
AdressOne	varchar(500)	Перша адреса
AdressTwo	varchar(500)	Друга адреса
AdressRegion	nvarchar(128)	Регион міста
AdressPostal	nvarchar(32)	Регіональний код

Таким чином була розроблена структура БД для подальшого її заповненням інформацією про цифрові графічні твори мистецтва, інформацію про користувачів та працівників веб-сервісу та веденням статистики за кількістю користувачів, тегів, цифрових графічних творів мистецтва та їхніх категорій.

2.3 Вибір засобів розробки інформаційної системи

2.3.1 Вибір мови розмітки

При розробці веб-сервісу для перегляду портфоліо цифрових графічних творів мистецтва були використані наступні мови програмування: мова розмітки HTML, мова стилів CSS та JavaScript.

HTML (HyperText Markup Language – мова гіпертекстової розмітки). Це самий базовий будівельний блок веб-сторінок. Він визначає вміст та структуру веб-контенту. Під гіпертекстом розуміють посилання, які об’єднують веб-сторінки один з одним або в межах одного веб-сайту, або в між багатьма веб-сайтами. HTML документ виділяється з іншого тексту в документі за допомогою тегів, які складаються з імені елементу вміщеного в дужки. Ім’я елементу всередині тегу не залежить від регістру. HTML розмітка використовує (“markup”) для відображення тексту, картинок та іншого контенту у веб-браузері. Інші технології, окрім HTML, використовують для описання зовнішнього вигляду сторінки CSS або поведінки JavaScript [17].

CSS (Cascading Style Sheets) – це мова ієрархічних правил (таблиця стилів), яка використовується для представлення зовнішнього вигляду документу, написаного на HTML або XML. CSS описує, яким чином елемент повинен відобразитись на екрані, на бумазі, голосом або з використанням інших медіа засобів [17].

JavaScript – це мова програмування, яка додає інтерактивності на ваш веб-сайт (наприклад: ігри, анімації при натисканні кнопок або при введені даних в форми, динамічні стилі, анімація). Це повноцінна динамічна мова програмування, яка застосовується до HTML документу і може забезпечити динамічну інтерактивність на веб-сайтах. Володіючи великим досвідом, можна створювати ігри, анімувати 2D та 3D графіку, повномасштабні додатки з базами даних тощо. JavaScript сам по собі доволі компактний, але дуже гнучкий. Розробниками написано велика кількість інструментів поверх основної мови,

різні фреймворки та бібліотеки, які розблоковують величезну кількість додаткових функцій з дуже невеликим зусиллям [18]. Такі як:

– Програмні інтерфейси API, вбудовані в браузери, забезпечуючи різноманітні функціональні можливості, такі як динамічне створення HTML установок CSS стилів, захоплення і маніпуляція відео-потоками, робота з веб-камерою користувача або генерація 3d графіки і аудіо семплів.

– Сторонні API дозволяють розробникам додавати функціональності в свої сайти розроблених іншими розробниками, такі як Twitter або Facebook.

– Можна застосовувати до HTML сторонні фреймворки і бібліотеки, що дозволяє прискорити процес створення сайтів та додатків.

Отож, для розробки всіх необхідних функцій для сайту-портфоліо достатньо цих трьох вище описаних мов.

2.3.2 Вибір комплексу програмних інструментів для керування веб-контентом

Створення веб-ресурсу – це комплекс заходів, об’єднуючих в собі розробку дизайну, інформаційне наповнення, використання вебу і маркетингових технологій, направлених на задоволення потреб користувачів майбутнього сервісу. Для керування веб-контентом та його створення був використаний комплекс програмних інструментів Chrome DevTools.

За допомогою Chrome DevTools можна змінювати анімацію, перевіряти доступність, знаходити помилки, слідкувати за продуктивністю сайту і виконувати багато інших завдань.

При створенню адаптивних сайтів або веб-додатків корисно знати, як буде виглядати сторінка на планшеті і мобільних пристроях. За допомогою інструментів розробника Chrome DevTools можна зробити це за декілька секунд. Для цього використовують функцію Toggle device toolbar. Над сторінкою з’явиться панель з режимами емуляції для мобільних пристроїв та планшетів. По замовчуванню панель інструментів відкривається в режимі адаптивного вікна

перегляду. Щоб змінити область до потрібних розмірів, можна задати ширину екрану або потягнути за робочі області. А якщо потрібно побачити, як сторінка відображається на інших пристроях, то використовують функцію Responsive, де потрібно вибрати підходящий девайс. На цій ж панелі є ще один інструмент – DPR. За його допомогою перевіряють, як буде виглядати зображення на ретіна-дисплеях (екранах з високою щільністю).

За допомогою цього комплексу інструментів в процесі розробки буває зручно змінювати стилі прямо в браузері. Наприклад, щоб перевірити, як виглядає елемент з новими CSS-правилами, або вирівняти його при верстці під PixelPerfect. Змінювати стилі можна у вкладці Elements. Спочатку потрібно вибрати елемент для редагування, а потім в розділі Styles додати, видалити або змінити стильові правила.

На вкладці Elements можна редагувати не тільки стилі, а також DOM-дерево: добавляти і видаляти елементи або блоки, змінювати текст, керувати атрибутами і класами. Це дуже зручно, особливо якщо потрібно протестувати якусь гіпотезу або перевіряти помилки в верстці.

При розробці сайту важливо думати про те, щоб він швидко завантажувався і був доступним для користувачів та пошукових систем. За допомогою інструменту Lighthouse можна протестувати свій сайт на швидкість, семантику, доступність, розмітку та інші характеристики. Lighthouse оцінює класичні сайти по чотирьох критеріях: продуктивність, SEO, кращі практики та доступність. Під час тестування інструмент буде змінювати розміри браузеру, імітувати відключення і підключення інтернету і виконувати інші операції. В кінці сайт отримує оцінку якості веб-ресурсу по 100-бальній шкалі. Чим вище оцінка, тим краще. При цьому на перевірку можуть впливати різні фактори, такі як доступність інтернету та версія розширення Chrome. Після перевірки, з'являються результати та показники, які вплинули на результати перевірки. Lighthouse може запропонувати оптимізувати картинки і шрифти, включити оптимізоване завантаження графіки, зменшити невикористаємий CSS і JavaScript або внести інші зміни.

Одне з завдань, яке було виконане за допомогою Chrome DevTools – тестування верстки на переповнення, тобто перевірка, як ведуть себе блоки та елементи при додаванні контенту або зміні розмірів шрифту. Chrome DevTools допомагає оптимізувати свій веб-ресурс та домогтися максимально можливої продуктивності.

2.3.3 Вибір фреймворку

На сьогоднішній день веб-технології дуже стрімко розвиваються і тому писати веб-сайти або додатки на нативному JavaScript не тільки робить роботу складнішою, а також гіршою і не достатньо оптимізованою. Тому для розробки сайту-портфоліо цифрових графічних творів мистецтва була використана бібліотека React для забезпечення хорошої оптимізації коду та зручного редагування коду шляхом розбиття його на частини (Компоненти).

React – JavaScript-бібліотека для роботи з користувацькими інтерфейсами UI. Потрібно лише описати, як різні частини інтерфейсу можуть виглядати у кожному стані вашої програми. React ефективно оновить необхідні частини коду, коли дані зміняться [19] .

За допомогою React розробники створюють веб-програми, які змінюють відображення без перезавантаження сторінки. Завдяки цьому програми швидко реагують на дії користувача, наприклад, заповнення форм, застосування фільтрів, додавання товарів у кошик і так далі.

React заснований на компонентах, це одна з ключових особливостей бібліотеки. Кожен компонент повертає частину користувацького інтерфейсу зі своїм станом. Важлива особливість React використання JSX. Це розширення синтаксису мови JavaScript, яку зручно використовувати для опису інтерфейсу. JSX схожий на HTML, тим не менш це все-таки JavaScript. JSX дозволяє писати JavaScript-код за допомогою готових компонентів, які практично повністю повторюють HTML.

До важливих особливостей React відноситься використання віртуального DOM (Virtual DOM). DOM – це об’єкт, в якому зберігається інформація про стан інтерфейсу. При зміні стану, наприклад, після відправки форми або натискання кнопки, React розраховує різницю між старим і новим станом. Після цього бібліотека рендерить новий стан. Використання віртуального DOM дозволяє бібліотеці ефективно оновлювати реальний DOM [20].

React популярна бібліотека для роботи з користувацьким інтерфейсом. До її основних особливостей відноситься декларативність, компонентний підхід, універсальність, використання віртуального DOM та JSX.

2.3.4 Вибір редактора програмного коду

Visual Studio Code — безкоштовний та дуже популярний редактор коду від Microsoft.

Visual Studio Code – безкоштовний та дуже популярний редактор коду від Microsoft для Windows, Linux та MacOS. Позиціонується як легкий редактор коду для кроссплатформеної розробки веб та хмарних додатків. Включає в себе налагоджувач коду, інструменти для роботи з Git, підсвітку синтаксису, IntelliSence та інструменти для рефактиронгу. Має широкі можливості для налаштування: користувацькі теми, поєднання клавіш і файли конфігурації. Поширюється безкоштовно, розробляється як програмне забезпечення з відкритим початковим кодом.

Visual Studio Code включає в себе інтеграцію налаштувань з Sublime, Atom. Після встановлення перевіряє наявність інших IDE. Якщо pyfqltyf Sublime або Atom, то VS Code запропонує частково перенести налаштування і клавіші швидкого доступу в нього. VS Code має зручну систему розширень. Розширення використовуються для покращення роботи з кодом та автоматизації деяких дій, що дає більшу часу на роботу з самим кодом замість того, щоб налагоджувати систему вручну, а також VS Code розширення можуть замінювати деякі бібліотеки. У VS Code зручна робота з системою контролю версій Git. Робота з

нею така сама, як і Visual Studio. Інтегрований термінал, де можна працювати через нього з Git і відлагоджувати код. Наявне відображення коммітів, історії, віток тощо. Розумний пошук з можливістю заміни по різних критеріях. Для веб проектів вже інтегрований відладчик коду. Для інших необхідно встановити розширення. Відображення використаємих змінних при дебагінгу та помилок.

VS Code можна сміливо назвати самим кращим редактором коду. Редактор дуже гнучкий для додавання плагінів та простий у використанні, що робить його дуже популярним серед розробників.

Також для розробки інформаційної системи було використано мову програмування JavaScript, бібліотека React, а також систему керування базами даних MS SQL Server.

Розділ 3 Програмна реалізація інформаційної системи

3.1 Структура та функціональне призначення програмних складових системи

Схему структури та функціонального призначення програмних складових веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва наведено на рисунку 3.1. Веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва містить базовий клас веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва «App», який включає в себе ряд класів, які групуються у компоненти.

Зокрема, такими компонентами, що представлені на рисунку 3.1, є:

– компоненти фільтрування цифрових графічних творів мистецтва;

– компоненти представлення цифрових графічних творів мистецтва на сторінці;

– компоненти додавання цифрових графічних творів мистецтва в каталог;

– компоненти розбиття цифрових графічних творів мистецтва на категорії;

– компоненти рекурсивного анкетування користувачів для формування індивідуального вектору ознак цифрових графічних творів мистецтва.

Зокрема, компонентами фільтрування цифрових графічних творів мистецтва є програмний компонент пошуку цифрових графічних творів мистецтва за іменем “SearchByName”, який виконує пошук по імені, компонент пошуку цифрових графічних творів мистецтва за іменем автору, роком та тегами є компонент “SearchFilter”, компонент сортування цифрових графічних творів мистецтва “AppFilter” за назвою роботи, іменем автора та роком зверху вниз і знизу вверх.

Компонентами представлення цифрових графічних творів мистецтва на сторінці є компонент “CardList”, який виконує функцію повернення всіх

цифрових графічних творів мистецтва в каталог та компонент “CardItem”, який слугує для опису інформації кожного цифрового графічного твору мистецтва.

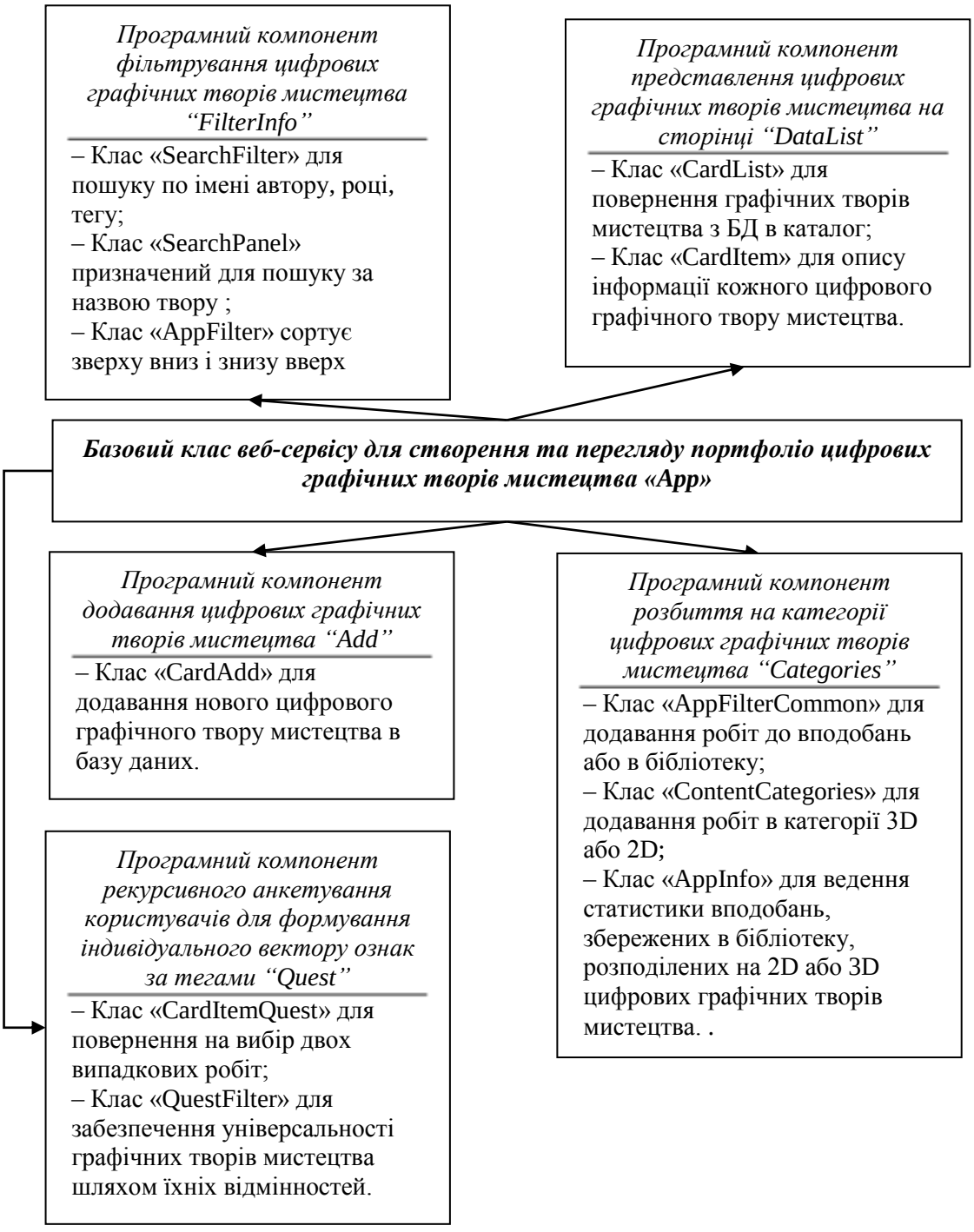


Рисунок 3.1 – Схема структури та функціонального призначення програмних складових веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва

Додавання цифрових графічних творів мистецтва в базу даних відбувається за допомогою компоненту “CardAdd”, який передає новоопублікований цифровий графічний твір мистецтва в головний клас “App”, а звідти в БД.

Компонентами розбиття на категорії цифрових графічних творів мистецтва зокрема є: компонент “AppFilterCommon”, який призначений для розбиття на категорії користувачем на вподобані та додані в бібліотеку цифрові графічні твори мистецтва, компонент “ContentCategories”, призначення якого розбиття цифрових графічних творів мистецтва на 3D та 2D та компонент “AppInfo”, що підраховує кількість вподобаних, доданих в бібліотеку, розділених на 2D та 3D цифрових графічних творів мистецтва.

Компонентами цифрових графічних творів мистецтва рекурсивного анкетування користувачів для формування індивідуального вектору ознак є: компонент “CardItemQuest”, призначення якого повернути на сторінку два випадкових цифрових графічних твори мистецтва за допомогою компоненту “QuestFilter”, що призначений для пошуку двох різних цифрових графічних творів мистецтва для забезпечення індивідуальності кожного шляхом підбирання персональних ключових слів (тегів) та забезпечення індивідуальності ідентифікатору. Новостворений список цифрових графічних творів мистецтва повертається в компонент “CardList”, в якому генерується індивідуальний вектор ознак цифрових графічних творів мистецтва за тегами.

В процесі розробки структури функціонального призначення системи була створена схема, що зображена на рисунку 3.1, для опису кожного компоненту системи цифрових графічних творів мистецтва та пояснення вмісту кожного компоненту для подальшої програмної реалізації.

3.2 Особливості реалізації програмних складових системи

Під час розробки веб-сервісу цифрових графічних творів мистецтва був зроблений пріоритет на створення багаторівневого фільтру для контенту в

каталозі та розробка рекурсивного анкетування користувачів для формування індивідуального вектору ознак цифрових графічних творів мистецтва за тегами. Була створена базова розмітка сторінки, що ділилась на дві підсистеми. Перша підсистема включала в себе фільтрацію, статистику та пошук контенту, а друга підсистема відображення контенту та його опис.

