КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття освітнього рівня бакалавра

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

ІНФОРМАЦІЙНІ СИСТЕМИ. РОЗРОБКА СПЕЦІАЛІЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ХУДОЖНИКІВ

Виконав студент 4-го курсу
Олександр ПОДВАЖУК _____
(підпис)

Науковий керівник:
доцент, кандидат фіз.-мат. наук
Євгеній ІВАНОВ _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Роботу розглянуто та допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«___» _____ 2021 р.,
протокол № __
Завідувач кафедри
Олександр ПРОВОТАР _____
(підпис)

Київ – 2021

РЕФЕРАТ

Обсяг роботи 79 сторінок, 16 ілюстрацій, 17 джерел посилання, 19 додатків.

JOINART, АРХІТЕКТУРА, ВИМОГА ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ, ЗАСІБ РОЗРОБКИ, ІНФОРМАЦІЙНА СИСТЕМА, ІНТЕРФЕЙС ПРОГРАМНОГО ПРОДУКТУ, КЛІЄНТ-СЕРВЕР, МОДЕЛЬ РОЗРОБКИ, ТЕСТУВАННЯ, ХУДОЖНИК.

Об’єктом роботи є процес створення спеціалізованої інформаційної системи для художників. Предметом роботи є програмний додаток інформаційної системи для художників «JoinArt».

Метою роботи є створення спеціалізованої інформаційної системи для художників, яка надає статистичну інформацію про популярність картин та жанрів, а також допомагає художникам у продажі картин та зборі відгуків про них.

Методи розроблення: комп’ютерне моделювання, розробка програмного продукту на основі каскадної моделі. Інструменти розроблення: інтегровані середовища програмування Mirosoft Visual Studio Community 2019 16.9.4 та JetBrains WebStorm 2021.1.1, фремворк ASP.NET Core та JavaScript-фреймворк React, мови програмування C#, JavaScript, мови розмітки HTML5, CSS3.

Результати роботи: виконано загальний огляд інформаційних систем та методів їх розробки, розроблено спеціалізовану інформаційну систему для художників «JoinArt», яка полегшує процес створення картин, надаючи художникам статистичну інформацію про популярність картин та жанрів, а також допомагає художникам продавати картини та збирати про них відгуки.

Інформаційна система «JoinArt» може застосовуватися для комерційного використання певною організацією, яка б надавала послуги для полегшення процесу створення та реалізації картин художників.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта розробки. Інформаційні системи відіграють надзвичайно важливу роль у сьогоднішньому світі, в деяких сферах вони навіть стали незамінними. Економічна, освітня, фінансова, урядова – це лише мала частина переліку сфер, діяльність яких неможливо уявити без використання інформаційних систем. Більш того, сучасні корпорації активно впроваджують їх у свою діяльність, а деякі навіть повністю побудовані на базі інформаційних систем. Промисловість не могла б здійснювати закупівлі, медичні центри не могли б якісно обслуговувати пацієнтів, банки не могли б обробляти транзакції, держава не могла б налагодити роботу податкової служби без інформаційних систем. Але крім таких важливих для функціонування суспільства сфер, інформаційні системи також знайшли застосування у не менш важливій для кожної людини сфері – соціальній.

З кожним роком комп’ютерна автоматизація все більше проникає у кожний вид повсякденної діяльності людини. Без використання технологій не обходяться такі звичні для людей речі, як спілкування, щоденна праця, відпочинок, захоплення, пошук нової цікавої інформації та прийняття рішень. Тому проблеми розробки ефективних та надійних систем для обробки інформації уже тривалий час непокоять спеціалістів в області інформаційних технологій, які намагаються створити з одного боку ефективні, надійні та захищені засоби інтеграції функціональних та структурних компонентів, а з іншого боку – прості та інтуїтивно зрозумілі способи керування кожним із них окремо.

Актуальність роботи та підстави для її виконання. Досягти потрібної інтеграції складових приватного життя кожної людини вдалося завдяки соціальним мережам, які надали людям можливість реалізувати свої потреби у соціалізації та самореалізації шляхом використання комп’ютерних технологій. Соціальні мережі – це такі інформаційні системи, що використовуються для створення нових соціальних зв’язків між людьми, спілкування, пошуку однодумців за інтересами, а також для трудової діяльності та розваг. Сучасна

соціальна мережа надає своєму користувачеві широкий спектр можливостей, без яких нині неможливо уявити життя. На даний момент існує безліч різних платформ, як загальних, так і спеціалізованих для конкретного виду діяльності та інтересів – інформаційні системи для студентів, геймерів, програмістів, дизайнерів тощо. Проте спеціалізованих інформаційних систем, які б стосувалися діяльності художників, незрівнянно мало, тому розробка інформаційної системи, де художники могли б оцінювати один одного, а звичайні люди могли б купувати їхні картини, є надзвичайно актуальною задачею.

Мета й завдання роботи. Метою кваліфікаційної роботи є створення спеціалізованої інформаційної системи для художників, яка надає статистичну інформацію про популярність картин та жанрів, а також допомагає художникам у продажі картин та зборі відгуків про них. Для досягнення цієї мети поставлено такі завдання.

- Дослідити існуючі інформаційні системи та проаналізувати процес їх розробки.
- Сформулювати вимоги до інформаційної системи для художників «JoinArt».
- Розробити технічне завдання до продукту.
- Розробити серверну частину інформаційної системи «JoinArt».
- Розробити інтерфейс та дизайн інформаційної системи «JoinArt».
- Протестувати розроблений програмний продукт за допомогою автоматичних тестів та власного досвіду використання.

Об’єкт, методи й засоби розроблення. Об’єктом розроблення інформаційної системи «JoinArt» є процес створення спеціалізованої інформаційної системи для художників.

Розробці інформаційної системи передувало визначення вимог до майбутнього продукту. Основу для цього склав аналіз вимог до схожих інформаційних систем, а також власний досвід у художній сфері.

Під час розробки була використана каскадна модель розробки програмного забезпечення. Дана модель передбачає поділ процесу створення програми на декілька етапів, та перехід до кожного наступного етапу лише після завершення попереднього. Для розробки була обрана саме така модель, оскільки при такій моделі легко здійснювати управління проектом, а розробка проходить швидко, не потребує великих фінансових витрат та кожен етап закінчується вчасно.

В якості інструментів створення інформаційної системи була обрана Microsoft Visual Studio Community 2019 16.9.4 – інтегроване середовище розробки мовою програмування C# – для розробки серверної частини, а також JetBrains WebStorm 2021.1.1 – інтегроване середовище розробки мовою програмування JavaScript та мовами розмітки HTML5, CSS3 – для розробки інтерфейсу інформаційної системи.

Серверна частина була створена засобами фреймворку ASP.NET Core, який надає багатий набір класів та бібліотек для створення серверного застосунку та розгортання його у мережі. Інтерфейс інформаційної системи, у свою чергу, був створений засобами JavaScript-фреймворку React, який був обраний через те, що він значно пришвидшує роботу веб-застосунку завдяки тому, що одразу завантажує усі потрібні ресурси на сторінку.

Можливі сфери застосування. Інформаційна система «JoinArt» може застосовуватися для комерційного використання певною організацією, яка б надавала послуги для полегшення процесу створення та реалізації картин художників.

РОЗДІЛ 1 ІНФОРМАЦІЙНІ СИСТЕМИ ЯК ВАЖЛИВА СКЛАДОВА ЛЮДСЬКОЇ ДІЯЛЬНОСТІ

1.1 Визначення інформаційної системи

Інформаційна система – це сукупність технічних та людських ресурсів, що забезпечують зберігання, обчислення, розподіл та передачу інформації, яка необхідна для функціонування деякої організації. Також під інформаційною системою розуміють набір обладнання, що задіяне у зборі, обробці, зберіганні та розповсюдженні інформації.

Апаратне, програмне забезпечення, інформація та зв’язки комп’ютерної системи, користувачі інформаційної системи, місце її знаходження – це все частини інформаційної системи. Персональні комп’ютери, смартфони, бази даних та мережеві пристрої – це лише складові частини інформаційної системи, які дозволяють їм функціонувати та надають користувачам доступ до інформації, що в ній знаходиться [1].

Підприємства та корпорації використовують інформаційні системи для взаємодії зі своїми постачальниками та клієнтською базою, здійснювати певні операції всередині організації, управляти корпорацією та проводити маркетингові кампанії. Інформаційні системи можна використовувати для найрізноманітніших цілей, починаючи від управління ланцюгами поставок і до взаємодії з цифровими ринками. Крім того, люди покладаються на інформаційні системи при взаємодії з однолітками та друзями через соціальні мережі, при здійсненні повсякденної діяльності, такої як банківська справа та покупки, або просто шукаючи інформацію.

Деякі популярні телевізійні шоу насправді можна розглядати як дивовижний та несподіваний приклад інформаційної системи. У шоу талантів телевізійній аудиторії пропонують віддати свій голос, щоб визначити, які кандидати проходять до наступного етапу. Система, яка фіксує та реєструє голоси – це апаратний пристрій, телефонні дзвінки, повідомлення або опитування в Інтернеті, за допомогою яких голоси фіксуються та правила, за

допомогою яких вираховують, хто саме покине шоу – це процес, а люди, що голосують – це користувачі, які залучені до системи [2].

1.2 Компоненти інформаційної системи

Хоча інформаційні системи можуть відрізнятися за тим, як вони використовуються в організації, вони, як правило, складаються з наступних компонентів:

1. апаратне забезпечення – комп'ютерні інформаційні системи використовують комп'ютерне обладнання, таке як, наприклад, процесори, монітори, клавіатура та принтери;
2. програмне забезпечення – це програми, які використовуються для організації, обробки та аналізу даних;
3. бази даних – інформаційні системи працюють з даними, що упорядковані у таблиці та файли;
4. мережа – різні елементи повинні бути пов'язані між собою, особливо якщо багато різних людей в організації використовують одну і ту ж інформаційну систему;
5. процедури – вони описують, як обробляються та аналізуються конкретні дані, щоб отримати відповіді, для знаходження яких розроблена інформаційна система.

Перші чотири компоненти є частиною загальних інформаційних технологій організації. Процедури, п'ятий компонент, дуже специфічні для інформації, необхідної для відповіді на конкретні питання [3].

1.3 Класифікація інформаційних систем

Тип або категорія інформаційної системи – це концепція, абстракція, яка була створена як спосіб спрощення складного та загального поняття інформаційної системи за допомогою виявлення спільних властивостей між різними речами. Більшість способів класифікувати інформаційні системи на різні типи покладаються на спосіб розподілу завдань та відповідальності в організаціях. Оскільки більшість організацій є ієрархічними, спосіб класифікації різних класів інформаційних систем має тенденцію слідувати ієрархії. Це часто

описується як «модель піраміди», оскільки спосіб упорядкування систем відображає характер завдань, що знаходяться на різних рівнях організації.

Чотирирівнева пірамідальна модель різних типів інформаційних систем, що зображена на рисунку 1, відображає різні рівні ієрархії в організаціях. Перший рівень представляє системи обробки транзакцій для робітників. Другий рівень представляє інформаційні системи управління для менеджерів середнього рівня. На третьому рівні знаходяться системи підтримки прийняття рішень для керівників вищого рівня. Четвертий рівень представляє виконавчі інформаційні системи для керівників [4].