Після реалізації базової розмітки для системи цифрових графічних творів мистецтва почалась реалізація класів на мові скриптів JavaScript та для забезпечення розбиття класів на компоненти була застосована бібліотека React. Фільтрація повинна була враховувати багато чинників, а саме ім’я, рік, теги, автора, тип та категорію зображення. Зі збільшенням кількості різних фільтрів з’являлась більша вкладеність і складність реалізації структури фільтрування, так як кожен фільтр має більший пріоритет над іншим. Рекурсивне анкетування користувачів мало найбільший пріоритет для формування нового списку контенту в каталозі, фільтрування за іменем автору, роком, назвою цифрового графічного твору мистецтва тощо, мало найменший пріоритет при фільтруванні для забезпечення роботи програми без помилок.

```
const visibleData = this.filterQuest(this.filterCategories(this.filterPost(this.filterSort(this.searchArt(data, term, date,
dateMax, termAuthor, termTag), sortFilter), menuFilter), categories), questTags, questDat);
```

Кожен компонент програми має певну кількість методів, що відповідають за роботу з базою даних. Кожен метод приймає в себе значення з головного класу “App” де формується БД веб-сервісу. Кожен наступний метод викликається знизу вверх, а потім кожен чекає виконання своєї вкладеної функції. Змінні функції створюються в конструкторі основного класу “App”, потім кожна змінна передається в клас свого компоненту і повертається звідти назад в методи класу “App”, далі вони виконують кожна свою умову в своєму методі і повертають значення в конструктор класу “App”.

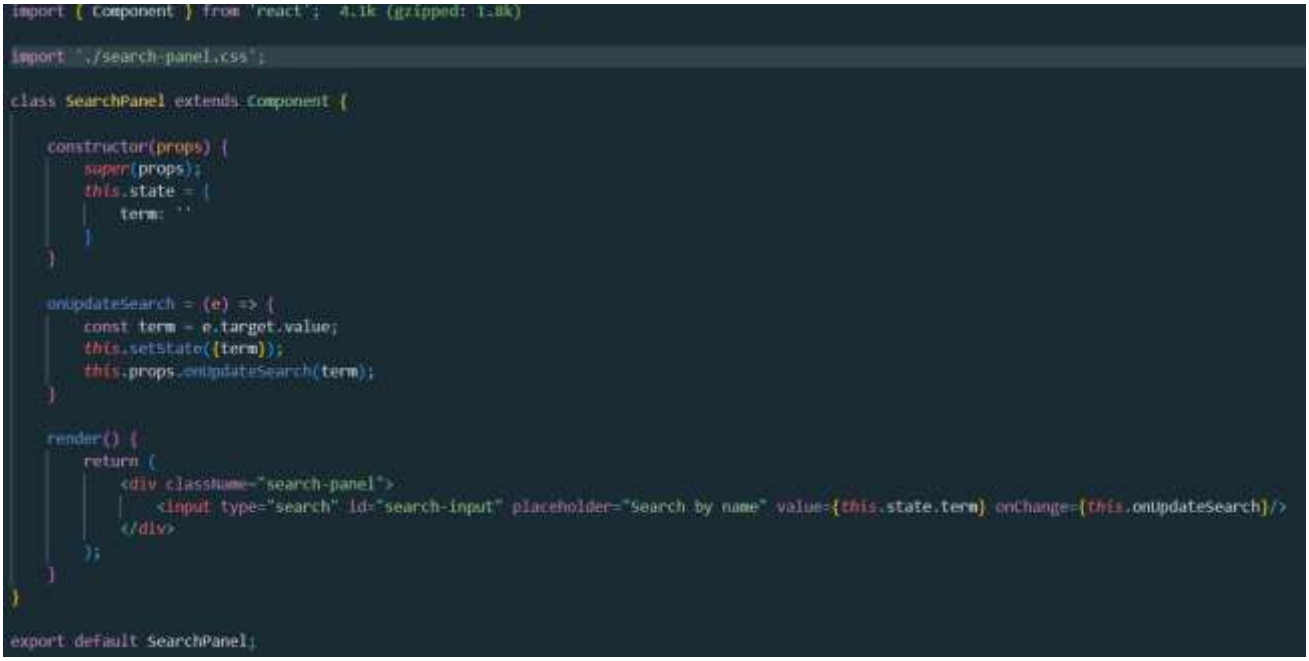
Клас “App” в свою чергу перебудовує конструктор і передає нові значення в компонент “CardList” (передається змінна “visibleData”) в якому будується нове представлення контенту. “CardList” далі передає свої дані в

CardItem (Одиниця контенту), де перевіряється чи відповідає кожна картка певним умовам, якщо так, то вона повертається в «CardList».

Рисунок 3.3 – Зліва компонент «CatdList», що повертає кожну одиницю компоненту «CardItem». Справа частина коду «CardItem», що повертається в конструктор класу «App»

Рисунок 3.4 – Повернення компоненту «CardList» в «App.js» при фільтруванні

Як видно з рисунку 3.4, контент було відфільтровано, а в клас “CardList” повернулись з бази даних тільки ті цифрові графічні твори мистецтва, що збігаються з кожним компонентом класу “CardItem”.



```
import { Component } from 'react';  4.1k (gzipped: 1.8k)

import './search-panel.css';

class SearchPanel extends Component {

    constructor(props) {
        super(props);
        this.state = {
            term: ''
        }
    }

    onUpdateSearch = (e) => {
        const term = e.target.value;
        this.setState({term});
        this.props.onUpdateSearch(term);
    }

    render() {
        return (
            <div className="search-panel">
                <input type="search" id="search-input" placeholder="Search by name" value={this.state.term} onChange={this.onUpdateSearch}/>
            </div>
        );
    }
}

export default SearchPanel;
```

Наприклад, клас “SearchPanel” приймає змінну term класу “App”. Змінна term використовується для зберігання значення поля вводу, що шукає по імені. Коли користувач щось ввів, то в «input» викликається функція “onUpdateSearch”, в якій значення «input» передається конструктору класу “SearchPanel”, а далі викликається метод класу “App”, що назначає цій самій змінній в своїй області видимості те саме значення. Щоб використати методи батьківського класу в компонент потрібно передати ці методи разом зі змінними, що там використовуються.

Інші фільтри по пошуку реалізуються аналогічно «SearchPanel», різниця тільки в пріорітеті, але щоб не було багів під час пошуку, потрібно реалізувати певні умови при яких треба повертати тільки певне значення. Це зроблено для того, щоб пофіксити деякі баги при виконанні. Наприклад, якщо всі поля будуть заповнені якимось значеннями, а поле імені і автора залишиться «null».

Рисунок 3.5 – Структура фільтрів пошуку і категорій, в кожен компонент передаються свої методи та змінні.

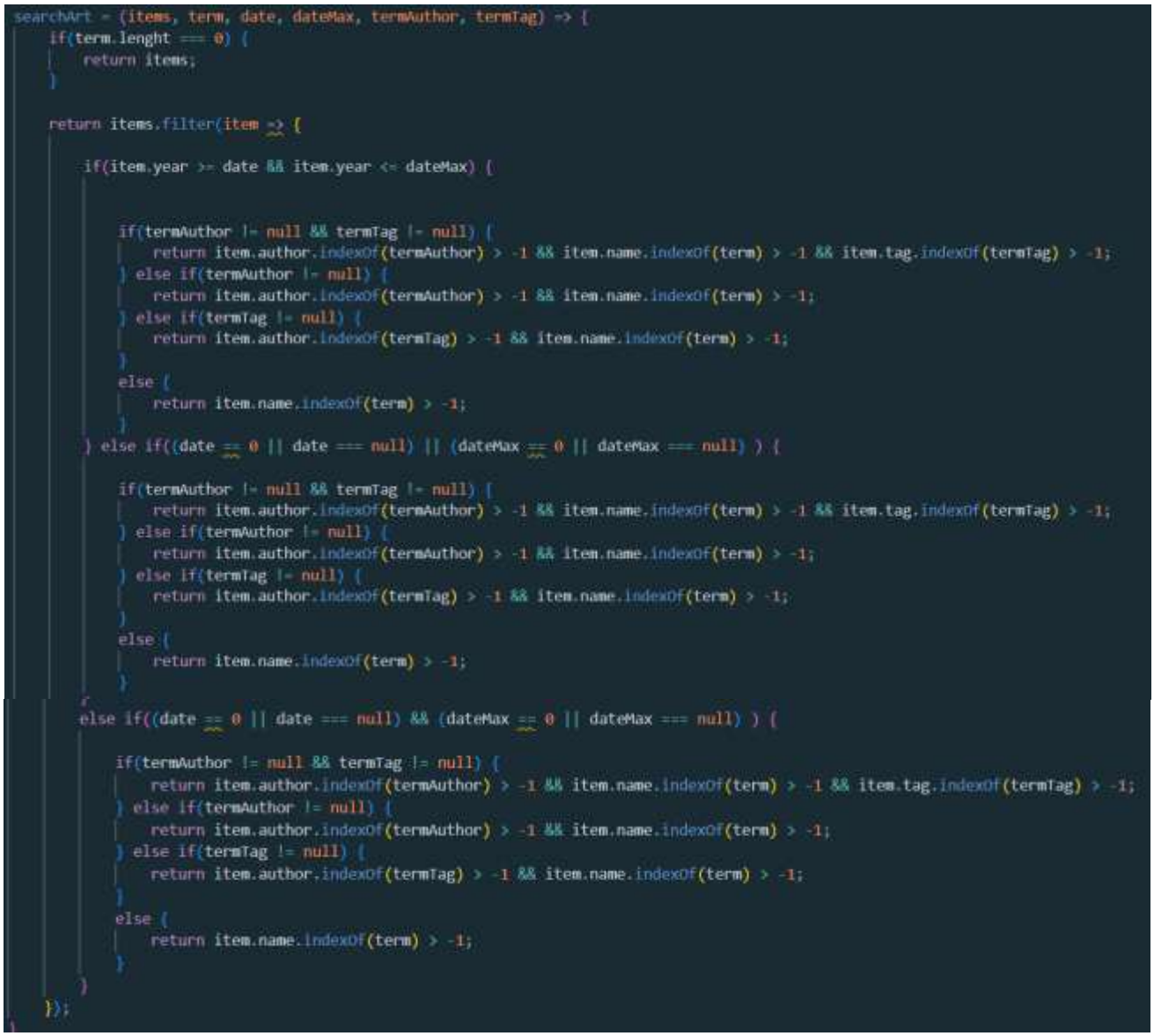


Рисунок 3.6 – Умови при яких повертаються різні значення при фільтруванні

Як видно з рисунку 3.6, якщо якісь значення відсутні, то ми переходим на наступну умову і перевіряєм фільтр на присутність і відсутність інших значень компонентів. Якщо якась умова вірна, то ми повертаємо це значення імені, автора, року тощо.

Наприклад, значення року. Якщо ми введемо дані тільки в поле де знаходиться мінімальне значення, то повернеться все, що більше за це значення, так само з полем більшого значення. Якщо ми залишимо ці поля пустими, то повертаються всі наявні роки, якщо ми фільтруємо також по іншим критеріям.

Якщо всі поля пусті, то повертається всі картки, що наявні в базі даних. Але така фільтрація не стосується сортування, анкетування, категорій карток (3D або Art) і категорій користувача (Вподобання або Бібліотека). В кожного з цих фільтрів своя реалізація. Всі вони фільтрують по наявній інформації, відфільтрованій або ні, окрім анкетування. Анкетування навпроти, створює новий список карток за допомогою тегів, по якому все буде фільтруватися.

Під час анкетування перед користувачем з’являється дві картинки. Він повинен вибрати одну на свій смак. Коли вибір зроблено то теги вибраних картинок переходять в масив, що створений в конструкторі класу “App”. Цей масив передається в метод “FilterQuest” і там перевіряється, чи теги наявних цифрових графічних творів мистецтва в базі даних співпадають з тегами в цьому масиві, якщо так, то повертається зображення з даним тегом, якщо ні, то зображення з бази даних не витягається. Далі створюється новий список, по критеріям якого буде проводитись фільтрування іншими способами.

При цьому виконується умова, що ідентифікатор картинки не співпадає сам собі, і що теги одного цифрового графічного твору мистецтва не співпадає повністю іншій. Це зроблено для забезпечення індивідуальності кожної з двох картинок при рекурсивному анкетуванні користувача.

Результат рекурсивного анкетування після якого було виведено тільки ті цифрові графічні твори мистецтва, які змогли пройти анкетування і відповідають тегам анкетування. Після чого дані цифрові графічні твори можуть піддаватись фільтруванню, під час якого весь інший каталог контенту буде прихована, так як

рекурсивне анкетування користувача має найбільший приорітет під час фільтрації.

На рисунку 3.7 зображено процес фільтрації по категоріях цифрового графічного твору мистецтва, категоріях бібліотеки та вподобань та по сортуванню каталогу зверху вниз і знизу вверх.

Рисунок 3.7 – Сортування та категорії

Можна додавати зображення в потрібні категорії під час перегляду робіт інших користувачів або під час публікації. В “3D” знаходяться зображення моделей, а в “Art” намальовані зображення, в “Like” вподобані зображення і в “Library” збережені зображення.

Аналогічно працюють дві категорії для користувача (Вподобання та Бібліотека), але якщо в простих категоріях значення вже наперед якесь створене і змінити його можна тільки, якщо користувач створює нову картку на сторінці або змінює свою картку в каталозі своїх зображень, то в цих категоріях користувач просто фільтує картки на вподобання і на зображення в бібліотеку. Відповідно для цього потрібно якось взаємодіяти з карткою.

У кожної картки крім базової інформації такої як зображення, ім’я, автор, рік, тег є також кнопки для взаємодії для переведення з одного стану в інший. На картці є дві кнопки – додати в бібліотеку і вподобати. При зміні категорії картки, змінюється колір полоси під зображення і колір кнопок категорій.

При натисканні на вподобання картка добавляється в вподобання, колір кнопок змінюється, полоска під зображенням теж і якщо перейти на цю вкладку там будуть чекати відфільтровані картки. Також можна добавити те саме

зображення в бібліотеку, і воно вже буде в двох категоріях. Якщо забрати з карток всі категорії, тоді там нічого не залишиться.

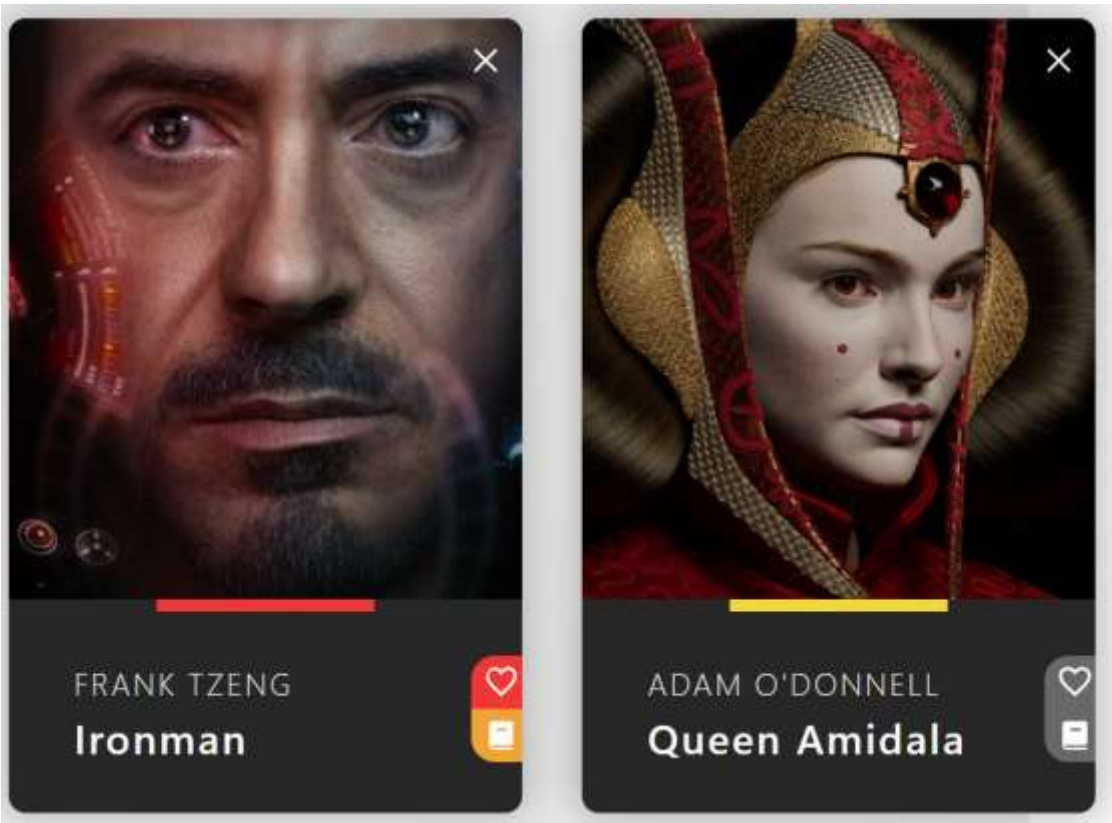


Рисунок 3.8 – Картки зображень з інформацією

```
const AppFilterCommon = (props) => {

    const buttonsData = [
        {name: 'all', label: allSvg},
        {name: 'likes', label: favouriteSvg},
        {name: 'myLib', label: librarySvg}
    ];

    const buttons = buttonsData.map(({name, label}) => {
        const active = props.menuFilter === name;

        const clazz = active ? 'app-filter-common__btn-item-active' : 'app-filter-common__btn-item-light';

        return (
            <button
                className={`app-filter-common__btn-item ${clazz}`}
                type="button"
                key={name}
                onClick={() => props.onFilterSelect(name)}>
                <img src={label} alt="" />
                </button>
        )
    })

    return (
        <div className="app-filter-common__btn-list">
            {buttons}
        </div>
    );
}

export default AppFilterCommon;
```