Рисунок 1 – Чотирирівнева пірамідальна модель класифікації інформаційних систем

Системи обробки транзакцій – це системи операційного рівня внизу піраміди. Зазвичай ними керують безпосередньо співробітники, які надають ключові дані, необхідні для підтримки управління операціями. Ці дані зазвичай отримують за допомогою автоматизованого або напівавтоматизованого відстеження діяльності на низькому рівні та базових транзакцій.

З історичних причин багато різних типів інформаційних систем, котрі знаходяться в комерційних організаціях, називають інформаційними системами управління. Однак, в рамках наведеної пірамідальної моделі, інформаційні системи управління – це системи рівня управління, які використовуються менеджерами середнього рівня, для того щоб допомогти забезпечити безперебійне функціонування організації в короткостроковій та

середньостроковій перспективі. Структорована інформація, що надається даними системами, дозволяє менеджерам оцінювати результати діяльності організації, порівнюючи поточні та попередні результати, та на їх основі робити висновки щодо діяльності організації.

Інформаційну систему підтримки прийняття рішень можна розглядати як систему, засновану на знаннях, які використовуються старшими менеджерами, які в свою чергу сприяють створенню таких знань та дозволяють інтегрувати їх в організацію. Ці системи часто використовуються для аналізу існуючої структурованої інформації та дозволяють менеджерам проектувати потенційні наслідки своїх рішень у майбутньому. Такі системи, як правило, є інтерактивними і використовуються для вирішення неправильно структурованих проблем. Вони надають доступ до баз даних, аналітичних інструментів, дозволяють моделювати випадки «що якщо», а також можуть підтримувати обмін інформацією в межах організації.

Виконавчі інформаційні системи – це інформаційні системи стратегічного рівня, які знаходяться на вершині піраміди. Вони допомагають керівникам та головним менеджерам аналізувати середовище, в якому функціонує організація, визначати довгострокові тенденції та планувати відповідні напрямки дій. Інформація в таких системах часто слабо структурована і надходить як із внутрішніх, так і із зовнішніх джерел. Виконавча інформаційна система розроблена для безпосереднього управління керівниками без потреби посередників і легко адаптується до уподобань особи, яка її використовує [5].

1.4 Особливості інформаційних систем

Ключовою особливістю будь-якої інформаційної системи є дані. Дані – це інформація, що зберігається в необробленому вигляді. Причина, чому інформація зберігається саме таким чином, полягає в тому, що повинна використовуватися лише та інформація, яка потрібна для певних функцій. Наприклад, якщо компанія зберігає інформацію про один із своїх основних продуктів, і виникла потреба дізнатися номер товару, немає сенсу знаходити тридцятисторінковий документ про товар. За допомогою інформаційної системи

користувач може шукати по назві товару та отримувати у відповідь лише його номер. Діапазон різних елементів даних можна використовувати разом, щоб потім сформувати результат, який включає в себе лише необхідні дані [6].

Люди – це ще одна ключова особливість інформаційних систем. Вони повинні розуміти, як система працює, щоб максимізувати потенціал даних, що зберігаються в інформаційній системі. По-перше, технічні працівники повинні фактично розробляти інформаційну систему і повинні мати навички щодо проектування та програмування баз даних, щоб вимоги до системи могли бути успішно реалізовані. Керівництво повинно приймати рішення про те, як працює інформаційна система, виходячи з видів звітів та інформації, необхідної для аналізу та успішного ведення бізнесу. Співробітники різних відділів повинні інформувати керівний та технічний персонал про те, що вони вимагають від інформаційної системи.

Для створення інформаційної системи може знадобитися цілий ряд програмних пакетів. Програмне забезпечення є ключовим компонентом будь-якої інформаційної системи, оскільки саме воно дозволяє кінцевому користувачу отримати доступ до системи введення та пошуку інформації. Ключовим програмним компонентом будь-якої інформаційної системи, безумовно, буде база даних або система керування базами даних. База даних буде використовуватися для зберігання даних, до яких можна отримати доступ у різних формах, щоб зробити вибірку актуальної та потрібної інформації шляхом виконання складних запитів у базі даних.

Якщо інформаційна система створюється як багатокористувацька система, то буде створено додатковий застосунок, що, ймовірно, означатиме, що база даних працюватиме у фоновому режимі і не може бути доступною безпосередньо. Зверху працюватиме прикладне програмне забезпечення, яке дозволить кінцевому користувачу отримати доступ до даних, що зберігаються в системі. Різні мови програмування, такі як Python, C#, Java тощо, можуть бути використані для написання програм, призначених для інтерфейсу інформаційної

системи. Ці програми будуть розроблені таким чином, щоб максимально спростити використання системи для прикінцевого користувача.

Підсумовуючи, варто зазначити, що існує багато ключових особливостей інформаційної системи, які повинні працювати разом, щоб система могла успішно виконувати свої завдання. Технічним працівникам потрібно постійно контролювати, робити резервні копії та оновлювати систему, щоб цінні дані не втрачалися. Для проектування системи в залежності від бюджету слід використовувати найкращі апаратні та програмні додатки та мережеві технології.

1.5 Етапи розробки інформаційних систем

Розробка інформаційної системи – це надзвичайно важливий процес, який визначає функціонал інформаційної системи та способи її використання. Розробка інформаційної системи складається із наступних етапів: визначення вимог, аналіз, проектування, реалізація, тестування, впровадження та супровід та оцінка ефективності [7].

Розробка будь-якої системи починається з постановки задачі. Інформаційна система, як правило, створюється для великої кількості користувачів, кожен з яких пропонує власні вимоги до системи. На етапі визначення вимог необхідно виявити всіх потенційних користувачів інформаційної системи, і для кожного з них скласти список вимог до неї. Так будуть сформовані основні функціональні вимоги до системи.

На етапі аналізу створюється аналітична модель, яка структурує функціональні вимоги до системи. Вона описує вже внутрішній вигляд системи, використовуючи мову розробників. Вона являє собою аналіз кожного варіанта використання і визначає його подальшу реалізацію.

Етап проектування – це найбільш трудомісткий етап розробки інформаційної системи. На цьому етапі необхідно розробити проекційну модель всієї системи в цілому і кожного з її блоків. Для кожного завдання, яке буде реалізоване в рамках інформаційної системи, необхідно описати можливі методи його вирішення. Ці методи слід порівняти між собою за критеріями, які є

значимими з точки зору системи, і на підставі цього обрати кращий з них. Саме такий метод згодом повинен бути реалізований у програмі. Також на цьому етапі відбувається проектування бази даних. Складні інформаційні системи, як правило, структуровані, тобто є сукупністю декількох функціональний блоків. На етапі проектування повинна бути чітко описана функціональність кожного з блоків, а також повинен бути обґрунтований вибір методів інтеграції блоків у єдиний інформаційний комплекс.

На етапі реалізації відбувається безпосередньо написання програми за допомогою обраної мови програмування. В технічному завданні повинен бути обґрунтований вибір саме цієї мови, а також вибір системи керування базами даних та інших програмних засобів.

На етапі тестування необхідно перевірити коректність функціонування системи в нормальний умовах функціонування (коли в систему вводяться коректні вхідні дані), в граничних умовах (коли на вхід подаються допустимі, але рідко використовувані, або граничні параметри) і в екстремальних умовах (коли на вхід системи подаються некоректні дані). Модель тестування повинна описувати результати, які були отримані при обробці всіх цих даних.

На етапі впровадження та супроводу відбувається забезпечення стабільної роботи і зниження ризиків виникнення збоїв у роботі інформаційних систем, оперативне виправлення технічних неполадок в роботі систем, надання нових версій, оновлення та доповнення системи. Також відбуваються консультації з питань експлуатації та адміністрування інформаційних систем, консультації по встановленню новий версій, оновлень, доповнень тощо.

На етапі оцінки ефективності інформаційної системи збираються відгуки у клієнтів про процес використання системи і виявляються вимоги щодо поліпшення її роботи.

1.6 Значення інформаційних систем

Бізнес-корпорації та інші організації покладаються на інформаційні системи для здійснення діяльності та управління нею, взаємодії зі своїми клієнтами та постачальниками, а також конкуренції на ринку. Інформаційні

системи використовуються для управління міжорганізаційними ланцюгами поставок та електронними ринками. Наприклад, корпорації використовують інформаційні системи для обробки фінансових рахунків, управління своїми людськими ресурсами та охоплення своїх потенційних клієнтів за допомогою таргет-реклами. Досить багато великих компаній повністю побудовані навколо інформаційних систем. До таких компаній відносяться eBay – ринок аукціонів, Amazon – електронний торговий центр і постачальник послуг хмарних обчислень, Alibaba – електронний ринок для бізнесу та Google – компанія-пошуковий рушій, яка отримує більшу частину свого доходу від реклами, що налаштована на пошук по ключовим словам в Інтернеті.

За останнє тисячоліття було винайдено дуже багато нових технологій запису та обробки інформації, що значно просунули людство у напрямку створення перших інформаційних систем. Так винахід друкарні Йоганнесом Гутенебергом у середині 15 століття та винахід механічного калькулятора Блезом Паскалем у 17 столітті – це лише два яскравих приклади. Ці винаходити призвели до глибокої революції у здатності фіксувати, обробляти, поширювати інформацію та знання. Це, у свою чергу, призвело до ще глибших змін в приватному житті, організації бізнесу та управлінні людьми. Першою широкомасштабною механічною інформаційною системою була таблиця перепису Германа Холлеріта. Винайдена для перепису населення США в 1890 році, машина Холлеріта зробила важливий крок у напрямку до автоматизації, а також показала прагнення до розробки комп’ютеризованих інформаційних систем.

Одним із перших комп’ютерів, що використовувався для подібної обробки інформації, був UNIVAC I, встановлений в Бюро перепису США в 1951 році для адміністративного використання та в головному офісі компанії General Electric у 1954 році для комерційного використання. Починаючи з кінця 1970-х років, персональні комп’ютери принесли переваги інформаційних систем малому бізнесу та приватним особам.

На початку того ж десятиліття Інтернет розпочав своє розширення як глобальна мережа мереж. У 1991 році Всесвітня павутина, винайдена Тімом Бернерсом-Лі як засіб доступу до взаємопов'язаної інформації, що зберігається у глобально розосереджених комп'ютерах, під'єднаних до Інтернету, почала працювати та стала основною послугою, що надається у мережі. Глобальне розповсюдження Інтернету дозволило отримати доступ до інформації та інших ресурсів та сприяло формуванню стосунків між людьми та організаціями у небаченому досі масштабі. Прогрес електронної комерції через Інтернет призвів до різкого зростання цифрових міжособистісних комунікацій (за допомогою електронної пошти та соціальних мереж), розподілу продуктів (програмного забезпечення, музики, електронних книг та фільмів) та бізнес транзакцій (покупка, продаж і реклама в Інтернеті).

Так як інформаційні системи зробили людську діяльність більш різноманітною, вони мали глибокий вплив на суспільство. Ці системи пришвидшили темп повсякденної діяльності, дозволили людям розвивати та підтримувати нові стосунки, впливали на структуру та поєднання організацій, змінювали тип закуповуваних продуктів та впливали на характер роботи. Інформація та знання стали життєво важливими економічними ресурсами [7].

1.7 Огляд спеціалізованих інформаційних систем для художників

Здійснивши пошук у мережі Інтернет серед інформаційних систем, які спеціалізуються на художній сфері, не вдалося виявити багато прикладів, які б надавали художникам потрібних можливостей для полегшення процесу написання картин та їх продажу. До найбільш популярних прикладів таких систем можна віднести прості каталоги картин, магазини, в яких можна придбати всілякі засоби для створення картин, системи-довідники, в яких можна знайти інформацію про картини відомих художників, жанри та художні напрями. Порівняння даних систем можна здійснювати с точки зору функціональності систем та зручності графічних інтерфейсів.

Однією із найбільш багатофункціональних систем у даній сфері є інформаційна система «Artfinder». Дана система надає користувачам можливість

створити власний акаунт користувача, додавати свої картини до системи, а також купувати картини інших користувачів та продавати власні. Крім того, даний застосунок надає можливості для фільрування картин за різними жанрами, надає список найбільш популярних художників та має легку навігацію, яка дозволяє досить легко повернутися до переглянутих раніше картин. До недоліків даної системи можна віднести відсутність статистики, яку міг би отримувати художник про свої картини, що б полегшувало процес створення картин, а також відсутність системи аукціонів, де користувачі могли б змагатися за право придбати найкращі картини, що б дозволяло художникам отримувати більші прибутки за продаж своїх картин.

Також варта уваги британська інформаційна система «GreatArt», яка надає художникам дуже широкий асортимент різних інструментів та аксесуарів для малювання. У каталозі даної системи представлені всі можливі засоби, що можуть знадобитися при створенні будь-яких видів картин, від пензлів та фарб до полотен та мольбертів. Але дана система має головний недолік – вона не надає можливостей художникам додавати до системи свої картини та продавати їх з її допомогою.

Крім того варто відмітити інформаційні системи «Saatchi Art» та «ArtGallery», які мають більш широку спеціалізацію та дозволяють користувачам додавати до системи не лише свої картини, а і фотографії, замальовки та скульптури. Крім того, користувачі даних систем можуть вести художні блоги, де мають змогу розміщувати будь-яку інформацію стосовно їхньої художньої діяльності. Серед недоліків систем також варто зазначити відсутність корисної статистики про популярність доданих картин.

Таким чином, Інтернет пропонує користувачам досить невелику кількість інформаційних систем для художників. Основними критеріями при порівнянні можуть бути зручність пошуку в системі, зручність механізму продажу картин та наявність статистики, яка б допомагала у їх створенні та просуванні. У цілому ж, такі системи є досить динамічними, адже потребують постійної підтримки, а тому не має потреби їх порівнювати більш детально.

РОЗДІЛ 2 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ХУДОЖНИКІВ «JOINART»

2.1 Формування вимог до інформаційної системи

Першим та одним із головних етапів розробки інформаційної системи є формування вимог до інформаційної системи. Важливість даного етапу досить складно недооцінити, адже він безпосередньо впливає на подальші етапи розробки. Вимоги до системи – це опис послуг, які повинна надавати інформаційна система, а також обмеження, накладені на її функціональності [8]. Ці вимоги відображають потреби користувачів, які система повинна задовольнити. Тому на даному етапі постало завдання у виявленні, аналізі та документуванні тих можливостей та послуг, які повинна надавати інформаційна система «JoinArt».

Під час етапу формування вимог до інформаційної системи можуть виникати деякі проблеми внаслідок того, що даний етап є досить великим та громіздким. Для того щоб полегшити процес відокремлення вимог, вимоги розділяють на дві головні групи – користувацькі вимоги – для позначення абстрактних вимог високого рівня – та системні вимоги – для детального опису того, що повинна робити система.

Користувацькі вимоги – це твердження на природній мові та діаграми, які описують можливості, що система повинна передбачати для користувачів інформаційної системи, та обмеження, за яких вона повинна працювати. Користувацькі вимоги можуть варіюватися від загального опису необхідних функцій системи до більш детальних та точних описів функціональних можливостей системи.

Системні вимоги – це більш детальний опис функцій, послуг та експлуатаційних обмежень інформаційної системи. Для кожної системи, як правило, розробляється документ системних вимог, який також називають ще функціональною специфікацією, в якому повинно бути чітко визначено, які функціональні можливості мають бути впроваджені. Наприклад, це може бути

частиною договору між замовником системи та розробниками програмного забезпечення.

Крім цього, вимоги в свою чергу поділяються на функціональні та нефункціональні:

1. функціональні вимоги – це перелік послуг, які повинна надавати система, як система повинна реагувати на певні дії користувачів та як система повинна поводитися в конкретних ситуаціях.
2. нефункціональні вимоги – це обмеження на послуги, запропоновані системою, та характеристики, які вона повинна мати. Вони включають в себе обмеження по часу, обмеження процесу розробки та обмеження, накладені стандартами розробки. Нефункціональні вимоги часто застосовуються до системи в цілому, а не до окремих функцій системи або послуг.

На практиці розмежування між різними типами вимог не настільки чіткі, як зазначено у визначеннях. Користувацька вимога, що стосується безпеки, наприклад, вимога, щоб існувало обмеження в доступі до деяких послуг для неавторизованих користувачів може здатися нефункціональною вимогою. Однак при більш детальному розгляді даної вимоги, можна дійти до висновку, що вона може породжувати інші вимоги, які є чітко функціональними, наприклад, необхідність включення до системи засобів аутентифікації користувачів.

Це показує, що вимоги не є незалежними, і що одна вимога часто породжує або обмежує інші вимоги. Отже, вимоги не просто визначають необхідні послуги або особливості системи, вони також визначають необхідну функціональність для забезпечення ефективного надання цих послуг та функцій.

2.1.1 Функціональні вимоги

Функціональні вимоги до системи фактично описують, що система повинна робити. Ці вимоги залежать від деяких факторів, таких як: тип програмного забезпечення, що розробляється, очікувані користувачі програмного забезпечення та загальний підхід, який застосовується при

написанні вимог. Якщо функціональні вимоги також є користувацькими вимогами, вони повинні бути написані природною мовою, щоб користувачі системи та менеджери могли їх зрозуміти. Функціональні системні вимоги розширюють користувацькі вимоги і написані для розробників системи. Вони повинні детально описувати функції системи, їх вхідні та вихідні дані та винятки [9].

Функціональні користувацькі вимоги задокументовані за допомогою use-case UML діаграми на рисунку 2.

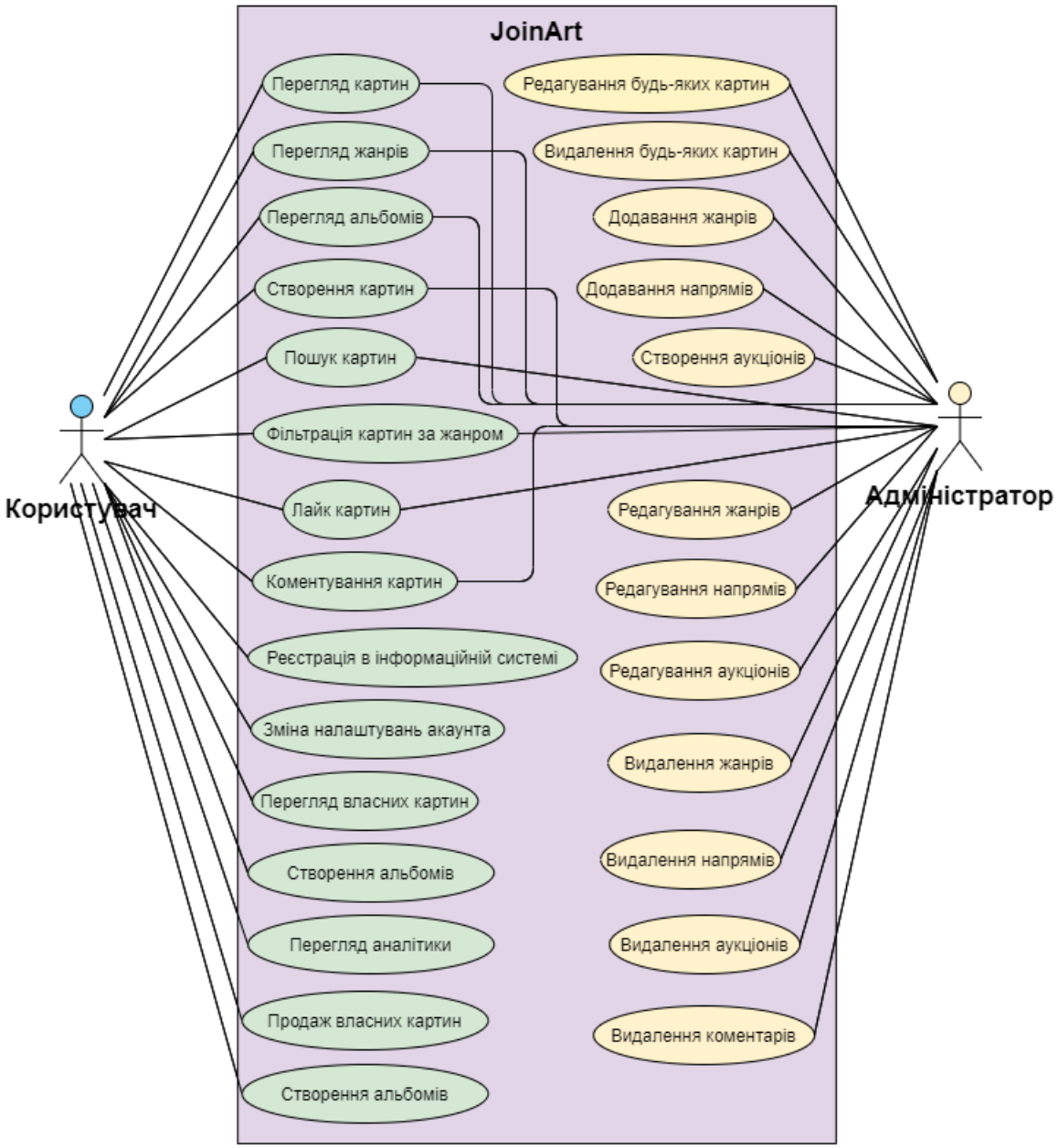

Рисунок 2 – Use-case діаграма функціональних користувацьких вимог

Функціональні системні вимоги варіюються від загальних вимог, тобто тих, де описано, що повинна робити система, до цілком конкретних вимог, що відображають певні сценарії роботи та організації інформаційної системи. До функціональних системних вимог до системи «ArtStyle», що використовується для збереження інформації про картини та їх художників, а також просування і популяризації картин серед користувачів даної системи, відносяться:

1. користувач повинен мати можливість здійснювати пошук серед всіх картин – система має надавати користувачу спеціальну форму для введення деякого рядка і після введення користувачем здійснити пошук серед усіх картин та видати в якості результату всі картини, які містять в якості підрядка рядок, що був введений користувачем;
2. користувач має мати змогу фільтрувати картини за певними показниками (стиль, жанр, ціна, популярність тощо) – система повинна надавати користувачу обирати за якими характеристиками будуть відфільтровані та відсортовані всі картини в інформаційній системі, повинна бути реалізована фільтрація за стилями, жанрами, ціною, популярністю;
3. користувачі повинні мати змогу переглядати картини один одного, ставити вподобання та залишати коментарі – система повинна реалізовувати можливість, щоб будь-які користувачі мали доступ до перегляду всіх картин, що були додані до інформаційної системи іншими користувачами. А також авторизовані в системі користувачі повинні мати змогу показувати своє відношення до картин, що вони переглядають, за допомогою вподобань та коментарів. Перед тим як додати коментар до картини, інформаційна система має проаналізувати чи коментар, що додається, відповідає вимогам інформаційної системи до правил взаємоввічливої взаємодії користувачів;
4. власник картини повинен мати змогу додавати свої картини до системи, встановлювати для них певні параметри (стиль, жанр, ціна, популярність тощо), редагувати їх та видаляти із системи;

5. система повинна надавати власникам картин виставляти свої картини на аукціон, а користувачам змогу взяти участь в аукціоні та придбати картину – інформаційна система повинна надавати можливість авторизованим користувачам продавати свої картини, а також купувати картини інших користувачів. Для цього повинна існувати система аукціонів, де для кожної картини можна створити окремий аукціон, де користувачі можуть пропонувати свою ціну і таким чином картину купує користувач, який запропонував найбільшу ціну;
6. авторизовані користувачі повинні мати змогу створювати добірки картин та групувати їх за будь-якими характеристиками, що бажає користувач – в інформаційній системі повинна існувати можливість створювати добірки картин – альбоми. Кожен авторизований користувач повинен мати змогу створити альбом, надати йому назву та опис, а також додавати в нього будь-які картини, які забажає користувач, таким чином згрупувавши їх за певними зручними характеристиками;
7. система повинна щодня для кожного авторизованого користувача генерувати статистику щодо перегляду його картин, кількість позитивних відміток, коментарів, та на основі даної інформації видавати поради власнику картин, як їх краще просувати. На основі глобальної статистики повинні видаватись поради для художників, які особливості картин є найбільш популярними та що саме краще створювати надалі;
8. кожен авторизований користувач має мати унікальний логін та пошту, за допомогою яких можна ідентифікувати особу – інформаційна система повинна бути реалізована таким чином, щоб кожен авторизований користувач був унікальним.
9. система повинна гарантувати безпеку даних, щоб сторонні особи не мали доступу до персональних даних користувачів та не могли завантажити зображення картин – важливою частиною інформаційної системи повинен бути безпековий прошарок, повинна бути реалізована

надійна система захисту даних користувачів, мінімізована ймовірність втрати або злому даних.

Крім того, функціональні системні вимоги можуть бути викладені у формі sequence UML діаграми. У додатку А зображена функціональна вимога для процесу звернення користувача до веб-сервісу з метою поставити лайк картині. У додатку Б зображена функціональна вимога для процесу звернення адміністратора до веб-сервісу з метою створити новий жанр картин.

2.1.2 Нефункціональні вимоги

Нефункціональні вимоги – це вимоги, які безпосередньо не стосуються конкретних послуг, що надає система своїм користувачам. Нефункціональні вимоги зазвичай визначають або обмежують характеристики інформаційної системи в цілому. Вони можуть стосуватися критичних властивостей системи, таких як надійність, час відгуку та використання пам’яті. Також вони можуть визначати обмеження щодо реалізації системи, такі як можливості пристроїв вводу-виводу або представлення даних, що використовуються разом з іншими системами [10].

Нефункціональні вимоги виникають через потреби користувачів внаслідок обмеженого бюджету, політики організації, необхідності взаємодії з іншими програмними чи апаратними системами або зовнішніх факторів, таких як правила безпеки чи законодавство про конфіденційність.

Інформаційна система «JoinArt» має наступні нефункціональні вимоги:

1. інформаційна система має бути реалізована як веб-сервіс, працювати у браузерах за протоколом HTTP, а також бути доступною для будь-яких користувачів, у яких є доступ до мережі Інтернет;
2. необхідність швидкої роботи алгоритмів видачі картин є критичною, тому система повинна працювати у два етапи, щоб забезпечити максимальну швидкість роботи:
    а) прийом даних і відправлення їх на обробку у асинхронному режимі;

b) обробка даних, у результаті якої обіцяний користувачу ресурс стає доступним.

Очікування результату тривалістю в 1 хвилину вважається допустимим, адже за цей час повинні виконатись всі операції отримання та обробки даних. Важливо наголосити на потенційній розширюваності інформаційної системи, оскільки передбачено використання системою часу на обробку інформації;

3. веб-інтерфейс інформаційної системи повинен бути реалізований за допомогою JavaScript-фреймворку React у зв'язку із простотою підтримуваності такого рішення. Серверна частина інформаційної системи має бути побудована за допомогою фреймворку ASP.NET Core, що пояснюється тим, що він є одним із найшвидших та найпродуктивніших на ринку;
4. інформаційна система має використовувати базу даних PostgreSQL та хмарний сервіс Azure Blob Storage як сховища даних із мінімальним необхідним обсягом роботи для його підтримання та супроводу;
5. веб-сторінка має проходити тест на адаптивність, передбаченою сервісом «responsivedesignchecker.com». Користувацький інтерфейс інформаційної системи має бути доступним на мобільних та десктопних пристроях;
6. результати виданих відповідей на запити користувача повинні кешуватися та зберігатися на сервері не більше ніж протягом однієї доби з моменту надходження запиту на сервер. Дотримання верхньої межі на обсяг інформації, що зберігається, зумовлене обмеженістю дискового простору серверу.

2.2 Проектування архітектури інформаційної системи

Архітектура інформаційної системи – це концепція, що визначає модель, структуру, виконувані функції та взаємозв'язок компонентів інформаційної системи. Іншими словами, архітектура інформаційної системи – це абстрактне

поняття, що визначає, з яких складових частин (елементів, компонентів) складається додаток і як ці частини між собою взаємодіють.

Під складовими частинами інформаційної системи, як правило, розуміють програми або програмні модулі, що виконують окремі, відносно ізольовані задачі. Компоненти інформаційної системи за виконуваними функціями розділяють на три шари:

1. шар представлення (користувацький інтерфейс) – все, що пов’язано з взаємодією з користувачем: натискання кнопок, рухи комп’ютерної миші, рендеринг зображення, вивід результатів пошуку тощо;
2. бізнес логіка – правила, алгоритми реакції програми на дії користувача або на внутрішні події, правила обробки даних;
3. шар доступу до даних – зберігання, вибір, модифікація та видалення даних, пов’язаних з вирішуваною інформаційною системою прикладною задачею.

В якості архітектури для інформаційної системи «JoinArt» через свої переваги, такі як повна підтримка багатокористувацької роботи та гарантії цілісності даних була обрана клієнт-серверна архітектура.

2.2.1 Клієнт-серверна архітектура

Клієнт-серверна архітектура – це обчислювана та мережева архітектура, в якій завдання та мережеве навантаження розділені між постачальниками послуг – серверами, та замовниками послуг – клієнтами. Клієнт-серверна система також відома як модель обчислень мережі, оскільки всі послуги та запити виконуються за допомогою мережі. Її робота подібна до розподілених обчислюваних систем, так як всі вузли виконують незалежне завдання. Це спільна обчислювана мережа, де різні віддалено підключені пристрої надсилають запити до централізованої системи – сервера або хост-машини. Клієнтська частина пропонує зручний інтерфейс для користувачів, який дозволяє відвідувачам запускати серверні сервісні запити та відображати результат обчислень на клієнтському пристрої [11].

Частина програми, де користувачі можуть здійснювати безпосередню взаємодію з програмою, називається стороною клієнта. Клієнтська сторона працює як інтерактивний front end інтерфейс, щоб користувачу було інтуїтивно зрозуміло, які послуги він бажає отримати від інформаційної системи виконавши певні дії. З іншого боку серверна сторона реалізує back end частину програми. Веб-браузери, що використовуються для отримання веб-сторінок, є клієнтською стороною, крім цього, існує сервер, з якого буде надіслана запитувана веб-сторінка та будуть надані дані для відображення. Багато мов програмування побудовані для роботи як на стороні клієнта, так і на стороні сервера.

Робота кожної архітектури вимагає декількох компонентів. Аналогічно архітектура клієнт-сервер базується на трьох взаємопов'язаних компонентах:

1. робоча станція – це система користувачів, які здійснюють запити на сервер;
2. сервер – надзвичайно ефективний пристрій, який має дуже велику швидкість обробки даних, накопичувач з великим об'ємом пам'яті для зберігання значної кількості інформації, а також великий об'єм оперативної пам'яті для обробки декількох запитів, що надходять одночасно з різних робочих станцій. Одночасно сервер може виконувати декілька різних функцій, наприклад, бути поштовим сервером, сервером баз даних, файловим сервером, контролером домену;
3. мережеві пристрої – це пристрої, за допомогою яких робочі станції та сервери взаємопов'язані між собою. Мережеві пристрої, що використовуються в архітектурі клієнт-сервер, мають свої функції та властивості. Для встановлення з'єднання сервера з декількома робочими станціями використовуються хаби, для передачі даних з пристрою на інший пристрій використовуються ретранслятори, для ізольованої сегментації мережі використовуються мости.

В архітектурі клієнт-сервер клієнти розглядаються як користувачі або споживачі, тоді як сервер працює як виробник. Клієнт часто вимагає від сервера

високоякісних обчислювальних послуг для виконання вимог. Сервери надають клієнтові різні послуги, що включають в себе доступ до певних програм, зберігання даних, спільний доступ до файлів, використання обчислюваної потужності за допомогою сервера. Користувач на своєму боці відправляє запит на сервер за допомогою мережі, який обробляється на різних етапах, а потім дані, які є результатом обробки, доставляються назад користувачеві відповідно до поданого запиту. Більш того, сервери можуть обробляти запити декількох споживачів одночасно. Тим часом клієнт повинен залишатися на зв'язку з серверами, на які було відправлено запити, оскільки кожен сервер надає послуги відповідно до свого домену і повинен знати куди відправляти результат запиту.

Переваги клієнт-серверної архітектури:

1. існує централізована мережа, яка має всю повноту управління процесами та діями в архітектурі клієнт-сервер;
2. мережевий захист, резервне копіювання та всі інші відповідні елементи налаштовуються централізовано;
3. користувачі мають право доступу до всіх файлів, що знаходяться у головному сховищі, у будь-який момент часу;
4. не існує обмежень щодо доступу до інформації з точки зору географії. Доступ до будь-якої інформації можна отримати з будь-якого місця;
5. архітектуру можливо розширити, оскільки зростання вимагає масштабування користувачів або будь-яких інших параметрів;
6. архітектура передбачає використання сервісів, що забезпечують авторизацію користувача для перегляду даних, а також обмеження прав для модифікації або видалення даних із системи;
7. клієнт-серверна архітектура реалізована на основі розподіленої моделі, забезпечує заміну, оновлення та переміщення сервера без впливу на клієнта;
8. дана архітектура може справлятися з дуже великою кількістю клієнтів, систем, пристроїв та навантажень мережі;

9. забезпечується простий та інтуїтивно зрозумілий користувацький інтерфейс, процедуру пошуку файлів та систему управління для збережених файлів.