Рисунок 3.9 – Реалізація компоненту категорій користувача

На рисунку 3.9 показано, реалізація кнопок для переходу між категоріями. “buttonsData” генерує три кнопки, при натисканні на одну з них ім’я кнопки передається в змінну класу “App” через метод”onFilterSelect” і там в методі класу перевіряється на умову: *якщо категорія картки дорівнює категорії кнопки, то повертається картка з цією категорією.*

Сортування каталогу здійснюється по роках, авторам, назвам зверху вниз та знизу вверх, а також одна кнопка для повернення в звичайний стан.

```
filterPost = (items, menuFilter) => {

    switch (menuFilter) {
        case 'likes':
            return items.filter(item=> item.like);
        case 'myLib':
            return items.filter(item => item.increase);
        default:
            return items
    }
}
```

Рисунок 3.10 – Метод класу “App”. В “menuFilter” передається назва кнопки

Абсолютно аналогічно реалізуються категорії. Та сама умова, що і в категоріях користувача.

```
filterCategories = (items, categories) => {

    switch (categories) {
        case '3D':
            return items.filter(item=> {
                if(item.workType == 'Art') {
                    return item.workType;
                }
            });
        case 'Art':
            return items.filter(item=> {
                if(item.workType == '3D') {
                    return item.workType;
                }
            });
        default:
            return items
    }
}
```

Рисунок 3.11 – Метод класу “App”. В “categories” передається назва кнопки

Категорії картинок можуть приймати всі значення в одну змінну у вигляді рядка, коли категорії користувача мають дві різних змінних в базі данних, одна для лайків та інша для додавання в бібліотеку, і приймають значення true або false.

```
const {author, name, img, onDelete, onToggleIncrease, onToggleLike, increase, like} = props;

let categoryColor = 'category-color',
    chooseIcon = 'choose-icon-like',
    chooseLibs = 'choose-icon-lib';

if (like) {
    categoryColor += ' category-color-red';
    chooseIcon += ' choose-icon-like-active';
}

if(increase) {
    chooseLibs += ' choose-icon-lib-active';
}
```

Рисунок 3.12 – Методи для зміни значень

```
onToggleIncrease = (id) => {

    this.setState(({data}) => ({
      data: data.map(item => {
          if(item.id === id) {
              return {...item, increase: !item.increase}
          }

          return item;
      })
    }))
}

onToggleLike = (id) => {
    this.setState(({data}) => ({
      data: data.map(item => {
          if(item.id === id) {
              return {...item, like: !item.like}
          }

          return item;
      })
    }))
}
```

Рисунок 3.13 – Методи для зміни значень. При true або false змінюється задній колір картинок

В категоріях картинок значення заданні на перед і при створенні елементу користуачем значення потрібно вибирати тільки при створенні і вибрати категорію можна тільки одну. В той ж момент в категоріях користувача

можна добавити картку одночасно і у вподобавння, і в бібліотеку. Тому компонент CardItem для кожної картки має методи для переводу з однієї категорії в іншу. Потім ці значення передаються в базу даних (data) класу “App” і змінюються на true або false.

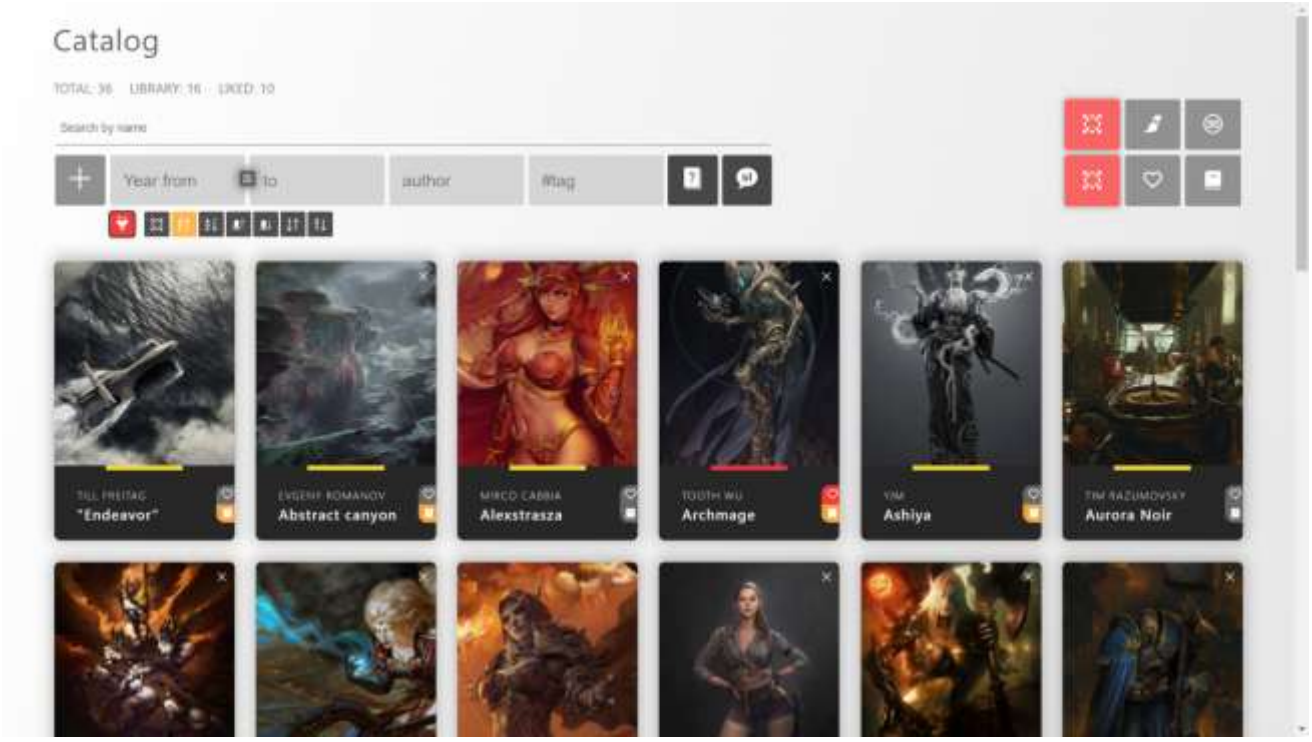


Рисунок 3.14 – Фільтрація по іменам картинок з початку до кінця

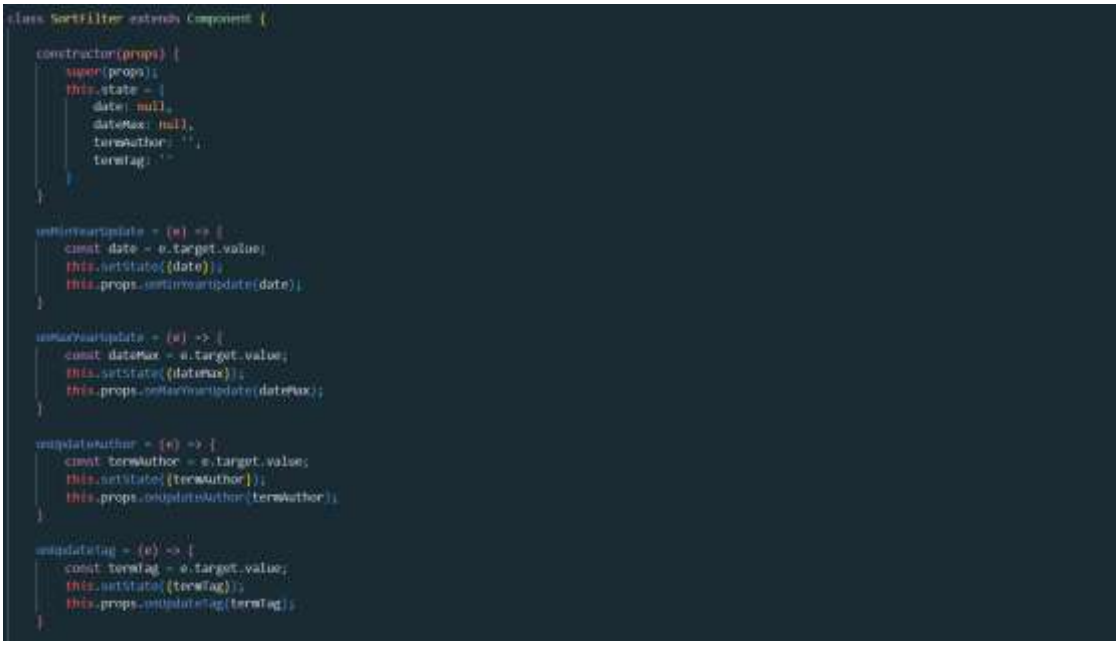


Рисунок 3.15 – Методи кнопки, що передаються у змінні класу App

Сортування контенту від попередньо згаданих функцій теж не відрізняється, різниця тільки в тому, що список генерується по інших умовах. Так само є сім кнопок і якщо ім’я однієї співпадає з умовою, то виконується сортування по певній умові.

```
filterSort = (items, sortFilter) => {

    switch(sortFilter) {
        case 'yearUp':
            return items.sort((a, b) => a.year < b.year ? 1 : -1);
        case 'yearDown':
            return items.sort((a, b) => a.year > b.year ? 1 : -1);
        case 'nameUp':
            return items.sort((a, b) => {
                let x = a.name.toLowerCase();
                let y = b.name.toLowerCase();
                return x < y ? -1 : x > y ? 1 : 0;
            });
        case 'nameDown':
            return items.sort((a, b) => {
                let x = a.name.toLowerCase();
                let y = b.name.toLowerCase();
                return x > y ? -1 : x < y ? 1 : 0;
            });
        case 'authorUp':
            return items.sort((a, b) => {
                let x = a.author.toLowerCase();
                let y = b.author.toLowerCase();
                return x < y ? -1 : x > y ? 1 : 0;
            });
        case 'authorDown':
            return items.sort((a, b) => {
                let x = a.author.toLowerCase();
                let y = b.author.toLowerCase();
                return x > y ? -1 : x < y ? 1 : 0;
            });
        default:
            return items
    }
}
```

Рисунок 3.16 – Умови фільтрування

В результаті розробки програмного забезпечення в розділі 3.2 були представлені головні функції інформаційної системи такі як: багаторівнева фільтрація цифрових графічних творів мистецтва та рекурсивне анкетування користувачів шляхом підбору універсального набору ознак за тегами. Під час розробки програмного забезпечення всі помилки були максимально виправлені для нормального функціонування веб-сервісу.

3.3 Тестування інформаційної системи

Для тестування інформаційної системи було створено два тест-кейси: перший тест-кейс, що зображений в таблиці 3.1, для тестування програмної реалізації методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів для цифрових графічних творів мистецтва, другий тест-кейс, що зображений в таблиці 3.2, для тестування багаторівневої фільтрації цифрових графічних творів мистецтва шляхом підбору цікавих для користувача критеріїв. Для тестування була використана база даних веб-сервісу цифрових графічних творів мистецтва, що зображена на рисунку 3.17.

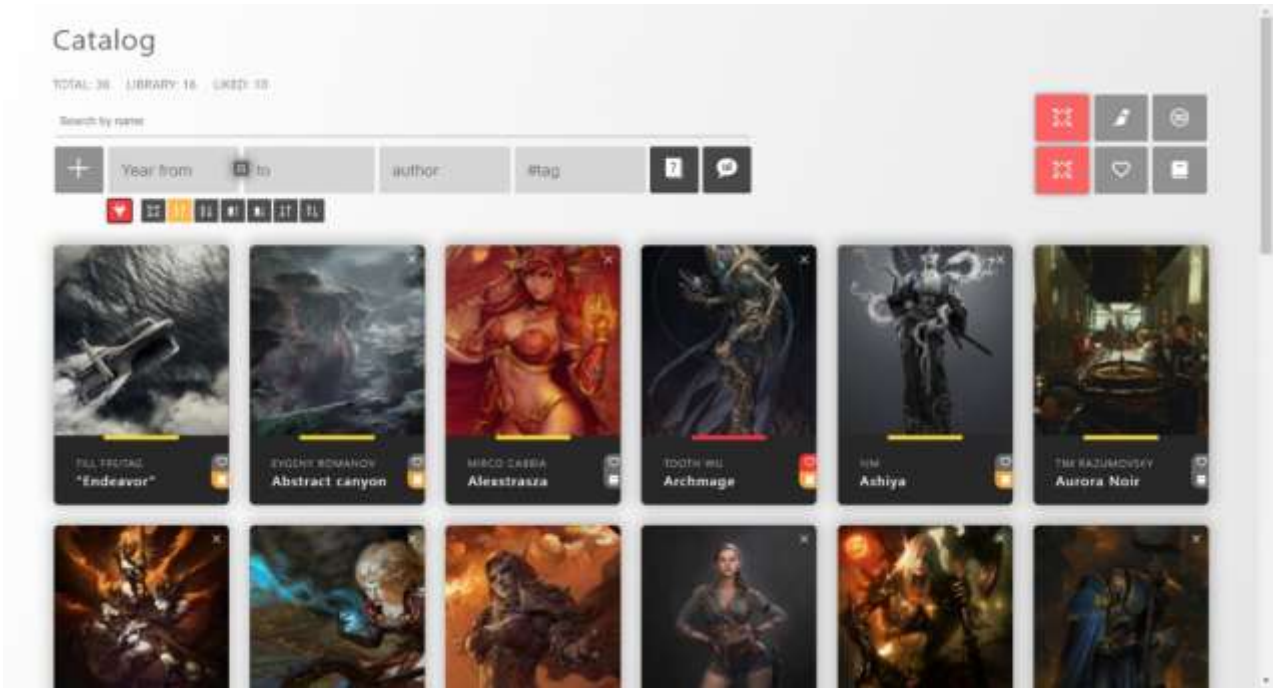


Рисунок 3.17 – Каталог цифрових графічних творів мистецтва

Рекурсивне анкетування користувача шляхом формування індивідуального вектору ознак за тегами зображено на рисунку 3.18. У першому тестуванню проводиться тестування методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів для цифрових графічних творів мистецтва. Для цього

користувач переходить на вкладку анкетування, де теги розподіляються по кластерам для формування тегів для рендерингу на сторінці і тегів для додавання в “Чорний список”, тестування AT0001 в таблиці 3.1.