Недоліки та обмеження клієнт-серверної архітектури:

1. користувачі постраждають, якщо основний сервер вийде з ладу;
2. архітектура вимагає певної операційної системи, пов’язаної з мережею;
3. конфігурація апаратних компонентів та програмних засобів вимагає великих витрат;
4. для обслуговування мережі, особливо серверних пристроїв, потрібен висококваліфікований персонал;
5. численні одночасні запити можуть спричинити проблеми з перевантаженнями.

2.2.2 Структурування системи «JoinArt»

На першому етапі процесу проектування архітектури система розбивається на декілька підсистем, що взаємодіють між собою. На найбільш абстрактному рівні архітектуру інформаційної системи можна представити графічно за допомогою блок-схеми, в якій підсистеми зображені окремими блоками. Якщо підсистему можна розділити на кілька частин, на діаграмі такі частини будуть зображені прямокутниками всередині великих блоків. Дані, які передаються між підсистемами будуть позначатися стрілками. Така діаграма компонентів дає загальне уявлення про структуру системи.

Хоча з такої блок-схеми не вийде нічого дізнатися ні про природу взаємин між компонентами, ні про їх властивості, однак, такі моделі виявляються ефективними на етапі попереднього проектування системи. Така діаграма не перевантажена деталями, з її допомогою зручно представити структуру системи. У структурній моделі визначені всі основні підсистеми, які можна розробляти незалежно від інших підсистем, отже, можна поділити розробку інформаційної системи на декілька етапів, в кожному з яких буде розроблена певна підсистема.

В інформаційній системі «JoinArt» на основі функціональних системних вимог було виділено вісім відносно незалежних підсистем. Їх відображає діаграма компонентів, зображена на рисунку 3.

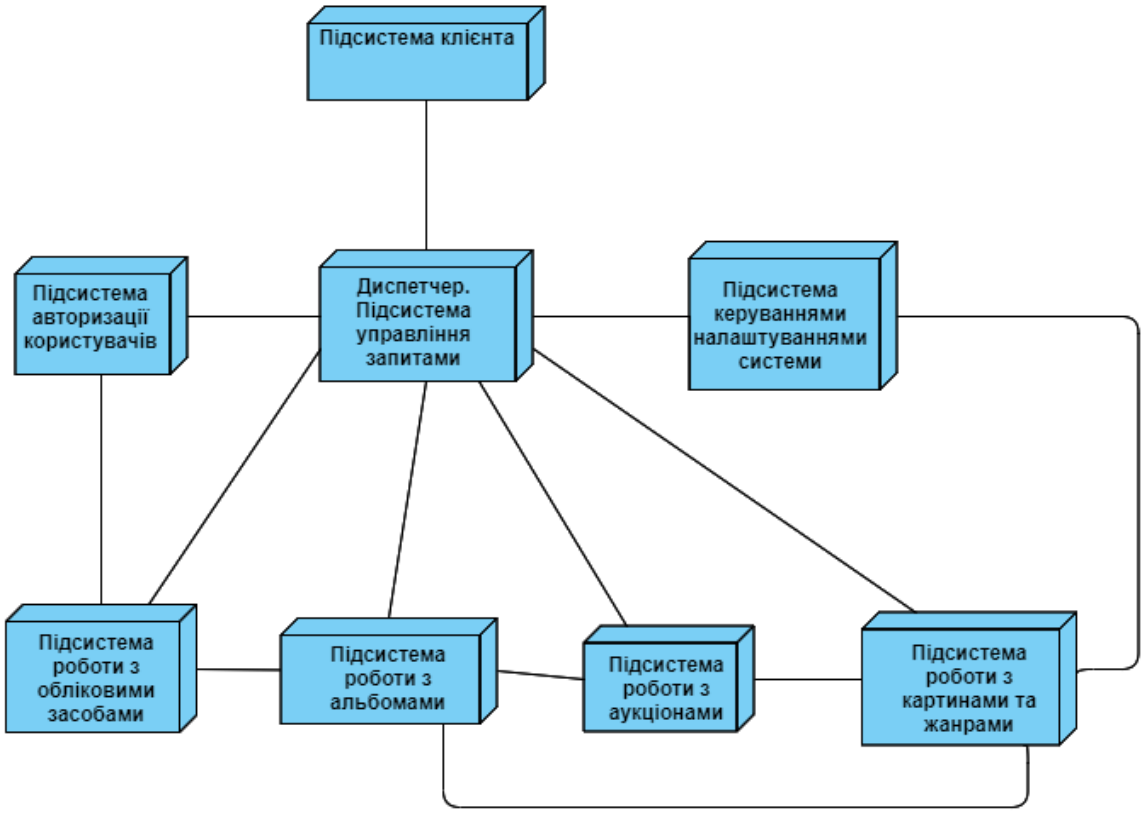

Рисунок 3 – Архітектура системи «JoinArt» в контексті поділу на підсистеми

За структурою система є клієнт-серверним додатком з одним сервером, тобто зі спільною базою даних для різних її підсистем. Переважна частина важливої для системи бізнес-логіки реалізована на серверній частині, тому на діаграмі розглядається саме її структурування. Відповідні серверним підсистеми також реалізовані на клієнтській частині, але через розподілену структуру додатку, клієнтські підсистеми не обов’язково збігаються з серверними, проте повинні реалізовувати для користувача доступ до них, тож серверні та клієнтські підсистеми можна ототожнити.

Передбачається використання однієї бази даних. Вона не відображена на діаграмі, оскільки кожна підсистема реалізує свій інтерфейс доступу до бази даних, та база даних не є функціональною підсистемою. Діаграма призначена для загального розуміння функцій системи та їх логічного розподілу.

Функції підсистем:

1. підсистема клієнта – користувач робить запити через клієнтський інтерфейс у вигляді веб-застосунку;
2. диспетчер. Підсистема управління запитами – головний контролер, що розподіляє запити між підсистемами;
3. підсистема авторизації користувачів – перевірка прав користувача на виконання дії в системі, реєстрація, зміна пароля;
4. підсистема керування налаштуваннями системи – реалізація керування функцією про кількість картин, що виводиться на сторінці, інші системні налаштування;
5. підсистема роботи з картинами та жанрами – пошук, додавання, редагування, видалення, завантаження картин та їх жанрів;
6. підсистема роботи з аукціонами – створення, видалення аукціонів, прийняття в них участі користувачами, а також купівля та продаж картин;
7. підсистема роботи з альбомами – створення, редагування та видалення альбомів, додавання картин до альбомів та їхнє видалення;
8. підсистема роботи з обліковими записами – декоратор для пошуку, додавання та видалення облікових записів, редагування персональної інформації користувачів.

2.2.3 ER-модель бази даних системи «JoinArt»

ER-модель (Entity-relationship model або Entity-relationship diagram) – це семантична модель даних, яка призначена для спрощення процесу проектування бази даних. В основі ER-моделі лежать поняття «сутність», «зв’язок» і «атрибут». З ER-моделі можуть бути породжені всі види баз даних: реляційні, ієрархічні, мережеві, об’єктні.

ER-модель бази даних інформаційної системи «JoinArt» зображена у додатку В.

Таблиці, з яких складається база даних інформаційної системи «JoinArt»:

1. таблиця Users – таблиця, де наведена інформація про зареєстрованих користувачів системи;
2. таблиця Roles – таблиця, де наведена інформація про ролі користувачів системи. Кожна роль має свої особливі права доступу до певних функцій системи;
3. таблиця Follows – таблиця, де наведена інформація про користувачів, що відслідковують активність інших користувачів;
4. таблиця Paintings – таблиця, де наведена інформація про картини, що знаходяться в системі;
5. таблиця Albums – таблиця, де наведена інформація про альбоми, створені користувачами;
6. таблиця AlbumPaintings – таблиця зв’язку, де наведена інформація про картини та альбоми, до яких вони відносяться;
7. таблиця Genres – таблиця, де наведена інформація про жанри та напрями картин;
8. таблиця GenrePaintings – таблиця зв’язку, де наведена інформація про картини та жанри, до яких вони відносяться;
9. таблиця PaintingsUsers – таблиця зв’язку, де наведена інформація про картини та користувачів, які їх додали;
10. таблиця Likes – таблиця, де наведена інформація про вподобання, що були поставлені картинам користувачами;
11. таблиця Comments – таблиця, де наведена інформація про коментарі, що були додані картинам користувачами;
12. таблиця BidProducts – таблиця, де наведена інформація про картини, що виставлені на аукціони;
13. таблиця BidBets – таблиця, де наведена інформація про аукціони та користувачів, які зробили ставки на відповідних аукціонах;
14. таблиця Orders – таблиця, де наведена інформація про замовлення та користувачів, які їх зробили;

15. таблиця OrderPaintings – таблиця зв’язку, де наведена інформація про картини та замовлення, до яких вони відносяться.

2.2.4 Моделювання управління системи «JoinArt»

У моделі структури системи показані всі підсистеми, з яких вона складається. Для того, щоб підсистеми функціонували як єдине ціле, необхідне централізоване управління ними. У структурних моделях не наведено жодної інформації по управлінню, однак розробляючи архітектуру системи, потрібно організовувати підсистеми згідно деякої моделі управління, яка б доповнювала наявну модель структури. У моделях управління на рівні архітектури проектується потік управління між підсистемами. Можна виділити два основних типи управління в програмних системах:

1. централізоване управління – одна з підсистем повністю відповідає за управління, запускає і завершує роботу інших підсистем. Управління від однієї підсистеми може перейти до іншої, проте потім обов’язково повертається до першої;
2. управління, засноване на подіях – замість однієї підсистеми, відповідальної за управління, на зовнішні події може відповідати будь-яка підсистема. Події, на які реагує система, можуть відбуватися або в інших підсистемах, або в зовнішньому оточенні системи.

Інформаційна система «JoinArt» реалізує управління, засноване на подіях, та може бути описаною моделлю передачі повідомлень. Це відображено на діаграмі компонентів у додатку Г.

На відміну від діаграми на рисунку 3 на діаграмі у додатку Г не відображені зв’язки між підсистемами, оскільки всі зв’язки відобразити неможливо та це є деталлю реалізації, тому це відведено безпосередньо на етап розробки додатку.

2.2.5 Модульна декомпозиція системи «JoinArt»

Після етапу розробки системної структури в процесі проектування слідує етап декомпозиції підсистем на окремі модулі. Між розділенням системи на підсистеми і підсистем на модулі немає принципових відмінностей. Однак

компоненти модулів зазвичай менше компонентів підсистем, тому можна використовувати спеціальні моделі декомпозиції.

Існують дві моделі, які використовуються на етапі модульної декомпозиції підсистем:

1. об’єктно-орієнтована модель – система складається з набору взаємодіючих об’єктів;
2. модель потоків даних – система складається з функціональних модулів, які отримують на вході дані і перетворюють їх певним чином у вихідні дані. Такий підхід також називають конвеєрним.