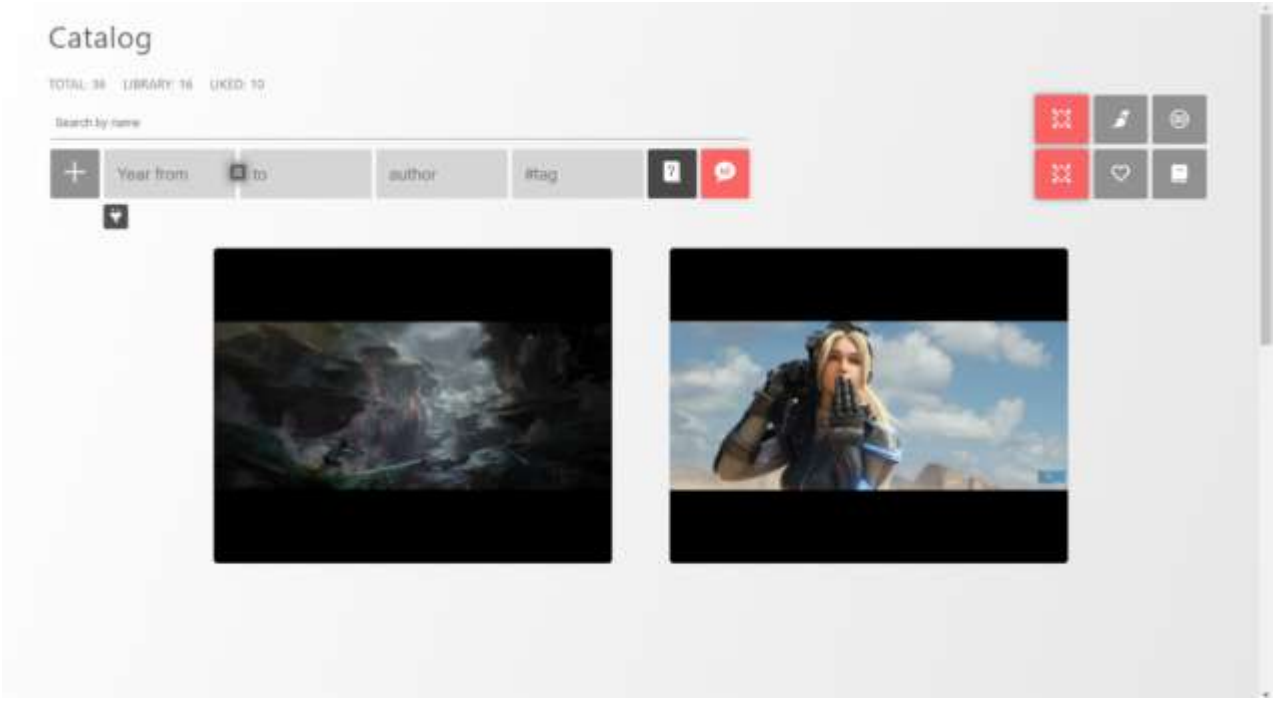


Рисунок 3.18 – Рекурсивне анкетування шляхом підбору двох зображень

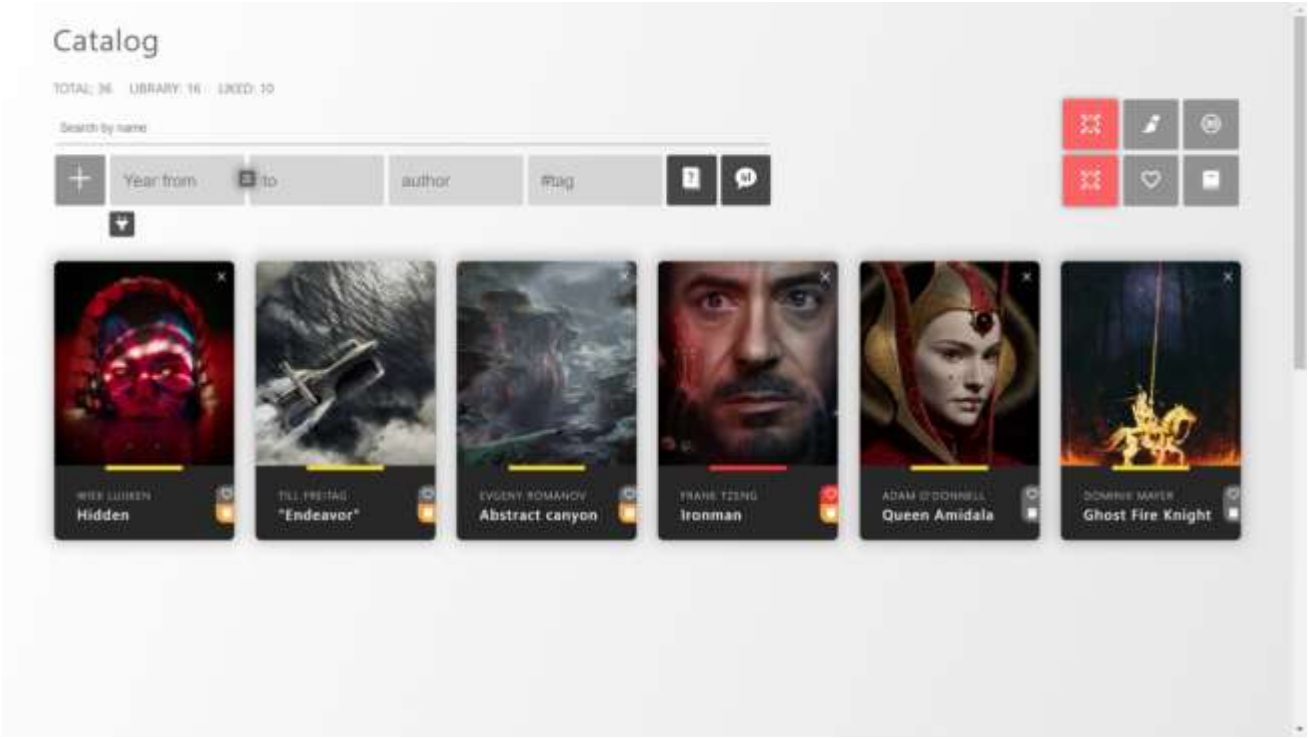


Рисунок 3.19 – Результат рекурсивного анкетування користувача

На рисунку 3.19 зображено результат методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів для цифрових графічних творів мистецтва. На сторінці були виведені тільки ті роботи у яких присутні теги анкетування.

Таблиця 3.1 – Тест-кейс AT0001

<table>
<tr><td colspan="2">Тест-кейс ID: AT0001</td><td>Пріоритет: 1</td><td>Створено: 09.06.2022</td></tr>
<tr><td colspan="4">Назва: рекурсивне анкетування користувача для формування індивідуального вектору ознак за тегами
Вхідні дані: набір випадкових цифрових графічних творів мистецтва під час рекурсивного анкетування користувача</td></tr>
<tr><td colspan="2">Кроки</td><td colspan="2">Результат</td></tr>
<tr><td colspan="2">Передумова: користувач, що знаходиться в каталозі переходить на вкладку рекурсивного тестування для підбору індивідуального набору тегів для пошуку цифрових графічних творів мистецтва під свої побажання
1. Увійти на веб-сервіс
2. Перейти в каталог цифрових графічних творів мистецтва
3. Натиснути на іконку тестування
4. Почати тестування шляхом підбору цікавих користувачу зображень
5. Пропустити нецікаві випадкові зображення шляхом натискання на кнопку “Change”
6. Закінчити тестування у будь-який момент
7. Перейти у каталог
8. Відфільтрувати підібраний контент по бажаним критеріям (рік, назва роботи, ім’я автору, сортування, випадкова робота, пошук конкретного тегу з вибраних)
9. Скинути всі налаштування анкетування шляхом натискання кнопки “Reset”</td><td colspan="2">Користувач пройшов рекурсивне анкетування для формування індивідуального вектору ознак за тегами

Теги, що пройшли тесування, були додані в окремий кластер для відображення необхідних елементів на сторінці. Інші теги булт додані в “Чорний список”

Користувач відфільтрував бажаний контент за своїм смаком: роком, назвою роботи, іменем автору, сортуванням, випадквою роботою, пошуком по конкретних тегах.

Користувач скинув всі налаштування анкетування, щоб повернути всі роботи в каталог або пройти анкетування знову.</td></tr>
<tr><td colspan="4">Результат тест-кейсу: успішно, помилок виявлено не було</td></tr>
</table>

Для тестування багаторівневої фільтрації цифрових графічних творів мистецтва шляхом підбору цікавих для користувача критеріїв була використана база даних веб-сервісу цифрових графічних творів мистецтва. Багаторівнева фільтрація проводиться за пріоритетом кожного класу фільтрування. Найвищий пріоритет у пошуку за тегами і найменший за пошуком по назві роботи. Тестування АТ002 зображене в таблиці 3.2.

Таблиця 3.2 – Тест-кейс АТ0002

<table>
<tr><td>Тест-кейс ID: АТ0002</td><td>Пріоритет: 2</td><td>Створено: 09.06.2022</td></tr>
<tr><td colspan="3">Назва: багаторівневе фільтрування цифрових графічних творів мистецтва шляхом підбору компонентів по бажаним критеріям
Вхідні дані: база даних цифрових графічних творів мистецтва</td></tr>
<tr><td colspan="2">Кроки</td><td>Результат</td></tr>
<tr><td colspan="2">Передумова: користувач переходить на вкладку “Каталог” для пошуку цікавого йому контенту
1. Увійти на веб-сервіс
2. Перейти в каталог цифрових графічних творів мистецтва
3. Відфільтрувати підібраний контент по бажаним критеріям (рік, назва роботи, ім’я автору, сортування, випадкова робота, пошук конкретного тегу з вибраних)
4. З відфільтрованих цифрових графічних творів мистецтва вибрати випадковий
5. Навести на цифровий графічний витвір мистецтва для детальнішої інформації
6. Закінчити фільтрування шляхом вибору всіх цікавих параметрів</td><td>Користувач відфільтрував бажаний контент за своїм смаком: роком, назвою роботи, іменем автору, сортуванням, випадквою роботою, пошуком по конкретних тегах.

Користувач закінчив фільтрування, в каталог були повернуті всі цифрові графічні твори мистецтва, що підпадають під критерії користувача.</td></tr>
<tr><td colspan="3">Результат тест-кейсу: успішно, весь контент відфільтровано по бажаним критеріях</td></tr>
</table>

На рисунку 3.20 результат виконання багаторівневої фільтрації шляхом підбору бажаних критеріїв пошуку до бази веб-сервісу цифрових графічних творів мистецтва.

Рисунок 3.20 – Результат багаторівневої фільтрації

На етапі завершення системи цифрових графічних творів мистецтва помилок знайдено не було. Всі вище перераховані функції функціонують нормально. При рекурсивному анкетуванні користувачів і формуванні індивідуального вектору ознак за тегами помилок виявлено не було.

3.4 Інструкція користувача

Після розробки веб-сервісу цифрових графічних творів мистецтва була розроблена інструкція користувача. Над всіма фільтрами в каталозі веб-сервісу знаходяться статистика цифрових-графічних творів мистецтва, що зображена на рисунку 3.21, та пошук цифрових-графічних творів мистецтва за назвою, що зображено на рисунку 3.22.

Рисунок 3.21 – Пошук цифрових графічних творів мистецтва за назвою

Рисунок 3.22 – Під словом “Catalog” зображено статистику робіт на веб-сервісі

На рисунку 3.22 під полем вводу назви роботи веб-сервісу зображено чотири поля вводу для року, імені автору та конкретних тегів цифрових графічних творів мистецтва.

На рисунку 3.23 зображено процес сортування бази даних веб-сервісу цифрових графічних творів мистецтва. Сортування здійснюється по роках, назві роботи, імені автору зверху вниз і знизу вверх. Є сім типів сортування одне з яких повертає назад невідсортовану базу даних цифрових графічних творів мистецтва. Сортування знаходиться під полями вводу.

Рисунок 3.22 – Фільтрування за всіма полями вводу

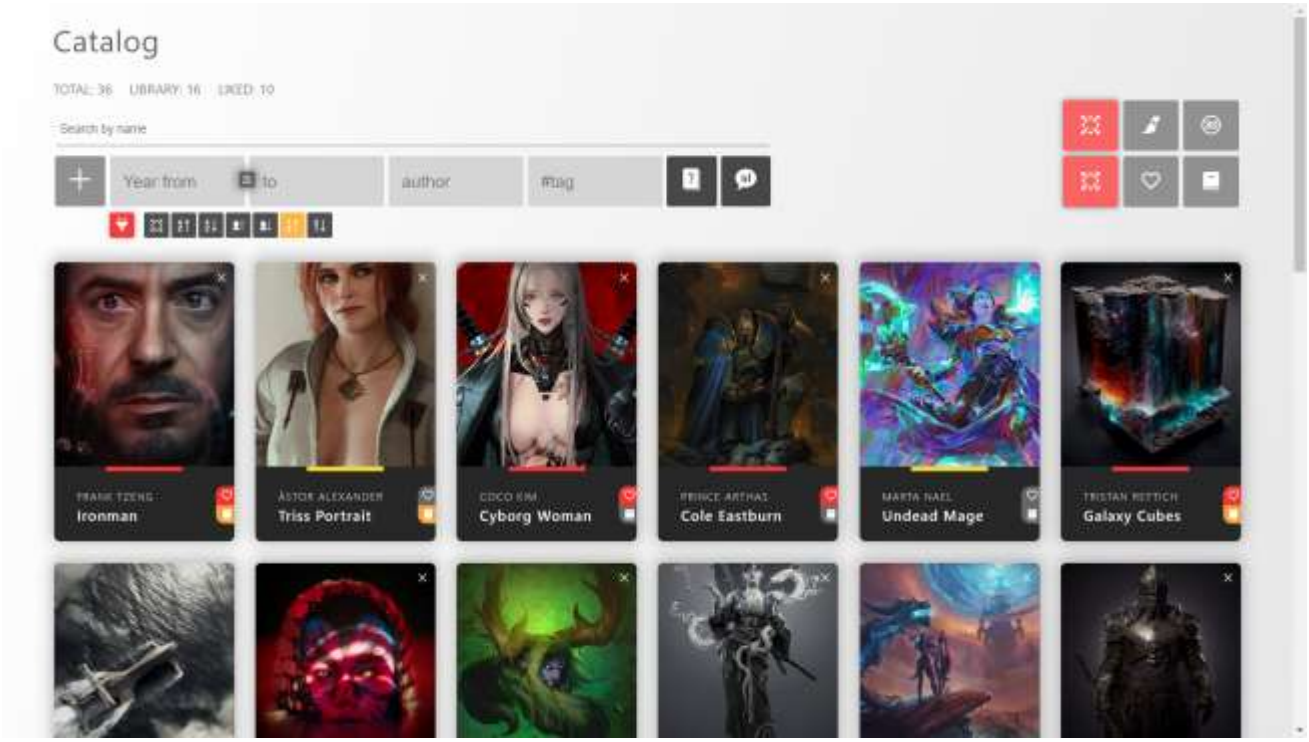


Рисунок 3.23 – Сортування цифрових графічних творів мистецтва

Біля полів вводу справа знаходиться кнопка для випадання випадкового цифрового графічного-твору мистецтва, що зображено на рисунку 3.24.

Рисунок 3.24 – Випадковий цифровий графічний витвір мистецтва

Рекурсивне анкетування користувача шляхом формування індивідуального вектору ознак за тегами зображено на рисунку 3.25.

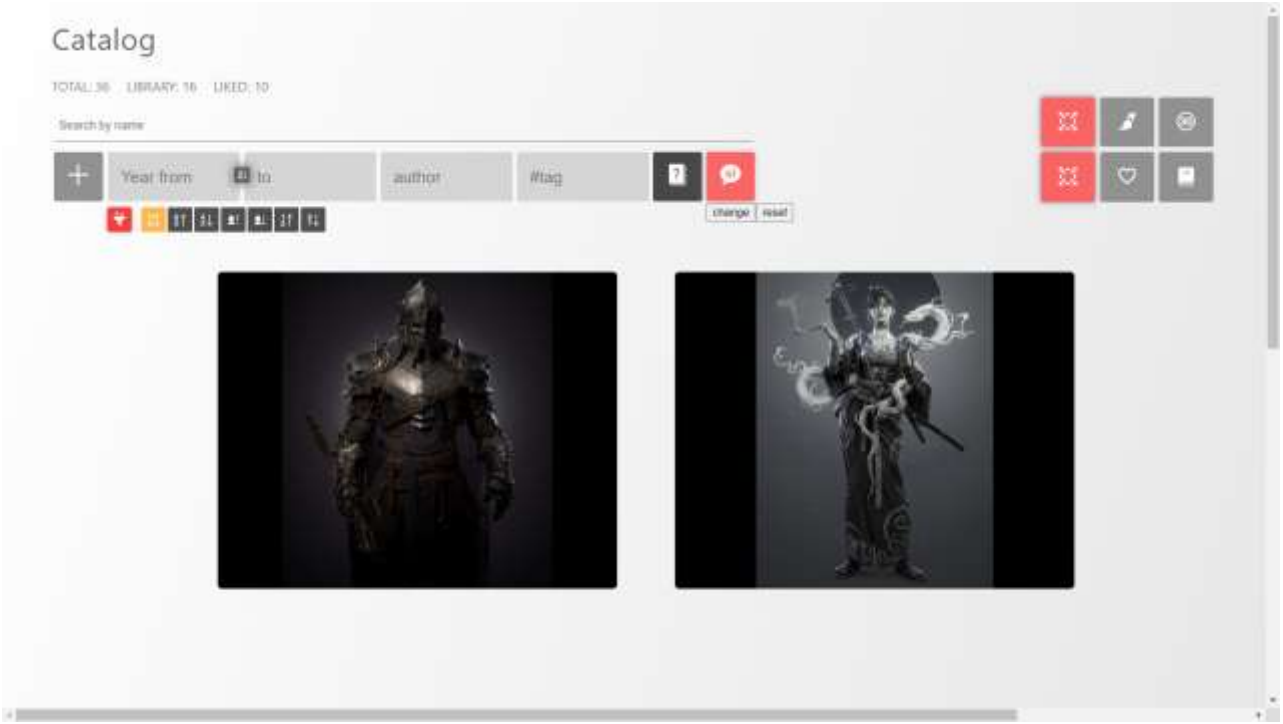


Рисунок 3.25 – Рекурсивне анкетування користувача

На рисунку 3.26 зображено результат рекурсивного анкетування користувача шляхом підбору індивідуальних ознак за тегуванням.

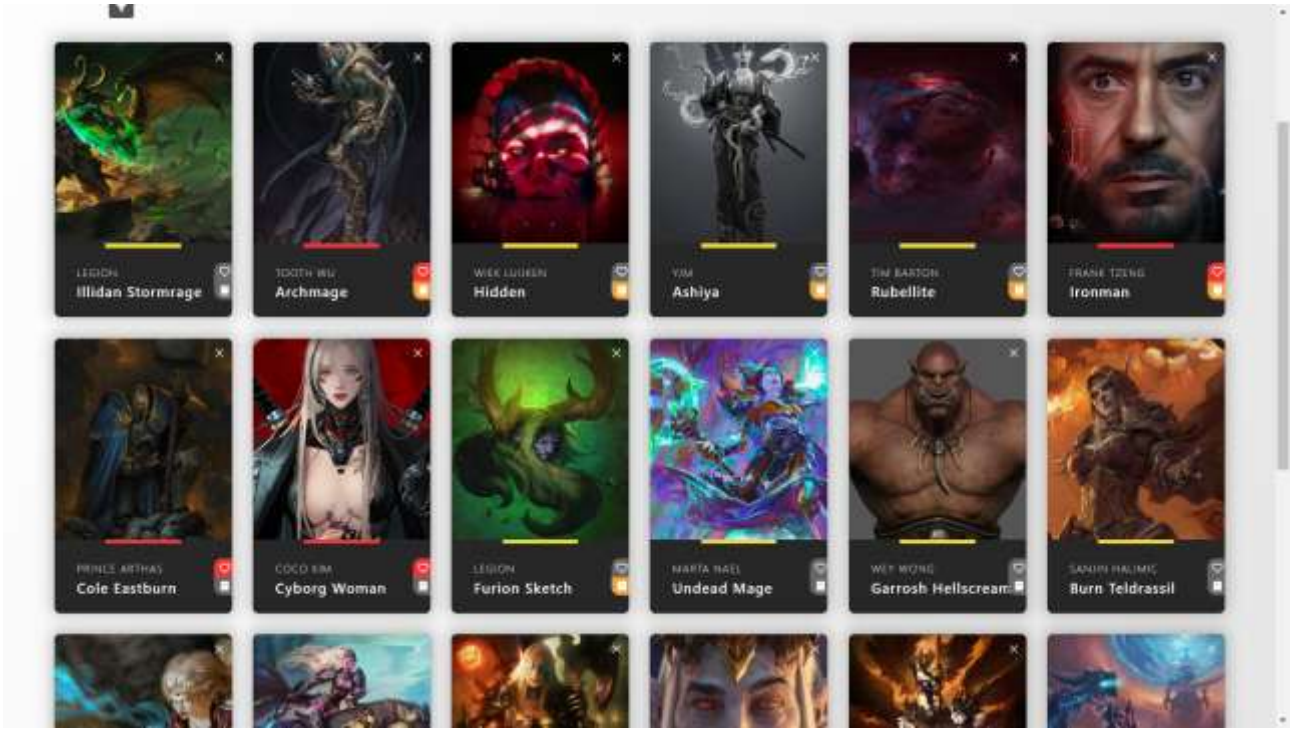


Рисунок 3.26 – Результат рекурсивного анкетування користувача

Категорії цифрових графічних творів мистецтва зображено на рисунках 3.27 та 3.28. Зверху розташовані категорії, що ділять роботи на типи 2D та 3D. Нижче знаходяться категорії якими маніпулює користувач – бібліотека вподобань та бібліотека збережених цифрових графічних творів мистецтва.

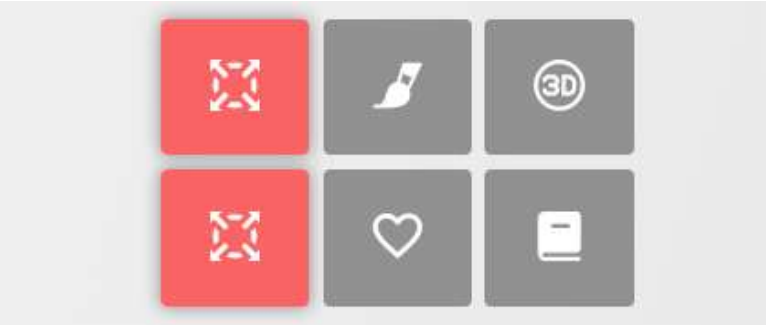


Рисунок 3.27 – Категорії. Вище 2D та 3D. Нижче вподобання та збереження до бібліотеки

На рисунку 3.29 зображено картки в яких є дві кнопки для додавання до бібліотеки або до вподобань.

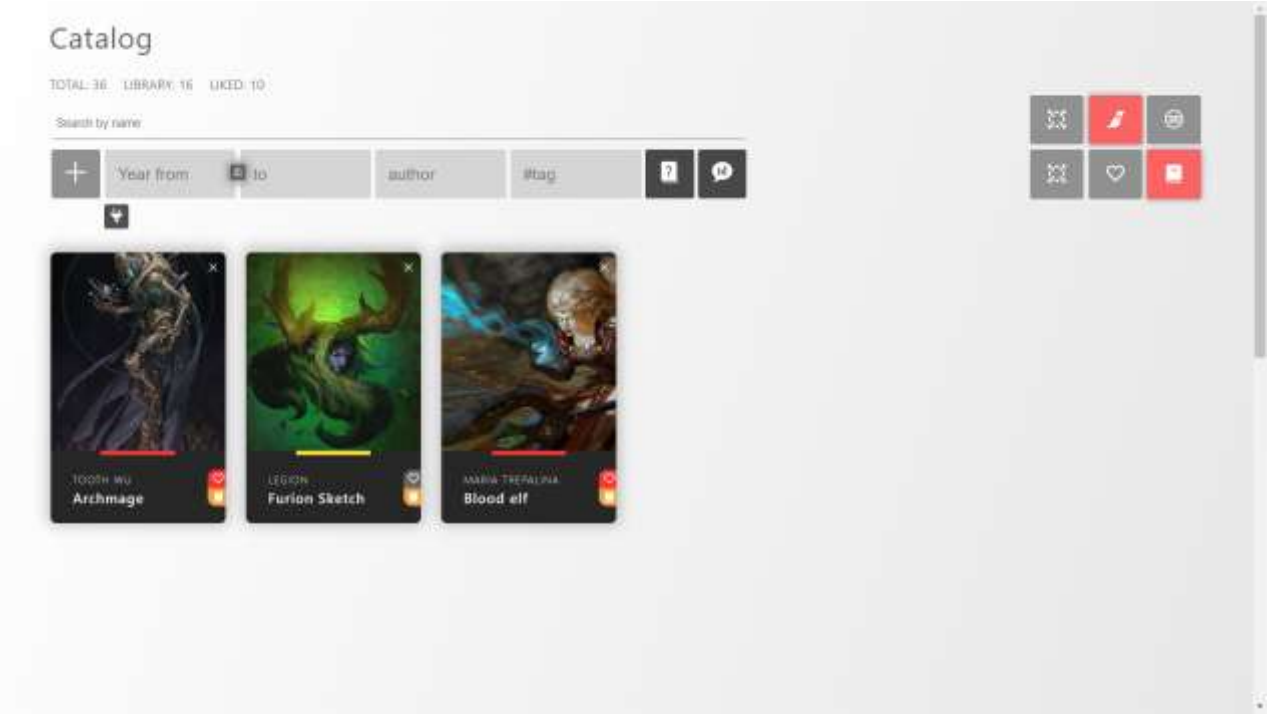


Рисунок 3.28 – Результат маніпуляції категоріями цифрових-графічних творів мистецтва

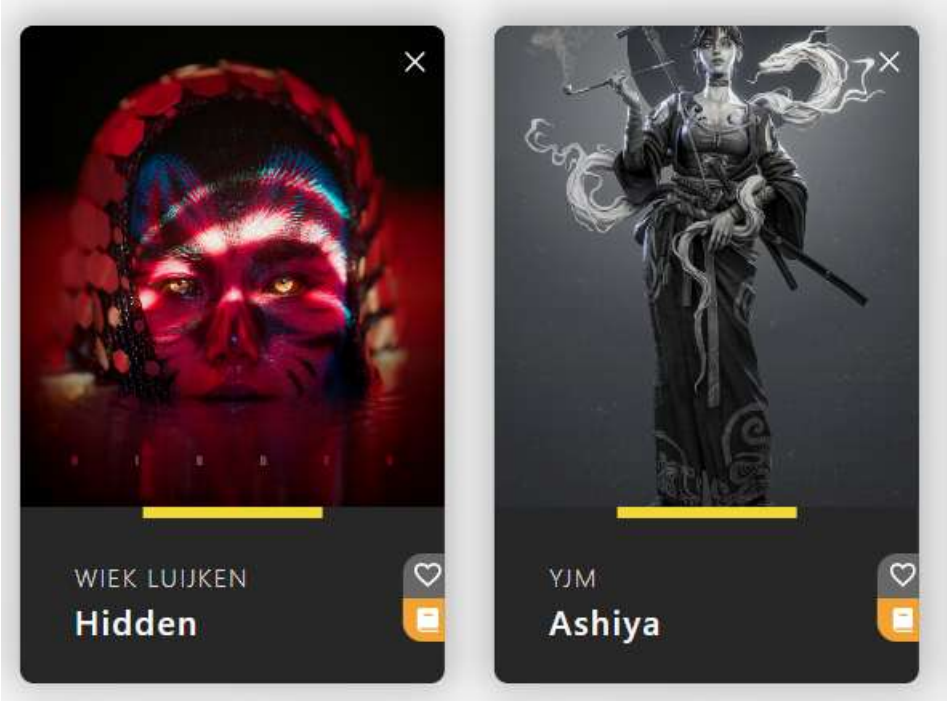


Рисунок 3.29 – Картки з кнопками додавання до вподобань або до бібліотеки

В ході тестування було продемонстровано головні функції для маніпуляції контентом у веб-сервісі цифрових графічних творів мистецтва та була надана інструкція користувачеві.

3.5 Вимоги до розгортання інформаційної системи

Для коректної роботи сайту потрібно: Firefox 36 або вище, Chrome 49.0.2623.112 або вище, Safari 7 або вище, Opera 36.0.2130.80 або вище, YaBrowser 16.11.1.673 або вище, Edge 41.0 або вище. Браузери для реалізації інтерактивних елементів клієнтської частини мають підтримувати JavaScript, DHTML. Для правильної роботи з модулями "Користувачі", "Кошик" та "Оформлення замовлення" необхідні включені Cookie.

Висновки

Під час розробки кваліфікаційної роботи бакалавра був розроблений веб-сервіс для створення та перегляду портфоліо цифрових графічних творів мистецтва з великою кількість матеріалів (цифрових графічних творів мистецтва), які розподіляються по багатьох різних критеріях – тегах. Було розроблено метод персоналізації підбору компонентів інтернет-контенту у вигляді веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва для підбору потрібної комбінації тегів для більш зручного пошуку графічних матеріалів для працедавців та потенційних покупців.

Для прикладного тестування методу персоналізації підбору компонентів інтернет-контенту було виконано його програмну реалізацію у вигляді веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва, що виконує наступні основні функції:

1. Робота користувачів з базою цифрових графічних творів мистецтва:

– навігація в системі (реєстрація, авторизація, кабінет);

– формування власного портфоліо цифрових графічних творів мистецтва;

– привласнення тегів роботам з власного портфоліо цифрових графічних творів мистецтва;

– перегляд портфоліо інших користувачів;

– фільтрування робіт інших користувачів за критеріями автора, тегів, року;

– проходження анкетування для формування індивідуального вектору ознак за тегами;

– фільтрування робіт інших користувачів за індивідуальним вектором ознак.

2. Забезпечення анкетування користувачів та автоматизоване формування індивідуального вектору ознак за тегами:

– рекурсивний вибір випадкових зображень в процесі тестування;

– рекурсивна корекція індивідуального вектору ознак в процесі тестування;

– ведення «чорного списку» опрацьованих цифрових графічних творів мистецтва;

– формування індивідуального вектору ознак за результатом тестування.

За досягнення мети створення й програмної реалізації методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів було вирішено наступні задачі:

– Проведений аналіз предметної області та проблеми персоналізованого підбору компонентів інтернет-контенту.

– Розроблений метод персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів.

– Виконане проєктування веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва на базі методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів.

– Здійснено вибір засобів розробки веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва.

– Розроблено програмну реалізацію методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів у вигляді веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва.

– Проведено тестування створеного веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва.

– Розроблена інструкція користувача для перегляду портфоліо цифрових графічних творів мистецтва інших користувачів.

– Складені вимоги до програмного забезпечення користувача для коректного використання інформаційної системи

Перелік посилань

1. UniverPL URL: https://univerpl.com.ua/blog/3d-grafika-aktualnist-napryami-ta-dumka-eksperta/

2. Klona URL: https://klona.ua/blog/3d-modelirovanie/trehmernaya-grafika-v-sovremennom-mire

3. Habr URL: https://habr.com/post/341050/

4. Prom.ua URL: https://support.prom.ua/hc/ru/articles/360005394777

5. YouTube URL: https://www.youtube.com/intl/ALL_ru/howyoutubeworks/product-features/search/

6. LinusBlog URL: https://linusblog.org/freelance/kak-zarabotat-na-3d-grafike.html

7. Seranking URL: https://seranking.com/blog/gayd-po-klasterizacii/

8. Simplecoding URL: https://www.simplecoding.org/oblako-tegov-dlya-sajta-klasterizaciya.html

9.3dclub.com URL: https://3dclub.com/blog/kak-zarabotat-na-3d-modelyah

10. 3dbaza URL: https://3dbaza.com/

11. turbosuid URL: https://www.turbosquid.com/

12. cgtarader URL: https://www.cgtrader.com/

13. shapeways URL: https://www.shapeways.com/

14. sketchfab URL: https://sketchfab.com/feed

15.blender3d.com URL: https://blender3d.com.ua/obzor-on-line-servisa-sketchfab/

16. Artstation URL: https://www.artstation.com/search

17. Developer.org URL: https://developer.mozilla.org/docs/Web/HTML

18.Developer.org URL: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/JavaScript_basics

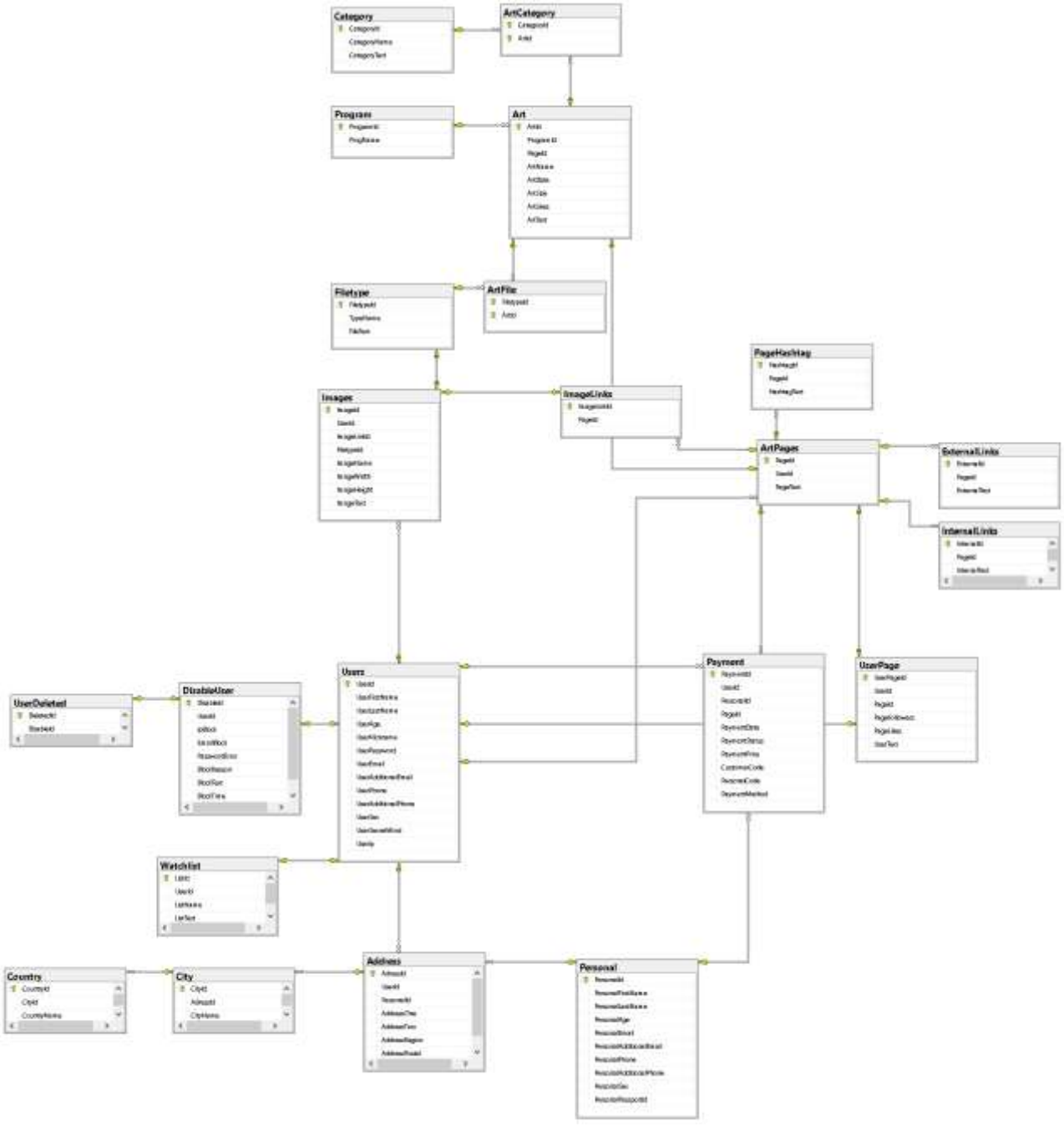
19. Reactjs.org URL: https://uk.reactjs.org/]