В об’єктно-орієнтованій моделі модулі є об’єктами з власними станами та певними операціями над цими станами. Натомість в моделі потоків даних модулі виконують функціональні перетворення. В обох моделях модулі реалізовані або як послідовні компоненти, або як процеси.

У додатку Ґ зображена модульна декомпозиція системи «JoinArt». За допомогою діаграми компонентів відображено декомпозицію в межах підсистем на функціональні модулі, тобто такі, що забезпечують одну або дуже близькі функції системи.

2.3 Засоби для розробки інформаційної системи

Засоби для розробки інформаційної системи – це сукупність прийомів, методів, методик, а також набір інструментальних програм (компілятори, прикладні та системні бібліотеки тощо), які використовуються для створення програмного коду системи, яка відповідає заданим раніше вимогам.

Розробка програми – це складний процес, основною метою якого, крім створення та супроводження програмного коду, також є забезпечення необхідного рівня якості та надійності системи. Тому досить важливо на даному етапі обрати найбільш надійні програмні інструменти, які б забезпечували якомога простіший процес створення програмного забезпечення, а також безпроблемну подальшу модифікацію та підтримку інформаційної системи.

2.3.1 Засоби для роботи з даними

Основним сховищем даних інформаційної системи «JoinArt» була обрана база даних, а саме PostgreSQL – це система керування базами даних, яка використовує реляційну модель для своїх баз даних та підтримує стандартну мову запитів SQL. PostgreSQL надає безліч різних можливостей, вона досить надійна та має високі характеристики продуктивності. Вона працює практично на всіх UNIX-платформах, включаючи UNIX-подібні системи, такі як FreeBSD та Linux. Крім того PostgreSQL вільно поширюється і має відкритий вихідний код.

PostgreSQL вигідно відрізняється від багатьох інших систем керування базами даних. Вона володіє всіма можливостями, які є в інших базах даних, а також деякими додатковими. Серед її можливостей можна відмітити: транзакції, вкладені запити, представлення, цілісність посилань – зовнішні ключі, складні блокування, типи, що задаються користувачами, спадковість, правила, перевірка сумісності версій.

Продуктивність PostgreSQL також заслуговує відзнаки, останні перевірки показують, що вона не поступається комерційним продуктам. Однак деякі системи, що володіють не таким повним набором можливостей перевершують PostgreSQL з точки зору продуктивності, але за рахунок втрати функціональності.

Однією із сильних сторін PostgreSQL є її архітектура. Як і багато комерційних систем управління базами даних, PostgreSQL може застосовуватися в архітектурі клієнт-сервер, що дає масу переваг як користувачам, так і розробникам.

Основа PostgreSQL становить серверний процес бази даних. Він виконується на одному сервері. Доступ з додатків до бази даних здійснюється за допомогою процесу бази даних. Клієнтські програми не можуть отримати доступ до даних самостійно, навіть якщо вони працюють на тому ж комп’ютері, на якому виконується серверний процес. Такий поділ клієнтів і сервера дозволяє побудувати розподілену систему. Можна відокремити клієнтів від сервера за

допомогою мережі і розробляти клієнтські програми в середовищі, зручному для користувача. Наприклад, можна реалізувати базу даних під UNIX та створити клієнтські програми, які будуть працювати в системі Microsoft Windows.

Завдяки тому, що маніпулювання даними зосереджено на сервері, PostgreSQL не доводиться контролювати численних клієнтів, які отримують доступ в каталозі сервера, що спільно використовується. PostgreSQL може підтримувати цілісність даних навіть при одночасному доступі великої кількості користувачів.

Клієнтські додатки з'єднуються з базою по спеціальному протоколу PostgreSQL. Проте можна встановити на стороні клієнта програмне забезпечення, яке надає стандартний інтерфейс для роботи потрібної програми, наприклад, за стандартом ODBC або JDBC. Доступність ODBC-драйвера дозволяє застосовувати PostgreSQL в якості бази даних для багатьох існуючих додатків [12].

Крім бази даних PostgreSQL для зберігання картинок використовується хмарне сховище від Microsoft – Azure BLOB Storage. Сховище BLOB-об'єктів Azure – це служба зберігання великої кількості неструктурованих даних та BLOB (Binary Large Objects) об'єктів на хмарі, таких як текстові або двійкові дані (наприклад, документи, файли мультимедіа або програмні компоненти), до яких можна отримати доступ практично з будь-якої точки світу по протоколу HTTP або HTTPS. Сховища BLOB-об'єктів можна використовувати для надання даних у відкритому доступі або для конфіденційного зберігання даних, що генеруються прикладним програмним забезпеченням [13].

Найбільш часті способи використання сховища BLOB-об'єктів:

1. обслуговування зображень або документів безпосередньо у браузері;
2. зберігання файлів для розподіленого доступу;
3. потокова передача відео та аудіо;
4. зберігання резервних копій та відновлення даних, аварійне відновлення та архівування;

5. зберігання даних для аналізу локальною службою або службою, розміщеною в Azure.

Також була використана система контролю версій Git для того, щоб полегшити процес розробки, зберігаючи дані централізовано в одному місці на віддаленому сервері, до якого завжди є доступ, що дозволило уникнути будь-якої втрати даних у процесі розробки інформаційної системи.

2.3.2 Засоби для розробки back end частини

Для розробки back end частини інформаційної системи було обрано фреймворк ASP.NET Core з використанням мови C# та Visual Studio в якості IDE (Integrated Development Environment). ASP.NET Core (Active Server Pages .NET Core) – це відкрита універсальна платформа розробки, яка підтримується корпорацією Microsoft і спільнотою .NET. Вона є кросплатформенною, підтримує Windows, Mac OS та Linux і може використовуватися на пристроях, на хмарах, у впроваджених системах і в сценаріях IoT (Інтернету речей). В її основі лежать технології .NET Framework і Silverlight. Вона оптимізована для мобільних та серверних робочих навантажень, оскільки забезпечує підтримку самодостатніх розгортувань XCOPY.

ASP.NET не обмежується мовами сценаріїв, він дозволяє використовувати мови .NET, такі як C#, J#, VB тощо. Він дозволяє створити дуже продуктивні застосунки, використовуючи Visual Studio – інструмент розробки, що надається компанією Microsoft. ASP.NET – це суто серверна технологія, вона побудована на загальномовному середовищі виконання, яке можна використовувати на будь-якому сервері Windows для розміщення потужних веб-сайтів і веб-додатків. ASP.NET виконується на стороні сервера, а його вихідні дані відправляються на веб-браузер користувача [14].

Головні переваги ASP.NET:

1. ASP.NET значно скорочує обсяг коду, необхідного для створення великих додатків;
2. завдяки вбудованій аутентифікації Windows та окремому налаштуванню для кожної програми, всі дані надійно захищені;

3. він забезпечує більш високу продуктивність завдяки використанню переваг раннього зв’язування, своєчасної компіляції, вбудованої оптимізації і служб кешування;
4. всі процеси ретельно відстежуються і управляються середовищем виконання ASP.NET, тому, якщо процес не працює, на його місці може бути створений новий процес, який допомагає підтримувати постійну доступність застосунку для обробки запитів;
5. будучи незалежним від мови, він дозволяє обирати мову, яка найкраще підходить для програми або розділити додаток на декілька мов;
6. ASP.NET полегшує розгортання, немає необхідності реєструвати компоненти, тому що інформація про конфігурацію вбудована;
7. веб-сервер постійно відслідковує компоненти і програми, що працюють на ньому. Якщо він помічає будь-які витоки пам’яті, нескінченні цикли, інші незаконні дії, він негайно руйнує ці дії і перезапускає себе.

Крім цього, інтегроване середовище розробки, за допомогою якого працюють з фреймворком ASP.NET Core, Visual Studio пропонує ряд високорівневих функціональних можливостей, які виходять за рамки базового управління кодом [15]. Серед таких можливостей:

1. вбудований веб-сервер – для обслуговування веб-додатків ASP.NET необхідний веб-сервер, який буде очікувати веб-запити і обробляти відповідні сторінки. Наявність в Visual Studio інтегрованого веб-сервера дозволяє запускати застосунок одразу з середовища проектування, а також підвищує безпеку, виключаючи ймовірність отримання доступу до тестового веб-додатку з якого-небудь зовнішнього комп’ютера, оскільки тестовий сервер може приймати з’єднання лише з локального комп’ютера;
2. інтуїтивний стиль кодування – за замовчуванням Visual Studio форматує код у міру його введення, автоматично вставляючи необхідні відступи і застосовуючи колірне кодування для виділення елементів на кшталт коментарів. Такі незначні відмінності роблять код більш

зручним для читання. Автоматично застосовувані Visual Studio параметри можна налаштовувати, що дуже зручно у випадках, коли, наприклад, розробник вважає за краще інший стиль розміщення дужок;

3. більш висока швидкість розробки – багато з функціональних можливостей Visual Studio спрямовані на те, щоб допомагати розробнику робити свою роботу якомога швидше. Зручні функції, на зразок функції IntelliSense (перехоплення помилок та пропонування правильних варіантів), функції пошуку і заміни (відшуковування ключових слів як в одному файлі, так і в усьому проекті) і функції автоматичного додавання та видалення коментарів (дозволяє тимчасово приховувати блоки коду), дозволяють розробнику працювати швидко і ефективно;

Visual Studio також має і безліч інших функцій: можливість управління проектом, вбудована функція управління вихідним кодом, можливість рефакторизації коду, потужна модель розширюваності. Більш того, в разі використання Visual Studio розробник отримує розширені можливості для модульного тестування, спільної роботи і управління версіями коду.

Крім того, для поліпшення швидкості пошуку картин в системі «JoinArt» використовується пошуковий рушій ElasticSearch – база даних NoSQL, яка основана на пошуковій системі Lucene і побудована на базі RESTful APIS. ElasticSearch має просту розгортуваність, максимальну надійність та просте керування. Він також пропонує розширені запити для детального аналізу та централізовано зберігає всі дані. Все це допомагає здійснювати швидкий пошук документів.

2.3.3 Засоби для розробки front end частини

Для розробки front end частини інформаційної системи було обрано JavaScript-фреймворк React та WebStorm в якості IDE. React – це бібліотека для створення користувацьких інтерфейсів. Однією з її характерних особливостей є можливість використання JSX – мови програмування з близьким до HTML синтаксисом, який компілюється в JavaScript код. Розробники можуть домогтися

більшої продуктивності додатків за допомогою Virtual DOM (Virtual Document Object Model). За допомогою React можна створювати застосунки, які допоможуть позбавитися ситуації, коли користувач мусить чекати, коли завершиться завантаження даних і на екрані комп'ютера з'явиться щось, крім анімації завантаження. Створені компоненти можуть бути з легкістю змінені і використані заново в нових проектах. Високий відсоток перевикористання коду підвищує здатність коду бути протестованим тестами, що, в свою чергу, призводить до більш високого рівня контролю якості [16].