20. Hexlet.io URL: https://hexlet.io/blog/posts/biblioteka-react-review-article

ДОДАТКИ

Додаток А

Структура бази даних веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва



Додаток Б

Програмні коди веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва

Лістинг App.js:

```
import { Component } from 'react';

import SearchPanel from '../search-panel/search-panel';
import AppFilterCommon from '../app-filter-common/app-filter-common';
import AppFilter from '../app-filter/app-filter';
import SortFilter from '../sort-filter/sort-filter';
import AppInfo from '../app-info/app-info';
import CardList from '../card-list/card-list';
import CardAdd from '../card-add/card-add';
import ContentCategories from '../content-categories/content-categories';
import QuestFilter from '../quest-filter/quest-filter';

import sortSvg from './app-icons/filter.svg';
import plusSvg from './app-icons/plus.svg';
import questSvg from './app-icons/question.svg';
import pollSvg from './app-icons/poll.svg';

import './app.css';

class App extends Component {

    constructor(props) {
        super(props);
        this.state = {
            data: [
                {id: 0, workType: 'Art', tag: 'warcraft', year: 2008, author: 'Legion', name: 'Illidan Stormrage',like: false, increase: false, img:
'https://cdna.artstation.com/p/assets/images/images/004/329/596/large/qichao-wang-2.jpg?1482504497'},
                {id: 1, workType: 'Art', tag: 'lead of soulbinders', year: 2005, author: 'Tooth Wu', name: 'Archmage ',like: true, increase: true, img:
'https://cdnb.artstation.com/p/assets/images/images/049/249/863/large/tooth-wu-final1.jpg?1652085015'},
                {id: 2, workType: '3D', tag: 'mysterious', year: 2015, author: 'Wiek Luijken', name: 'Hidden',like: false, increase: true, img:
'https://cdna.artstation.com/p/assets/images/images/049/482/558/large/wiek-luijken-hidden.jpg?1652619899'},
                {id: 3, workType: '3D', tag: 'city', year: 2005, author: 'Ken Fairclough', name: 'Ecumenopolis ',like: true, increase: true, img:
'https://cdnb.artstation.com/p/assets/images/images/049/343/187/large/ken-fairclough-megastructure-ecumenopolis-
forweb.jpg?1652277302'},
                {id: 4, workType: '3D', tag: 'water machine', year: 2015, author: 'Till Freitag', name: '"Endeavor"',like: false, increase: true, img:
'https://cdna.artstation.com/p/assets/images/images/049/339/244/large/till-freitag-tillfreitag-endeavor-onwater-003.jpg?1652270466'},
                {id: 5, workType: '3D', tag: 'adult girl', year: 2005, author: 'Chinease', name: 'Iron girl ',like: true, increase: true, img:
'https://cdnb.artstation.com/p/assets/images/images/049/390/461/large/-gaomoxuanrang01-91.jpg?1652374669'},
                {id: 6, workType: '3D', tag: 'vip', year: 2011, author: 'YJM', name: 'Ashiya',like: false, increase: true, img:
'https://cdnb.artstation.com/p/assets/images/images/049/407/839/large/yjm-a-3-4-6.jpg?1652417367'},
                {id: 7, workType: '3D', tag: 'the elder scrolls', year: 2005, author: 'Damien Peinoit', name: 'High Isle',like: true, increase: true, img:
'https://cdnb.artstation.com/p/assets/images/images/049/351/803/large/damien-peinoit-dpeinoit-elderhighisle-02.jpg?1652289279'},
                {id: 8, workType: '3D', tag: 'space', year: 2007, author: 'Tim Barton', name: 'Rubellite ',like: false, increase: true, img:
'https://cdnb.artstation.com/p/assets/images/images/049/249/191/large/tim-barton-skybox-1-30-2022-24.jpg?1652063068'},
                {id: 9, workType: 'Art', tag: 'concept', year: 2000, author: 'kim ssang', name: 'character concept ',like: true, increase: true, img:
'https://cdna.artstation.com/p/assets/images/images/049/351/636/large/kim-ssang-06.jpg?1652289048'},
                {id: 10, workType: 'Art', tag: 'avatar', year: 2001, author: 'Evgeny Romanov', name: 'Abstract canyon',like: false, increase: true,
img: 'https://cdnb.artstation.com/p/assets/images/images/049/300/153/large/evgeny-romanov-abstract-canyon-shot-1-
06.jpg?1652180595'},
                {id: 11, workType: '3D', tag: 'marvel', year: 2022, author: 'Frank Tzeng', name: 'Ironman ',like: true, increase: true, img:
'https://cdnb.artstation.com/p/assets/images/images/000/640/115/large/frank-tzeng-render23.jpg?1443928705'},
                {id: 12, workType: '3D', tag: 'witcher', year: 2021, author: 'Ástor Alexander', name: 'Triss Portrait',like: false, increase: true, img:
'https://cdna.artstation.com/p/assets/images/images/010/092/424/large/astor-alexander-triss-2100px.jpg?1522528469'},
                {id: 13, workType: 'Art', tag: 'warcraft', year: 2018, author: 'Prince Arthas', name: 'Cole Eastburn',like: true, increase: false, img:
'https://cdnb.artstation.com/p/assets/images/images/010/064/161/large/cole-eastburn-chronicles12.jpg?1522357744'},
                {id: 14, workType: '3D', tag: 'stone', year: 2016, author: 'Tristan Rettich', name: 'Galaxy Cubes ',like: true, increase: true, img:
'https://cdnb.artstation.com/p/assets/images/images/049/364/319/large/tristan-rettich-post3-003.jpg?1652312882'},
                {id: 15, workType: '3D', tag: 'nature', year: 2029, author: 'Marc Schneider', name: 'Wild spring river',like: false, increase: true,
img: 'https://cdnb.artstation.com/p/assets/images/images/049/450/707/large/marc-schneider-wild-river-1920x960.jpg?1652524244'},
                {id: 16, workType: 'Art', tag: 'warcraft', year: 2020, author: 'coco kim', name: 'Cyborg Woman',like: true, increase: false, img:
'https://cdna.artstation.com/p/assets/images/images/049/230/234/large/coco-kim-3.jpg?1652016197'},
                {id: 17, workType: 'Art', tag: 'warcraft', year: 2012, author: 'Legion', name: 'Furion Sketch',like: false, increase: true, img:
'https://cdnb.artstation.com/p/assets/images/images/010/123/289/large/cole-eastburn-furionnew2.jpg?1522704439'},
                {id: 18, workType: 'Art', tag: 'warcraft', year: 2017, author: 'Marta Nael', name: 'Undead Mage',like: false, increase: false, img:
'https://cdnb.artstation.com/p/assets/images/images/008/019/925/large/marta-nael-world-of-warcraft-undex-mage-by-martanael-
daelgki.jpg?1509974839'},
```

```
            {id: 19, workType: '3D', tag: 'star wars', year: 2006, author: "Adam O'Donnell", name: 'Queen Amidala',like: false, increase: false,
img: 'https://cdnb.artstation.com/p/assets/images/images/040/033/339/large/adam-o-donnell-thumbnail-mainlight.jpg?1627649652'},
            {id: 20, workType: '3D', tag: 'warcraft', year: 2008, author: 'Wey Wong', name: 'Garrosh Hellscream',like: false, increase: false,
img: 'https://cdnb.artstation.com/p/assets/images/images/035/215/813/large/wey-wong-garr-final-pass-surfacing-v0048-
0002.jpg?1614381815'},
            {id: 21, workType: 'Art', tag: 'warcraft', year: 1998, author: 'Sanjin Halimic', name: 'Burn Teldrassil',like: false, increase: false,
img: 'https://cdna.artstation.com/p/assets/images/images/030/355/858/large/sanjin-halimic-sanjinhalimic-sylvanas-
closeup.jpg?1600351186'},
            {id: 22, workType: 'Art', tag: 'warcraft', year: 1999, author: 'Maria Trepalina', name: 'Blood elf',like: true, increase: true, img:
'https://cdna.artstation.com/p/assets/images/images/000/989/532/large/maria-trepalina-bloodelf-mariatrepalina-1600.jpg?1443927959'},
            {id: 23, workType: 'Art', tag: 'warcraft', year: 1977, author: 'Maria Trepalina', name: 'Death Knight',like: false, increase: false, img:
'https://cdnb.artstation.com/p/assets/images/images/000/517/899/large/maria-trepalina-night-elf-dark-knight-by-ketka.jpg?1425393577'},
            {id: 24, workType: '3D', tag: 'starcraft', year: 1998, author: 'Wey Wong', name: 'Nova', like: false, increase: false, img:
'https://cdna.artstation.com/p/assets/images/images/035/015/944/large/wey-wong-nova08-copy.jpg?1613885253'},
            {id: 25, workType: 'Art', tag: 'warcraft', year: 2003, author: 'Dmitry Prozorov', name: 'Cissenitaria',like: false, increase: false, img:
'https://cdna.artstation.com/p/assets/images/images/000/455/530/large/dmitry-prozorov-.jpg?1423219464'},
            {id: 26, workType: '3D', tag: 'warcraft', year: 2002, author: 'Wey Wong', name: 'Denathrius',like: false, increase: false, img:
'https://cdna.artstation.com/p/assets/images/images/034/958/400/large/wey-wong-den01-copy.jpg?1613697359'},
            {id: 27, workType: 'Art', tag: 'warcraft', year: 2004, author: 'Dmitry Prozorov', name: "Awakening",like: false, increase: false, img:
'https://cdna.artstation.com/p/assets/images/images/013/516/594/large/dmitry-prozorov-awakening.jpg?1539952601'},
            {id: 28, workType: 'Art', tag: 'warcraft', year: 2010, author: 'Manurgo Falls', name: 'The Maw',like: false, increase: false, img:
'https://cdna.artstation.com/p/assets/images/images/034/158/460/large/manurgo-falls-fiernes-oc-fin.jpg?1611571731'},
            {id: 29, workType: '3D', tag: 'knight', year: 2009, author: 'Ludovic Plouffe', name: 'Death Knight',like: false, increase: false, img:
'https://cdna.artstation.com/p/assets/images/images/036/899/668/large/ludovic-plouffe-02.jpg?1618934600'},
            {id: 30, workType: '3D', tag: 'warcraft', year: 1996, author: 'Jaime Jasso', name: 'Stormwind',like: false, increase: false, img:
'https://cdna.artstation.com/p/assets/images/images/011/614/320/large/jaime-jasso-sg0030gen-concept-1001.jpg?1530498460'},
            {id: 31, workType: 'Art', tag: 'mysterious', year: 1995, author: 'Dominik Mayer', name: 'Ghost Fire Knight',like: false, increase:
false, img: 'https://cdna.artstation.com/p/assets/images/images/049/015/224/large/dominik-mayer-ghostfireknight-displatepremium-
artstation.jpg?1651503897'},
            {id: 32, workType: '3D', tag: 'robot', year: 1997, author: 'Alex Senechal', name: 'Limbo Division 209',like: false, increase: false,
img: 'https://cdnb.artstation.com/p/assets/images/images/046/497/385/large/alex-senechal-paper-2.jpg?1645257262'},
            {id: 33, workType: '3D', tag: 'warcraft', year: 2003, author: 'Tim Razumovsky', name: 'Aurora Noir',like: false, increase: false, img:
'https://cdna.artstation.com/p/assets/images/images/048/677/972/large/tim-razumovsky-an-highrollers-as.jpg?1650638806'},
            {id: 34, workType: 'Art', tag: 'warcraft', year: 2002, author: 'Milivoj Ćeran', name: 'The Vainglorious',like: false, increase: false,
img: 'https://cdnb.artstation.com/p/assets/images/images/028/873/317/large/milivoj-ceran-mceran-wow-the-
vainglorious.jpg?1595787244'},
            {id: 35, workType: 'Art', tag: 'warcraft', year: 1995, author: 'Mirco Cabbia', name: 'Alexstrasza',like: false, increase: false, img:
'https://cdnb.artstation.com/p/assets/images/images/012/982/297/large/mirco-cabbia-sciamano240-alex-finale-1-3.jpg?1537469012'}
        ],
        term: '',
        menuFilter: 'all',
        categories: 'all',
        date: 0,
        dateMax: 2023,
        termAuthor: '',
        termTag: '',
        sortFilter: 'all',
        modalActive: false  ,
        AddActive: false,
        questActive: false,
        questTags: '',
        questDat: [],
        questImg: ''
    }

    this.maxId = 36;
    this.tagId = 0;
  }

  addItem = (author, name, img, workType, tag, year) => {
    const newItem = {
      workType,
      tag,
      year,
      author,
      name,
      img,
      increase: false,
      like: false,
      id: this.maxId++
    }
    this.setState(({data}) => {
      const newArr = [...data, newItem];
      return {
        data: newArr
      }
    });
  }

  deleteItem = (id) => {
```

```
    this.setState(({data}) => {
        return {
            data: data.filter(item => item.id !== id)
        }
    });
}

onToggleIncrease = (id) => {

    this.setState(({data}) => ({
        data: data.map(item => {
            if(item.id === id) {
                return {...item, increase: !item.increase}
            }

            return item;
        })
    }))
}

onToggleLike = (id) => {
    this.setState(({data}) => ({
        data: data.map(item => {
            if(item.id === id) {
                return {...item, like: !item.like}
            }

            return item;
        })
    }))
}

searchArt = (items, term, date, dateMax, termAuthor, termTag) => {
    if(term.lenght === 0) {
        return items;
    }

    return items.filter(item => {

        if(item.year >= date && item.year <= dateMax) {


            if(termAuthor != null && termTag != null) {
                return item.author.indexOf(termAuthor) > -1 && item.name.indexOf(term) > -1 && item.tag.indexOf(termTag) > -1;
            } else if(termAuthor != null) {
                return item.author.indexOf(termAuthor) > -1 && item.name.indexOf(term) > -1;
            } else if(termTag != null) {
                return item.author.indexOf(termTag) > -1 && item.name.indexOf(term) > -1;
            }
            else {
                return item.name.indexOf(term) > -1;
            }
        } else if(item.year <= dateMax && (date == 0 || date === null)) {
            if(termAuthor != null && termTag != null) {
                return item.author.indexOf(termAuthor) > -1 && item.name.indexOf(term) > -1 && item.tag.indexOf(termTag) > -1;
            } else if(termAuthor != null) {
                return item.author.indexOf(termAuthor) > -1 && item.name.indexOf(term) > -1;
            } else if(termTag != null) {
                return item.author.indexOf(termTag) > -1 && item.name.indexOf(term) > -1;
            }
            else {
                return item.name.indexOf(term) > -1;
            }
        } else if(item.year >= date && (dateMax == 0 || dateMax === null)) {
                if(termAuthor != null && termTag != null) {
                    return item.author.indexOf(termAuthor) > -1 && item.name.indexOf(term) > -1 && item.tag.indexOf(termTag) > -1;
                } else if(termAuthor != null) {
                    return item.author.indexOf(termAuthor) > -1 && item.name.indexOf(term) > -1;
                } else if(termTag != null) {
                    return item.author.indexOf(termTag) > -1 && item.name.indexOf(term) > -1;
                }
                else {
                    return item.name.indexOf(term) > -1;
                }
        } else if (date == null && dateMax == null) {
            if(termAuthor != null && termTag != null) {
                return item.author.indexOf(termAuthor) > -1 && item.name.indexOf(term) > -1 && item.tag.indexOf(termTag) > -1;
            } else if(termAuthor != null) {
                return item.author.indexOf(termAuthor) > -1 && item.name.indexOf(term) > -1;
            } else if(termTag != null) {
```

```
                return item.author.indexOf(termTag) > -1 && item.name.indexOf(term) > -1;
            }
            else {
                return item.name.indexOf(term) > -1;
            }
        }
    });
}

onUpdateSearch = (term) => {
    this.setState({term});
}

onUpdateTag = (termTag) => {
    this.setState({termTag});
}

searchAuthor = (items, term, termAuthor) => {
    if(termAuthor.lenght === 0) {
        return items;
    }

    return items.filter(item => {

        if(termAuthor.lenght >= 1) {
            return item.name.indexOf(term) > -1;
        }
    });
}

onUpdateAuthor = (termAuthor) => {
    this.setState({termAuthor});
}

searchMinYear = (items, date) => {
    if(date === 0) {
        return items;
    }

    return items.filter(item => {

        if(item.year >= date) {
            return item.year;
        }
    });
}

onMinYearUpdate = (date) => {
    this.setState({date});
}

searchMaxYear = (items, dateMax) => {
    if(dateMax === 0) {
        return items;
    }

    return items.filter(item => {

        if(item.year <= dateMax) {
            return item.year;
        }
    });
}

onMaxYearUpdate = (dateMax) => {
    this.setState({dateMax});
}

filterShow = (items => {
    return items;
})

addItemQuest = (author, name, img) => {
    const newItem = {
        author,
        name,
        img,
        increase: false,
        like: false,
        id: this.maxId++
```

```
    }
    this.setState(({data}) => {
        const newArr = [...data, newItem];
        return {
            data: newArr
        }
    });
}

filterQuest = (items, questTags, questDat) => {


        if(questTags.length > 1) {
            if (Array.isArray(questDat)) {
                questDat.push(questTags);
            }
        }

        let count = 0;
        for(let i in questDat) {
            count++;
        }

        if(count == 0) {
            return items;
        }

        if(count == 0) {
            return items;
        }

        if(questDat[0] == 'all') {
            return items;
        }

        if(count > 0) {

            return items.filter(item => {
                for(let j in questDat) {
                    if(item.tag == questDat[j]) {
                        return item.tag;
                    }
                }
            })
        } else {
            return items;
        }
}

onFilterQuest = (questTags) => {
    this.setState({questTags});
}

onFilterDat = (questDat) => {

    this.setState({questDat});
}

onFilterDatZero = (questDat) => {

    this.setState({questDat});
}

onFilterQuestImg = (questTags, questImg) => {

    this.setState({questTags});
    this.setState({questImg});
}

onFilterQuestData = (tag) => {
    const newItem = {
        tag,
        id: this.tagId++
    }
    this.setState(({questTags}) => {
        const newArr = [...questTags, newItem];
        return {
            questTags: newArr
        }
    });
```

```
}



filterPost = (items, menuFilter) => {

    switch (menuFilter) {
        case 'likes':
            return items.filter(item=> item.like);
        case 'myLib':
            return items.filter(item => item.increase);
        default:
            return items
    }
}

onFilterSelect = (menuFilter) => {
    this.setState({menuFilter});
}

filterCategories = (items, categories) => {

    switch (categories) {
        case '3D':
            return items.filter(item=> {
                if(item.workType == 'Art') {
                    return item.workType;
                }
            });
        case 'Art':
            return items.filter(item=> {
                if(item.workType == '3D') {
                    return item.workType;
                }
            });
        default:
            return items
    }
}

onCategorySelect = (categories) => {
    this.setState({categories});
}

filterSort = (items, sortFilter) => {

    switch(sortFilter) {
        case 'yearUp':
            return items.sort((a, b) => a.year < b.year ? 1 : -1);
        case 'yearDown':
            return items.sort((a, b) => a.year > b.year ? 1 : -1);
        case 'nameUp':
            return items.sort((a, b) => {
                let x = a.name.toLowerCase();
                let y = b.name.toLowerCase();
                return x < y ? -1 : x > y ? 1 : 0;
            });
        case 'nameDown':
            return items.sort((a, b) => {
                let x = a.name.toLowerCase();
                let y = b.name.toLowerCase();
                return x > y ? -1 : x < y ? 1 : 0;
            });
        case 'authorUp':
            return items.sort((a, b) => {
                let x = a.author.toLowerCase();
                let y = b.author.toLowerCase();
                return x < y ? -1 : x > y ? 1 : 0;
            });
        case 'authorDown':
            return items.sort((a, b) => {
                let x = a.author.toLowerCase();
                let y = b.author.toLowerCase();
                return x > y ? -1 : x < y ? 1 : 0;
            });
        default:
            return items
```

```
    }
  }

  onSortSelect = (sortFilter) => {
    this.setState({sortFilter});
  }

  onModalActive = (modalActive) => {
    if(modalActive == true) {
      modalActive = false;
      this.setState({modalActive});
    } else {
      modalActive = true;
      this.setState({modalActive});
    }
  }

  onAddActive = (AddActive) => {
    if(AddActive == true) {
      AddActive = false;
      this.setState({AddActive});
    } else {
      AddActive = true;
      this.setState({AddActive});
    }
  }

  onQuestActive = (questActive) => {
    if(questActive == true) {
      questActive = false;
      this.setState({questActive});
    } else {
      questActive = true;
      this.setState({questActive});
    }
  }

 render() {

  const {data, term, menuFilter, date, dateMax, termAuthor, sortFilter, modalActive, AddActive, questActive, termTag, categories,
questTags, questImg, questDat} = this.state;

  const arts = this.state.data.length;
  const libCount = this.state.data.filter(item => item.increase).length;
  const likeCount = this.state.data.filter(item => item.like).length;



  const visibleData = this.filterQuest(this.filterCategories(this.filterPost(this.filterSort(this.searchArt(data, term, date, dateMax, termAuthor,
termTag), sortFilter), menuFilter), categories), questTags, questDat);

  const visibleMaxYear = this.filterQuest(data, questTags);
  const visible = this.filterShow(data);

  return (
    <div className="app">
      <div className="menu-pos">
        <SortFilter onMinYearUpdate={this.onMinYearUpdate} onMaxYearUpdate={this.onMaxYearUpdate}
onUpdateAuthor={this.onUpdateAuthor} onUpdateTag={this.onUpdateTag}/>
        <SearchPanel onUpdateSearch = {this.onUpdateSearch}/>

        <div className="AppFilter-show">
          <button onClick={() => this.onModalActive(modalActive)} className={modalActive ? 'filter__button filter__main filter__main-
active' : 'filter__button filter__main'}><img src={sortSvg} alt="" /></button>
          <div className={modalActive ? 'AppFilter-close' : 'AppFilter-close hide'}>
            <AppFilter sortFilter={sortFilter} onSortSelect={this.onSortSelect}/>
          </div>
        </div>

        <ContentCategories categories={categories} onCategorySelect={this.onCategorySelect}/>
        <AppFilterCommon menuFilter={menuFilter} onFilterSelect={this.onFilterSelect}/>
      </div>
      <AppInfo arts={arts} libCount={libCount} likeCount={likeCount}/>

      <div className="CardAdd-show">
          <button onClick={() => this.onAddActive(AddActive)} className={AddActive ? 'add__button add__main add__main-active' :
'add__button add__main'}><img src={plusSvg} alt="" /></button>
          <div className={AddActive ? 'AppAdd-close' : 'AppAdd-close hide'}>
            <CardAdd onAdd={this.addItem}/>
          </div>
```

```
            </div>

            <div className="CardOne-show">
                <button className='one__button one__main'><img src={questSvg} alt="" /></button>
            </div>

            <div className="Questions-show">
                <button onClick={() => this.onQuestActive(questActive)} className={questActive ? 'questions__button questions__main
questions__main-active' : 'questions__button questions__main'}><img src={pollSvg} alt="" /></button>

                <div className={questActive ? '' : 'hide'}>
                    <button onClick={() => this.onFilterDat(questDat.splice(0))} questDat={questDat}
onFilterDat={this.onFilterDat}>change</button>
                    <button onClick={() => this.onFilterDatZero(this.questDat = ['all'])} questDat={questDat}
onFilterDatZero={this.onFilterDatZero}>reset</button>
                    <QuestFilter data={visible} questTags={questTags} questImg={questImg} questDat={questDat} onFilterDat={this.onFilterDat}
onFilterQuest={this.onFilterQuest} onFilterQuestImg={this.onFilterQuestImg}/>
                </div>
            </div>

            <div className={questActive ? 'hide' : 'show'}>
                <CardList
                data={visibleData}
                onDelete={this.deleteItem}
                onToggleIncrease={this.onToggleIncrease}
                onToggleLike={this.onToggleLike}
                />
            </div>
        </div>
    );
    }
}
export default App;
```

Лістинг AppFilter.js:

```
import './app-filter.css';
import filterSvg from './app-filter-icons/filter.svg';
import allSvg from './app-filter-icons/all.svg';

import sortSvg from './app-filter-icons/sort.svg';
import numUpSvg from './app-filter-icons/numUp.svg';
import numDownSvg from './app-filter-icons/numDown.svg';
import wordUpSvg from './app-filter-icons/wordUp.svg';
import wordDownSvg from './app-filter-icons/wordDown.svg';
import authorUpSvg from './app-filter-icons/authorUp.svg';
import authorDownSvg from './app-filter-icons/authorDown.svg';


const AppFilter = (props) => {

    const sortData = [
        {name: 'all', label: allSvg},
        {name: 'nameUp', label: wordUpSvg},
        {name: 'nameDown', label: wordDownSvg},
        {name: 'authorUp', label: authorUpSvg},
        {name: 'authorDown', label: authorDownSvg},
        {name: 'yearUp', label: numUpSvg},
        {name: 'yearDown', label: numDownSvg}
    ];

    const sortButtons = sortData.map(({name, label}) => {

        const active = props.sortFilter === name;
        const clazz = active ? 'app-filter__button-active' : 'app-filter__button-light';

        return (
            <button key={name} onClick={() =>props.onSortSelect(name)} type="button" className={`app-filter__button ${clazz}`}><img
src={label} alt="" /></button>
        )
    })

    return (
        <div className="app-filter">
```

```
            {sortButtons}

            {/* <button className="app-filter__button app-filter__main"><img src={filterSvg} alt="" /></button>

            <button className="app-filter__button app-filter__sort"><img src={sortSvg} alt="" /></button> */}


            {/* <button className="app-filter__button app-filter__add"><img src={addSvg} alt="" /></button> */}

        </div>
    );

}

export default AppFilter;
```

Лістинг AppFilterCommon.js:

```
import './app-filter-common.css';
import allSvg from './app-filter-common-icons/all.svg';
import favouriteSvg from './app-filter-common-icons/favourite.svg';
import librarySvg from './app-filter-common-icons/library.svg';

const AppFilterCommon = (props) => {

    const buttonsData = [
        {name: 'all', label: allSvg},
        {name: 'likes', label: favouriteSvg},
        {name: 'myLib', label: librarySvg}
    ];

    const buttons = buttonsData.map(({name, label}) => {
        const active = props.menuFilter === name;

        const clazz = active ? 'app-filter-common__btn-item-active' : 'app-filter-common__btn-item-light';

        return (
            <button
                className={`app-filter-common__btn-item ${clazz}`}
                type="button"
                key={name}
                onClick={() => props.onFilterSelect(name)}>
                <img src={label} alt="/>
                </button>
        )
    })

    return (

        <div className="app-filter-common__btn-list">
            {buttons}
        </div>
    );
}

export default AppFilterCommon;
```

Лістинг AppInfo.js:

```
import './app-info.css';

const AppInfo = ({arts, libCount, likeCount}) => {
    return (
        <div className="app-info">
            <div className="app-info__title">Catalog</div>

            <div className="app-info__catalog-count">
                <div>TOTAL: <span className="count"> {arts}</span></div>
                <div>LIBRARY: <span className="count"> {libCount}</span></div>
                <div>LIKED: <span className="count"> {likeCount}</span></div>
            </div>
    </div>
    );
}

export default AppInfo;
```

Лістинг CardAdd.js:

```
import { Component } from 'react'

import './card-add.css';

import crossSvg from './card-add-icons/cross.svg';

class CardAdd extends Component {

  constructor(props) {
    super(props);
    this.state = {
      workType: '',
      tag: '',
      year: 0,
      author: '',
      name: '',
      img: '',
      close: true
    }
  }

  onValueChange = (e) => {
    this.setState({
      [e.target.name]: e.target.value
    });
  }

  onSubmit = (e) => {
    e.preventDefault();
    if (this.state.author.length < 3 || this.state.name.length < 3 || this.state.img.length < 3 ) return;
    this.props.onAdd(this.state.author, this.state.name, this.state.img);
    this.setState({
        author: '',
        name: '',
        img: '',
        workType: '',
        tag: '',
        year: 0
    })
  }

  render() {

    const {author, name, img, workType, tag, year, close} = this.state;

    return (
      <div class={close ? 'card' : 'card hide'}>

          <form onSubmit = {this.onSubmit}>
            <div onClick={() => this.onClose(close)} class="img-cross"><img src={crossSvg} alt=""/></div>

            <div class="card__search-author-container">
                <input name='author' onChange={this.onValueChange} class="card__search-author" value={author} type="text"
placeholder="Author"/>
            </div>
            <div class="card__search-name-container">
                <input name='name' onChange={this.onValueChange} class="card__search-name" value={name} type="text"
placeholder="Name"/>
            </div>
            <div class="card__search-href-container">
                <input name='img' onChange={this.onValueChange} class="card__search-href" value={img} type="text" placeholder="Image
address"/>
            </div>
            <div class="card__search-warktype-container">
                <input name='workType' onChange={this.onValueChange} class="card__search-author" value={workType} type="text"
placeholder="Worktype"/>
            </div>
            <div class="card__search-tag-container">
                <input name='tag' onChange={this.onValueChange} class="card__search-name" value={tag} type="text" placeholder="Tag"/>
            </div>
            <div class="card__search-year-container">
                <input name='year' onChange={this.onValueChange} class="card__search-href" value={year} type="text" placeholder="Year"/>
            </div>

          <button type='submit' class="card-apply">Apply</button>
          </form>
      </div>
```

```
    )
  }
}

export default CardAdd
```

Лістинг CardItem.js:

```
// import { Component } from 'react';

import './card-item.css';

// import sylvanas from './card-item-img/anduin-sylvana.jpg';

import favouriteSvg from './card-item-icons/favourite.svg';
import librarySvg from './card-item-icons/library.svg';
import crossSvg from './card-item-icons/cross.svg';

const CardItem = (props) => {

        const {author, name, img, onDelete, onToggleIncrease, onToggleLike, increase, like} = props;

        let categoryColor = 'category-color',
            chooseIcon = 'choose-icon-like',
            chooseLibs = 'choose-icon-lib';

        if (like) {
            categoryColor += ' category-color-red';
            chooseIcon += ' choose-icon-like-active';
        }

        if(increase) {
            chooseLibs += ' choose-icon-lib-active';
        }


        return(
            <div className="main__catalod-all">

                <div onClick={onDelete} className="img-cross-catalog">
                    <img src={crossSvg} alt=""/>
                </div>

                    <div className="main__catalog-card">
                        <img src={img} alt=""/>
                        <div className="main__catalog-card-descr">
                            <div className="author">{author}</div>
                            <div className="title">{name}</div>
                        </div>
                    </div>

                    <div className="choose-category">
                        <img onClick={onToggleLike} src={favouriteSvg} alt="" className={chooseIcon}/>
                        <img onClick={onToggleIncrease} src={librarySvg} alt="" className={chooseLibs}/>
                    </div>

                    <div className={categoryColor}></div>
            </div>
        );
    }

export default CardItem;
```

Лістинг CardItemQuest.js:

```
import './card-item-quest.css';

const CardItemQuest = (props) => {

        const {img} = props;


        return(
            <div className="card-choose">
                <img src={img} alt=""/>
            </div>
```

```
        );
    }

export default CardItemQuest;
```

Лістинг CardList.js:

```
import './card-list.css';

import CardItem from "../card-item/card-item";

const CardList = ({data, onDelete, onToggleIncrease, onToggleLike}) => {


    const elements = data.map((item) => {
        const {id, ...itemProps} = item;

        return (
            <CardItem
                key={id}
                {...itemProps}
                onDelete = {() => onDelete(id)}
                onToggleIncrease={() => onToggleIncrease(id)}
                onToggleLike={() => onToggleLike(id)}
            />
        );
    });

    return(
        <div className="main__catalog-content">

      {elements}

        </div>
    );
}

export default CardList;
```

Лістинг ContentCategories.js:

```
import './content-categories.css';

import allSvg from './content-categories-icons/all.svg';
import brushSvg from './content-categories-icons/arts.svg';
import modelSvg from './content-categories-icons/model.svg';

const ContentCategories = (props) => {

    const categoriesData = [
        {name: 'all', label: allSvg},
        {name: '3D', label: brushSvg},
        {name: 'Art', label: modelSvg}
    ];

    const buttons = categoriesData.map(({name, label}) => {
        const active = props.categories === name;

        const clazz = active ? 'content-categories__btn-item-active' : 'content-categories__btn-item-light';

        return (
            <button
                className={`content-categories__btn-item ${clazz}`}
                type="button"
                key={name}
                onClick={() => props.onCategorySelect(name)}>
                <img src={label} alt="/>
                </button>
        )
    })

    return (

        <div className="content-categories__btn-list">
            {buttons}
        </div>
```

```
    );
}

export default ContentCategories;
```

Лістинг QuestFilter.js:

```
import './quest-filter.css';

import CardItemQuest from '../card-item-quest/card-item-quest';

const QuestFilter = (props) => {

    let count = 0;
    for (let key in props.data) {
        count++;
    }

    let x = Math.floor(Math.random() * (count - 0)) + 0;
    let y = Math.floor(Math.random() * (count - 0)) + 0;

    for(let i = 0; i < 10; i++){
        if(x == y || props.data[x].tag == props.data[y].tag) {
            x = Math.floor(Math.random() * (count - 0)) + 0;
            y = Math.floor(Math.random() * (count - 0)) + 0;
        }
    }

    let a, b;

    let img1, img2;

    props.data.filter(item => {
        if (x == item.id) {
            a = item.tag;
            img1 = item.img;
        }

        if (y == item.id) {
            b = item.tag;
            img2 = item.img;
        }
    })


    const buttonsData = [
        {name: a, label: img1},
        {name: b, label: img2}
    ];

    const buttons = buttonsData.map(({name, label}) => {
        const active = label;

        return (
            <div>
                <img className='card-choose-img' key={label} src={active} onClick={() => props.onFilterQuest(name)} alt=''/>
            </div>
            )
        })

    return(
        <div className="main__catalog-quest">

            {buttons}
            {/* {elements} */}
        </div>
    );
}

export default QuestFilter
```

Лістинг SearchPanel.js:

```
import { Component } from 'react';

import './search-panel.css';
```

```
class SearchPanel extends Component {

    constructor(props) {
        super(props);
        this.state = {
            term: ''
        }
    }

    onUpdateSearch = (e) => {
        const term = e.target.value;
        this.setState({term});
        this.props.onUpdateSearch(term);
    }

    render() {
        return (
            <div className="search-panel">
                <input type="search" id="search-input" placeholder="Search by name" value={this.state.term}
onChange={this.onUpdateSearch}/>
            </div>
        );
    }
}

export default SearchPanel;
```

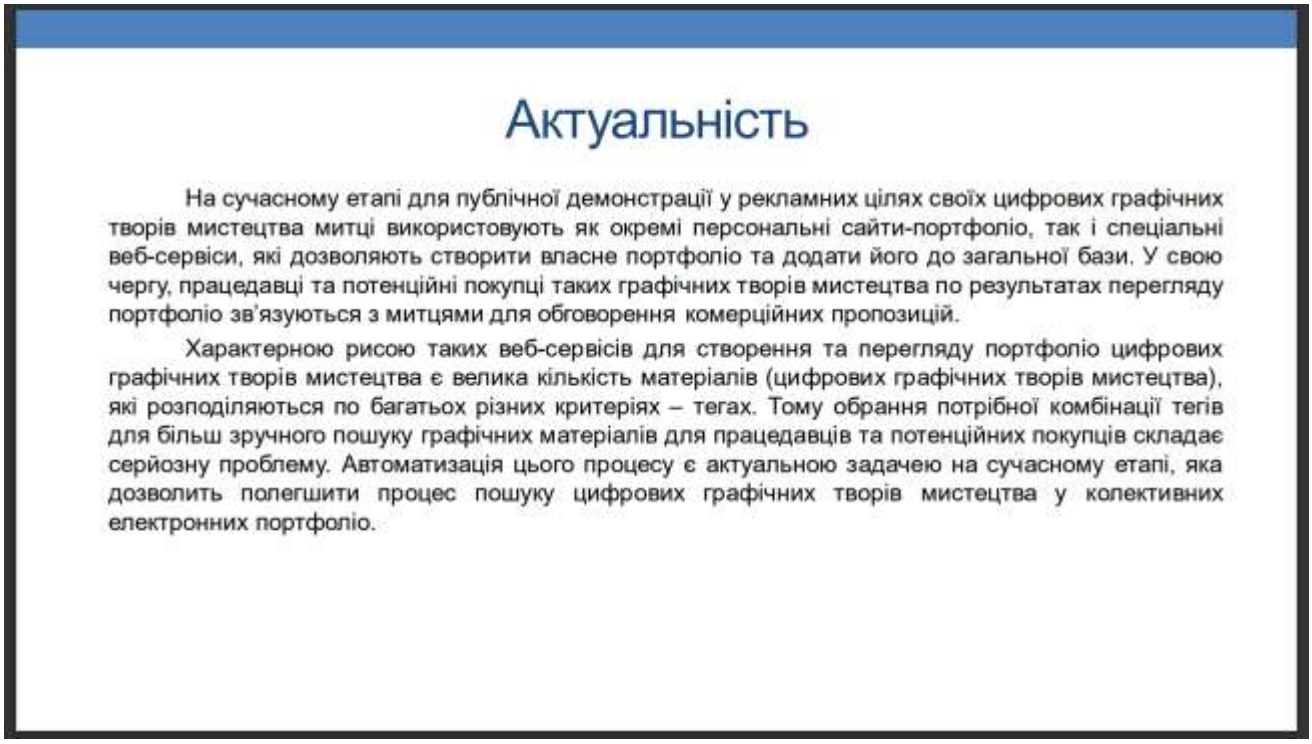
Лістинг SortFilter.js:

```
import { Component } from 'react';

import './sort-filter.css';

import countSvg from './sort-filter-icons/count.svg';

class SortFilter extends Component {

    constructor(props) {
        super(props);
        this.state = {
            date: null,
            dateMax: null,
            termAuthor: '',
            termTag: ''
        }
    }

    onMinYearUpdate = (e) => {
        const date = e.target.value;
        this.setState({date});
        this.props.onMinYearUpdate(date);
    }

    onMaxYearUpdate = (e) => {
        const dateMax = e.target.value;
        this.setState({dateMax});
        this.props.onMaxYearUpdate(dateMax);
    }

    onUpdateAuthor = (e) => {
        const termAuthor = e.target.value;
        this.setState({termAuthor});
        this.props.onUpdateAuthor(termAuthor);
    }

    onUpdateTag = (e) => {
        const termTag = e.target.value;
        this.setState({termTag});
        this.props.onUpdateTag(termTag);
    }

    render() {
        return(
            <div className="sort-filter">
                <img className="sort-filter__img" src={countSvg} alt=""/>
                <input type="text" placeholder="Year from" value={this.state.date} onChange={this.onMinYearUpdate}/>
                <input type="text" placeholder="to" value={this.state.dateMax} onChange={this.onMaxYearUpdate}/>
```

```
            <input className='input-long' type="text" placeholder="author" value={this.state.termAuthor}
onChange={this.onUpdateAuthor}/>
            <input className='input-long' type="text" placeholder="#tag" value={this.state.termTag} onChange={this.onUpdateTag}/>
          </div>
        );
    }
}

export default SortFilter;
```

Додаток В

Презентаційний матеріал



Мета і задачі роботи

Мета кваліфікаційної роботи бакалавра – розробка методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів за допомогою рекурсивного анкетування користувача для формування індивідуального вектору ознак за тегами, що створюються користувачем при публікації цифрового графічного твору мистецтва і подальшого передавання роботи в базу даних для подальшого представлення на веб-сервісі

Об'єкт дослідження – процес підбору комбінацій тегів для пошуку цифрових графічних творів мистецтва у колективних електронних портфоліо.