Основні переваги фреймворку React:

1. Virtual DOM підвищує продуктивність високонавантажених додатків, що знижує ймовірність виникнення можливих незручностей і покращує користувацький досвід;
2. використання ізоморфного підходу допомагає виконувати рендеринг сторінок швидше, тим самим дозволяючи користувачам відчувати себе більш комфортно під час роботи з додатком. Пошукові системи індексують такі сторінки краще. Оскільки один і той же код може бути використаний як у клієнтській, так і в серверній частині програми, немає необхідності в дублюванні одного і того ж функціоналу. В результаті час розробки і витрати знижуються;
3. завдяки перевикористанню коду стало набагато простіше створювати мобільні додатки. Код, який був написаний під час створенні веб-сайту, може бути повторно використаний для створення мобільного застосунку. Немає необхідності створювати з самого початку мобільний додаток, а створювати його одночасно із сайтом.

WebStorm – середовище розробки для JavaScript, розроблене компанією JetBrains, що так само підходить для front end розробки, так і для створення Node.JS додатків. Його головною перевагою є зручний інтерфейс і розумний редактор JavaScript, HTML і CSS, який підтримує і багато інших мов, таких як TypeScript, CoffeeScript, Dart, Less, Sass та Stylus, а також JavaScript-фреймворки, наприклад, Angular, React і Meteor.

WebStorm робить розробку проекту простою та зручною, забезпечуючи підсвічування і автодоповнення коду, його аналіз по ходу редагування, швидку навігацію і рефакторинг. Він має потужні інструменти налагодження та інтеграції з системами управління версіями (Git, GitHub, Subversion, Perforce, Mercurial, CVS), розуміє структуру проекту і код, відстежує помилки за допомогою системи ESLint, JSHint, JSLint, TSLint, Stylelint і пропонує їх вирішення. Вбудовані в IDE інструменти для тестування і роботи з проектом допомагають в розробці і роблять її зручніше і продуктивніше.

Крім того, у WebStorm можна ефективно розробляти програми на Node.JS. Він підтримує повноцінне налагодження Node.JS додатків. Новий додаток можна створити, використовуючи шаблон Node.JS Express, а необхідні модулі встановити за допомогою встановленого в WebStorm менеджера npm [17].

Також для створення дизайну сайту була обрана CSS бібліотека Bootstrap, яка дозволяє використовувати вже готові стилі для створення найбільш приємного користувацького інтерфейсу. Для надсилання запитів на back end частину та отримання з неї відповідей використовується JavaScript- бібліотека Axios, яка дозволяє за допомогою HTTP запитів здійснювати зв’язок між програмами у мережі. А для красивого виводу користувачу проаналізованих даних за допомогою графіків використовується JavaScript- бібліотека Chart.js.

2.4 Розробка back end частини

На початку розробки серверної частини програми постало завдання розробити класи сутностей, які б відповідали таблицям із бази даних інформаційної системи. Для таких сутностей був створений namespace (простір імен) Models, у який було додано класи Album, Bid, Comment, Genre, Like, Order, Painting, Role та User.

На наступному кроці виникла потреба розробити механізм зв’язку програми з базою даних. Для цього було використано шаблон Data Access Object (DAO) – один із найбільш розповсюджених патернів проектування. Головна ідея цього шаблону – це реалізація методів для доступу до даних. Він відповідає за прошарок доступу до даних та реалізує методи, які займаються внесенням,

зміною, видаленням та вибіркою інформації з бази даних. Для розробки механізму зв'язку програми з базою даних був створений простір імен DataAccessLayer, в який були додані класи AlbumAdapter, AuctionAdapter, GenreAdapter, OrderAdapter, PaintingAdapter та UserAdapter, щоб відокремити логіку зв'язку з базою даних різних сутностей. В кожному з цих класів за допомогою бібліотеки Npgsql, яка реалізує зв'язок з базою даних PostgreSQL, були створені запити, за допомогою яких безпосередньо і відбувалася взаємодія з базою даних, а саме – додавались, редагувались та отримувались дані. В лістингу 1 наведений приклад запитів до бази даних.

```
private NpgsqlCommand CreateGetPaintingsOfGenreCommand(int genreId)
{
    string query = @"select *
                     from paintings
                     where paintingid in (
                            select paintingId
                            from paintinggenres
                            where genreId = @genreId)";

    var sqlCommand = new NpgsqlCommand(query);

    sqlCommand.Parameters.AddWithValue("genreId", genreId);

    return sqlCommand;
}
```

Лістинг 1 – Запит для отримання всіх картин жанру

Крім цього був розроблений клас PaintingSearch за допомогою якого здійснювався зв'язок програми з сервісом ElasticSearch. В ньому за допомогою бібліотеки Nest були реалізовані методи, які дозволяли здійснювати пошук за різними параметрами картин, такими як назва, опис, матеріали, ціна, жанр, напрям тощо. Дані методи дозволяють здійснювати пошук серед картин, що були проіндексовані у сервіс ElasticSearch на етапі запуску програми на сервері. В лістингу 2 наведений приклад запиту до ElasticSearch, за допомогою якого здійснюється пошук серед картин.

```
ISearchResponse<Painting> response = await client.SearchAsync<Painting>(search => search
    .Index(aliasName)
    .Query(query => query
        .Bool(boolean => boolean
            .Must(must => must
                .DisMax(dm => dm
                    .Queries(
                        qs => qs.MultiMatch(match => match
                            .Fields(fields => fields
                                .Field(field => field.Title)
                                .Field(field => field.Materials)
                                .Field(field => field.Description)
                            )
                            .Operator(Operator.And)
                            .Query(queryValue)
                            .Boost(1000000)
                        ),
                        qs => qs.ConstantScore(cs => cs
                            .Filter(f => f.Match(m => m
                                    .Field(field => field.Title)
                                    .Query(queryValue)
                                    .MinimumShouldMatch("100%")
                                )
                            )
                            .Boost(100000)
                        )
                    )
                )
            )
        )
    )
);
```

Лістинг 2 – Запит до ElasticSearch для пошуку картини по назві

Далі для реалізації бізнес логіки було створено namespace Providers. У даному просторі імен аналогічно з простором імен DataAccessLayer було розділено логіку різних сутностей в окремі класи, так було створено класи AlbumProvider, AuctionProvider, GenreProvider, OrderProvider, PaintingProvider та UserProvider. У кожному такому класі була реалізована логіка обробки даних для подальшої передачі їх на front end частину. В лістингу 3 наведений метод для отримання найпопулярніших альбомів в якості прикладу обробки даних.

```
public async Task<IList<Album>> GetTopAlbums(int take, bool withCache = false)
{
    if (take <= 0)
    {
        throw new ArgumentOutOfRangeException(nameof(take));
    }

    string cacheKey = "get_top_albums";

    IList<Album> topAlbums = withCache ? Helper.GetFromCache<IList<Album>>(this._cache, cacheKey) : null;

    if (topAlbums == null)
    {
        IList<Album> allAlbums = await GetAllAlbums();

        if (allAlbums != null)
        {
            topAlbums = allAlbums.Take(take).OrderByDescending(x => x.Rating).ToList();

            Helper.SetToCache(this._cache, cacheKey, topAlbums);
        }
    }

    return topAlbums;
}
```

Лістинг 3 – Метод для отримання найпопулярніших альбомів

Саме на цьому етапі був реалізований прошарок безпеки інформаційної системи. Були створені методи шифрування та дешифрування вразливих даних користувачів, таких як, наприклад, паролі. В якості алгоритму шифрування був використаний алгоритм SHA-256. SHA-256 – це криптографічна хеш-функція, розроблена Агентством національної безпеки США, розшифровується як Secure Hash Algorithm (Захищений Хеш Алгоритм). Сама по собі хеш-функція – це математична операція, що виконується над цифровими даними. Порівнюючи обчислений хеш (результат виконання алгоритму) з відомим та очікуваним хеш-значенням, можна визначити цілісність даних. Односторонній хеш може бути згенерований з будь-якого фрагменту даних, але дані не можуть бути згенеровані з хешу. Таким чином, при додаванні нового користувача до системи, його пароль буде шифруватися за допомогою алгоритму SHA-256 та у зашифрованому вигляді додаватися до бази даних. А коли буде необхідність отримати дані цього користувача, зашифрований пароль буде діставатися з бази даних та порівнюватися з попередньо зашифрованим паролем, що надійшов у якості ключа. В лістингу 4 зображено метод для верифікації паролю.

```
private bool VerifyPassword(string inputPassword, string passwordHash)
{
    using (SHA256 sha256Hash = SHA256.Create())
    {
        string hashOfInputPassword = GetHash(sha256Hash, inputPassword);

        StringComparer comparer = StringComparer.OrdinalIgnoreCase;

        return comparer.Compare(hashOfInputPassword, passwordHash) == 0;
    }
}
```

Лістинг 4 – Метод для верифікації паролю

Крім цього, на даному етапі була розроблена формула знаходження рейтингу конкретної картини. Функція підрахунку рейтингу картини має наступний вигляд:

$$R = BC + ViewsFactor * VC + LikesFactor * LC + CommentsFactor * CC$$

де BC – кількість ставок на аукціоні картини;

ViewsFactor – константа, множник переглядів картини, дорівнює 0.7;

VC – кількість переглядів картини;

LikesFactor – константа, множник вподобань картини, дорівнює 0.5;

LC – кількість вподобань картини;

CommentsFactor – константа, множник коментарів картини, дорівнює 0.3;

CC – кількість коментарів картини.

Рейтинг жанрів, напрямів та альбомів вираховується, виходячи із рейтингу картин, що входять до певного жанру, напряму або альбому, і дорівнює сумарному рейтингу картин, поділеному на їхню кількість. Рейтинг аукціонів дорівнює рейтингу картини, що знаходиться в даному аукціоні.