Предмет дослідження – інформаційні технології, моделі, методи та засоби для автоматизованого підбору комбінацій тегів для полегшення процесу пошуку цифрових графічних творів мистецтва у колективних електронних портфоліо.

Схема методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів

Структура та функціональне призначення програмних складових веб-сервісу

Схему структури та функціонального призначення програмних складових веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва наведено на рисунку 3.1. Веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва містить базовий клас веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва «App», який включає в себе ряд класів, які групуються у компоненти.

Зокрема, такими компонентами є:

– Компоненти фільтрування цифрових графічних творів мистецтва

– Компоненти представлення цифрових графічних творів мистецтва на сторінці

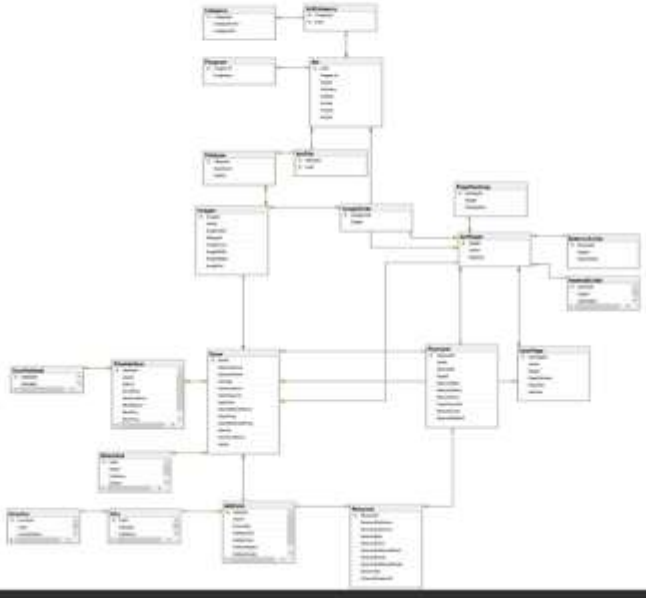
– Компоненти додавання цифрових графічних творів мистецтва в каталог

– Компоненти розбиття цифрових графічних творів мистецтва на категорії

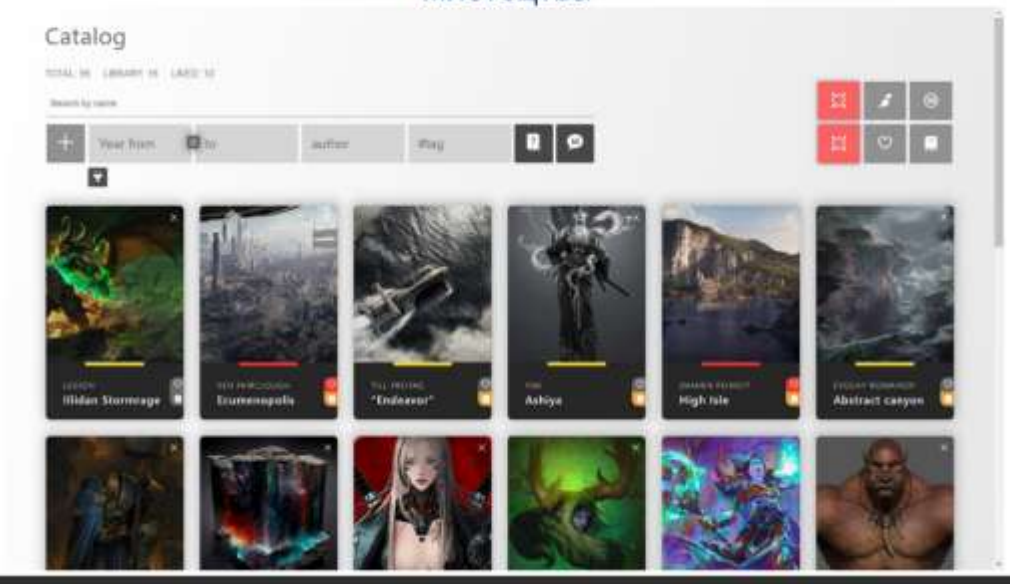
– Компоненти рекурсивного анкетування користувачів для формування індивідуального вектору ознак цифрових графічних творів мистецтва

Схема структури та функціонального призначення програмних складових веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва

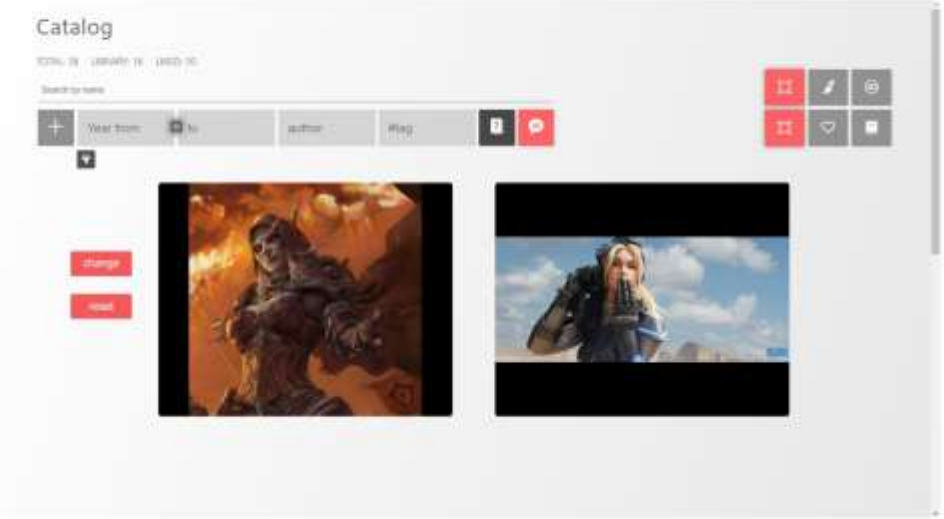
Даталогічна модель бази даних



Програмна реалізація веб-сервісу цифрових графічних творів мистецтва



Програмна реалізація методу рекурсивного анкетування користувача шляхом формування індивідуального вектору ознак за тегами



Висновки

Під час розробки кваліфікаційної роботи бакалавра був розроблений веб-сервіс для створення та перегляду портфоліо цифрових графічних творів мистецтва з великою кількість матеріалів (цифрових графічних творів мистецтва), які розподіляються по багатьох різних критеріях – тегах. Було розроблено метод персоналізації підбору компонентів інтернет-контенту у вигляді веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва для підбору потрібної комбінації тегів для більш зручного пошуку графічних матеріалів для працедавців та потенційних покупців.

Для прикладного тестування методу персоналізації підбору компонентів інтернет-контенту було виконано його програмну реалізацію у вигляді веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва, що виконує наступні основні функції:

- *1. Робота користувачів з базою цифрових графічних творів мистецтва:*
  - навігація в системі (реєстрація, авторизація, кабінет);
  - формування власного портфоліо цифрових графічних творів мистецтва;
  - привласнення тегів роботам з власного портфоліо цифрових графічних творів мистецтва;
  - перегляд портфоліо інших користувачів;
  - фільтрування робіт інших користувачів за критеріями автора, тегів, року;
  - проходження анкетування для формування індивідуального вектору ознак за тегами;
  - фільтрування робіт інших користувачів за індивідуальним вектором ознак.
- *2. Забезпечення анкетування користувачів та автоматизоване формування індивідуального вектору ознак за тегами:*
  - рекурсивний вибір випадкових зображень в процесі тестування;
  - рекурсивна корекція індивідуального вектору ознак в процесі тестування;
  - ведення «чорного списку» опрацьованих цифрових графічних творів мистецтва;
  - формування індивідуального вектору ознак за результатом тестування.

UNICHECK

Ім'я користувача:
Кафедра КН

ID перевірки:
1011582869

Дата перевірки:
15.06.2022 08:35:37 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
15.06.2022 08:38:14 EEST

ID користувача:
100005671

Назва документа: **Щіпайло_ЗАПИСКА_short**

Кількість сторінок: **73** Кількість слів: **11436** Кількість символів: **86277** Розмір файлу: **3.79 MB** ID файлу: **1011452354**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

3.46%
Схожість

Найбільша схожість: **1.64%** з джерелом з Бібліотеки (ID файлу: **1011421027**)

1.08% Джерела з Інтернету	41	Сторінка 75
2.57% Джерела з Бібліотеки	62	Сторінка 75

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	15 сторінок

Wed Jun 15 07:45:43 EEST 2022, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 2.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 13%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 105392 Название: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА на тему Метод персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів Добавлено в БД: 2022-06-15 Авторы: О.М. Щіпайло Руководители: О.В. Мазурець Консультанты: Опоненты:	Символы	Лексемы	Символы	Лексемы
	69227	1002	1872 (3%)	31 (3%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Метод персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів*

Автор: *студент групи КН-18-1 Щіпайло Олександр Михайлович*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *доцент кафедри КН Мазурець О.В.*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	*відповідає*
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі є законними і не є плагіатом, оскільки:

Підтвердження: запозичення, виявлені в роботі Щіпайло О.М., є законними і не є плагіатом, оскільки:

1) за програмою Anti-Plagiarism виявлені 3% запозичень вказують на документ автора роботи та містять його ж Звіт з практики.

2) За програмою UNICHECK виявлені 3,46%, які є фрагментарними, не більше 1,64% на оджерело – містять поширені конструкції, загальновідомі терміни та визначення.

3) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;

Керівник роботи _____ *Олександр МАЗУРЕЦЬ*

Гарант ОП _____ *Олександр МАЗУРЕЦЬ*

Завідувач кафедри КН _____ *Олександр БАРМАК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КІ-18-1 Шпайло Олександра Михайловича*
за темою: *Метод персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів*

1. Актуальність обраної теми

На сучасному етапі для публічної демонстрації у рекламних цілях своїх цифрових графічних творів мистецтва митці використовують як окремі персональні сайти-портфоліо, так і спеціальні веб-сервіси, які дозволяють створити власне портфоліо та додати його до загальної бази. У свою чергу, працедавці та потенційні покупці таких графічних творів мистецтва по результатах перегляду портфоліо зв'язуються з митцями для обговорення комерційних пропозицій. Відповідно, метод персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів є актуальною задачею комп'ютерних наук.

2. Повнота розкриття мети та завдань роботи

Метою кваліфікаційної роботи бакалавра була реалізація методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів. Для досягнення поставленої мети виконано наступні задачі: проведений аналіз предметної області та проблеми персоналізованого підбору компонентів інтернет-контенту, розроблений метод персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів, виконане проєктування веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва на базі методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів, здійснено вибір засобів розробки веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва, розроблено програмну реалізацію методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів у вигляді веб-сервісу для створення та перегляду

портфоліо цифрових графічних творів мистецтва, проведено тестування створеного веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва, розроблена інструкція користувача для перегляду портфоліо цифрових графічних творів мистецтва інших користувачів, складені вимоги до програмного забезпечення користувача для коректного використання інформаційної системи. Зміст роботи виконано в повному обсязі, належним чином. Тема є змістовно розкритою.

3. Зміст кожного розділу роботи

Перший розділ присвячений проведенню аналізу предметної області та визначенню основних параметрів для розв'язку поставленої задачі. Другий розділ присвячений проєктуванню функціональної структури інформаційної системи. Третій розділ присвячений програмній реалізації спроєктованої функціональної структури інформаційної системи.

4. Оцінка розробленої інформаційної системи, її практична цінність

Спроєктований веб-сервіс на базі методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування зразками компонентів для створення та перегляду портфоліо цифрових графічних творів мистецтва з великою кількість матеріалів (цифрових графічних творів мистецтва), які розподіляються по багатьох різних критеріях – тегах. Було розроблено метод персоналізації підбору компонентів інтернет-контенту у вигляді веб-сервісу для створення та перегляду портфоліо цифрових графічних творів мистецтва для підбору потрібної комбінації тегів для більш зручного пошуку графічних матеріалів для працедавців та потенційних покупців.

5. Якість оформлення кваліфікаційної роботи бакалавра

Робота виконана на належному науково-методичному рівні та відповідає встановленим вимогам щодо оформлення такого роду праць.

6. Недоліки кваліфікаційної роботи бакалавра

У пояснювальній записці кваліфікаційної роботи є ряд помилок в оформленні, зокрема відсутні посилання на рисунку 1.11 та 1.13. Суттєві недоліки відсутні.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

Враховуючи рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка ***«добре»***.

Рецензент Макарчишин Р.А. доцент каф. ЯКІТ

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КІ-18-1 Шпайло Олександра Михайловича*

за темою *Метод персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів*

1. Актуальність теми

Не кожен дизайнер знається у веб-розробці та рекламі, тому для публічної демонстрації у рекламних цілях своїх цифрових графічних творів мистецтва митці використовують як окремі персональні сайти-портфоліо, так і спеціальні веб-сервіси, які дозволяють створити власне портфоліо та додати його до загальної бази цифрових графічних творів мистецтва. В таких системах зазвичай відсутній будь-який автоматизований підбір тегів, візуальний або текстовий, що також обмежує фільтрацію, особливо коли користувач або замовник не знає про існування тегів. Відповідно, метод персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів є актуальною задачею комп'ютерних наук.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

Тема кваліфікаційної роботи відповідає предметній області спеціальності 122 Комп'ютерні науки та вимогам до кваліфікаційної роботи бакалавра, оскільки метою роботи є створення Метод персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів є актуальною задачею комп'ютерних наук. При вирішенні поставленої задачі використано математичні моделі, методи та алгоритми розв'язання теоретичних і прикладних задач, що виникають при розробці інформаційних технологій.

3. Професійні та особистісні якості бакалавра

Студент Шпайло Олександр Михайлович під час роботи над кваліфікаційною роботою бакалавра продемонстрував високий рівень знань та умінь за спеціальністю "Комп'ютерні науки", проявив себе відповідальним та сумлінним студентом. Опанував необхідні професійні навички за напрямком «Комп'ютерні науки».

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Одержані в роботі результати є наслідком особистої діяльності студента, який самостійно виконував всі поставлені задачі.

5. Ступінь оволодіння методами дослідження

При реалізації кваліфікаційної роботи студент Щіпайло О.М. показав достатній рівень компетентностей та володіння необхідними інструментами та обладнанням, методами, методиками та технологіями предметної області комп'ютерних наук.

6. Повнота та якість розкриття теми роботи

Усі поставлені вимоги до роботи виконані в повному обсязі, проведено аналіз актуальності та відомих досліджень в межах обраної теми, також реалізоване програмне забезпечення.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Викладення матеріалу логічне, послідовне та аргументоване. Мова і стиль викладення кваліфікаційної роботи відповідають стандартам, що забезпечує доступність сприймання матеріалу і відповідає вимогам до сучасних наукових робіт.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Реалізований програмний застосунок на базі методу персоналізованого підбору компонентів інтернет-контенту шляхом формування ієрархії класів засобами анкетування випадковими зразками компонентів може бути використаний на різного роду інтернет-платформах, що здійснюють роботу із цифровими творами мистецтва, зокрема на таких як OpenSea та інші.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

Враховуючи високий рівень виконання та забезпечення усіх необхідних вимог, робота може бути допущена до захисту. Рекомендована оцінка «відмінно».

Керівник _____ *к.т.н., доцент кафедри КН МАЗУРЕЦЬ О.В.*