На останньому етапі розробки серверної частини постало завдання передавати спершу отримані з бази даних, а потім оброблені за допомогою провайдерів дані на front end частину програми. Для цього було вирішено скористатися контролерами, які досить легко реалізуються засобами фреймворку ASP.NET Core. Для того, щоб відокремити логіки передачі даних, що стосуються різних сутностей, були створені наступні контролери: AlbumsController, AuctionsController, GenresController, OrdersController, PaintingsController та UsersController. Кожному контролеру було присвоєно свій унікальний route (шлях), для того, щоб здійснювати запити саме на потрібний контролер. Крім цього, для кожного методу у контролері також було виділено унікальний в рамках контролера route. Таким чином, лише задавши правильний route, можна було отримати потрібні дані з back end частини. В лістингу 5 наведений приклад роуту, перейшовши по якому викликаються методи для додавання картини до інформаційної системи «JoinArt».

```
[HttpPost, Route("add-painting")]
public async Task<IActionResult> AddPainting(AddPaintingRequestModel model)
{
    if (model == null)
    {
        throw new ArgumentNullException(nameof(model));
    }

    AddPaintingResponseModel response = await _paintingProvider.AddPainting(model);

    return Json(response);
}
```

Лістинг 5 – Route для додавання картини до системи

2.5 Розробка front end частини

На першому етапі розробки front end частини виникло завдання отримати дані з back end частини для виводу їх користувачеві у зрозумілому вигляді. На допомого прийшла JavaScript-бібліотека – Axios, яку можна завантажити, скориставшись сервісом Node.JS. Вона дозволяє робити запити на потрібні роути, а у відповідь отримувати потрібні дані. Для цього завдання був розроблений пакет Services, в який були додані класи AlbumService, AuctionService, GenreService, OrderService, PaintingService та UserService. В кожному з цих класів були розроблені методи, які за допомогою бібліотеки Axios реалізують зв'язок з back end частиною, а саме, використовуючи потрібний HTTP метод, наприклад, GET, POST тощо, роблять запити на відповідні роути контролерів, що знаходяться на серверній частині, які, в свою чергу, повертають дані, які запитує front end частина. У лістингу 6 зображено метод для запиту даних по id картини з back end частини в якості прикладу.

```
static getPainting(paintingId){
    return new Promise( executor: async (resolve, reject) => {
        try {
            const url = `${settings.apiUrl}/paintings/get-painting/${paintingId}`;

            const res = await axios.get(url, config: {
                headers : {
                    'Content-Type' : 'application/x-www-form-urlencoded;'
                }
            });
            const data = res.data;

            resolve(data);
        } catch (e) {
            reject(e);
        }
    })
}
```

Лістинг 6 – Метод для запиту даних по id картини з back end частини

Оскільки питання про отримання даних з back end частини було вирішено, потрібно було переходити до розробки дизайну сайту. За допомогою програми Adobe Photoshop був створений макет дизайну застосунку. Далі потрібно було

зверстати розроблений дизайн за допомогою фреймворку React. Уся front end частина була поділена на окремі логічні частини – компоненти, кожен з яких відповідав за певну частину сайту. Так були створені компоненти для картин, жанрів, альбомів, спливаючих вікон та інших сутностей, що зустрічаються в інформаційній системі. Загалом було розроблено сорок два компоненти. В лістингу 7 зображено приклад компоненту.

```
import React, {Component} from 'react';
import '../componentsStyles/InfoPopup.css';

class InfoPopup extends Component{
    render() {
        return(
            <div className={'info-popup-container'} onClick={this.props.closePopup}>
                <div className={'info-popup'}>
                    {this.props.message}
                </div>
            </div>
        );
    }
}

export default InfoPopup;
```

Лістинг 7 – Компонент спливаючого вікна з повідомленням

Далі потрібно було кожному компоненту надати відповідний дизайн, який би був інтуїтивно зрозумілим кожному користувачеві. Для цього було використано мову CSS3 та CSS3 фреймворк Bootstrap. У додатку Д наведено приклад використання фреймворку Bootstrap для створення дизайну компонента.

На останньому етапі потрібно було зв’язати всі компоненти в одне ціле. Для цього спочатку було виділено сімнадцять головних компонентів, кожен з яких відповідав за певну логічну частину сайту – веб-сторінку. Так компонент PaintingPage відповідає за сторінку картини, Home – за головну сторінку, Login – за сторінку авторизації, Albums – за сторінку, на якій знаходяться всі альбоми, UserPage – за сторінку користувача тощо. Залишалось лише реалізувати навігацію між даними компонентами, щоб користувачі мали змогу

переміщуватися між сторінками. Для цього був використаний такий пакет React як React Router. З його допомогою можна зробити переключення компонентів за певними роутами, так компоненту PaintingPage був наданий роут «/painting/:id», Home – «/home», Login – «/login», Albums – «/albums», UserPage – «/user» тощо. В додатку Д зображений головний компонент, де можна побачити всі роути, які існують в розробленому додатку. При переході на певний роут відбувається рендер компонента, який знаходиться по даному роуті. У цей момент викликається потрібний метод із сервісного класу та завантажуються потрібні дані з back end частини. Далі засобами мови JavaScript дані обробляються та виводяться у компонентах.

2.6 Тестування інформаційної системи

Після того, як була завершена розробка back end та front end частини, постало завдання протестувати їх роботу. Так як продукт є клієнт-серверним додатком, його тестування полягає в перевірці правильності роботи окремо логіки front end частини, окремо логіки back end частини та інтеграційного тестування для атестації роботи системи в цілому.

Для того, щоб протестувати роботу інформаційної системи, були продумані та розроблені тести для перевірки відповідності системи функціональним та нефункціональним користувацьким та функціональним системним вимогам. Тести розділені на три частини: перші дві атестують вище наведені вимоги окремо в контекстах front end та back end частин, третя частина атестує вимоги при інтеграційному тестуванні. Таким чином, тестування будується на перевірці сценаріїв та структурується відповідно до раніше зазначених вимог до інформаційної системи «JoinArt».

Тестування front end частини виконується виключно в ручному режимі, інтеграційне тестування та тестування серверної частини – в ручному та автоматичному.

Тестування back end частини розпочалося з unit-тестів. Unit-тести – це модульні тести, які застосовуються на різних прошарках додатку. Вони тестують найменшу подільну логіку програми, наприклад, клас, але найчастіше – метод.

Такі тести зазвичай намагаються ізолювати від зовнішньої логіки роботи програми для того, щоб вони не впливали на її роботу, а лише допомагали виявити помилки, якщо такі присутні.

Таких тестів завжди має бути багато (більше, ніж інтеграційних тестів), оскільки вони тестують маленькі частини коду, яким не потрібно багато обчислюваних ресурсів. Так для того, щоб протестувати серверну частину інформаційної системи «JoinArt», було написано двадцять три unit-тести, які всі виконались та не виявили помилок. У лістингу 8 зображено приклад такого unit-тесту.

```
[TestMethod]
private async Task TestGetPaintingMethod()
{
    AddPaintingRequestModel paintingInfo = new AddPaintingRequestModel
    {
        OwnerId = OwnerId,
        Title = Title,
        Materials = Materials,
        Painter = Painter,
        Price = Price,
        Description = Description,
        Status = Status,
        Image = Image,
        GenresIds = GenresIds,
    };

    AddPaintingResponseModel addPaintingResponse = await _paintingProvider.AddPainting(paintingInfo);

    PaintingResponseModel painting = await _paintingProvider.GetPainting(addPaintingResponse.PaintingId, 0);

    await _paintingProvider.DeletePainting(addPaintingResponse.PaintingId);

    Assert.IsNotNull(painting);
    Assert.AreEqual(painting.Owner.UserId, OwnerId);
    Assert.AreEqual(painting.Title, Title);
    Assert.AreEqual(painting.Materials, Materials);
    Assert.AreEqual(painting.Painter, Painter);
    Assert.AreEqual(painting.Price, Price);
    Assert.AreEqual(painting.Description, Description);
    Assert.AreEqual(painting.Status, Status);
}
```

Лістинг 8 – Unit-тест для перевірки методу GetPainting

Але навіть якщо охопити unit-тестами майже всю логіку роботи застосунку, можуть виникати моменти, коли програма буде працювати не так, як потрібно. В таких випадках на допомогу приходять інтеграційні тести, вони перевіряють, щоб різні частини програми коректно працювали разом. На відміну від модульного тестування, інтеграційні тести часто запитують компоненти інфраструктури додатку, наприклад, базу даних, файлову систему, веб-запити та

відповіді тощо. На місці цих компонентів unit-тести використовують mock-об’єкти, а інтеграційні тести повинні перевіряти, чи правильно дані компоненти працюють у системі.

Оскільки інтеграційні тести працюють з великими сегментами коду і оскільки їм потрібні елементи інфраструктури, вони повільніші, ніж unit-тести. Тому не варто створювати занадто багато інтеграційних тестів, особливо якщо даний функціонал можна перевірити за допомогою unit-тестів. Так для того, щоб протестувати інформаційну систему «JoinArt», було написано сім інтеграційних тестів, кожен з яких виконався та не виявив помилок. У лістингу 9 наведено приклад інтеграційного тесту.

```
[Fact]
public async Task ControllerShouldReturnTopGenres()
{
    Mock<DataAccessSettings> dataAccessSettings = new Mock<DataAccessSettings>();
    Mock<IDataAccessAdapter> dataAccessLayer = new Mock<IDataAccessAdapter>(dataAccessSettings);
    Mock<IMemoryCache> cache = new Mock<IMemoryCache>();
    Mock<IPaintingProvider> paintingProvider = new Mock<IPaintingProvider>(dataAccessLayer, cache);
    Mock<IGenreProvider> genreProvider = new Mock<IGenreProvider>(dataAccessLayer, cache);
    Mock<IOptions<AppSettings>> settings = new Mock<IOptions<AppSettings>>();

    GenresController controller = new GenresController(
        dataAccessLayer.Object,
        cache.Object,
        settings.Object,
        genreProvider.Object,
        paintingProvider.Object);

    IActionResult result = await controller.GetTopGenres();

    result
        .Should()
        .BeOfType<IList<Genre>>()
        .And
        .NotBeNull();
}
```

Лістинг 9 – Інтеграційний тест для перевірки роуту, що повертає найпопулярніші жанри

2.7 Можливості інформаційної системи

Розроблений програмний застосунок надає користувачам інформаційної системи наступні можливості:

1. реєстрація та авторизація в системі;
2. користування особистим кабінетом користувача;
3. участь в аукціонах;

4. оформлення замовлень та придбання картин;
5. збереження даних користувачів у захищеному та зашифрованому вигляді;
6. пошук картин у системі за назвою;
7. перегляд інформації про жанри, напрями, альбоми та картин, що до них входять;
8. створення власних альбомів та додавання картин до них;
9. додавання власних картин до інформаційної системи;
10. розподіл функцій усередині системи в залежності від категорії користувача;
11. наявність функціоналу для керування інформацією про картини, жанри, напрями, альбоми, аукціони та користувачів (видалення, редагування, додавання нових записів) з боку адміністратора;
12. простий та зрозумілий інтерфейс для кожної ролі користувача задля спрощення навігації системою.

При переході на початкову сторінку користувач побачить головну сторінку, на якій по черзі виводяться найпопулярніші картини, аукціони, альбоми, жанри та напрями та усі картини, які знаходяться в системі. У верхній частині сторінки знаходиться назва інформаційної системи, форма для пошуку картин в системі по назві, а також кнопки для авторизації та реєстрації користувачів. Зліва знаходиться головне навігаційне меню, яке можна збільшити в разі потреби, натиснувши на кнопку у вигляді трьох горизонтальних ліній. У навігаційному меню знаходяться посилання на головну сторінку, сторінку жанрів, сторінку альбомів та сторінку аукціонів. Для того, щоб користувачу було зрозуміло, на якій сторінці він знаходиться, відповідна кнопка у меню підсвічується сірим та жовтим кольором. У додатку Є зображена головна сторінка інформаційної системи «JoinArt».

Перейшовши на сторінку авторизації, користувач має змогу зареєструватися, використовуючи свій акаунт в інших соціальних мережах чи звичайну пошту, або увійти зі створеним раніше обліковим записом. Користувач

має змогу це зробити через форму реєстрації або логіну. Сторінка реєстрації містить обов'язкові поля – поштова адреса, телефон, логін користувача, його ім'я та прізвище і пароль, який необхідно ввести повторно для підтвердження. Після реєстрації користувач повинен підтвердити свою пошту (на поштову скриньку надійде лист з посиланням для підтвердження). Після успішної реєстрації та її підтвердження необхідно перейти на сторінку входу, ввести логін та пароль, після чого користувача буде перенаправлено на стартову сторінку. У випадку збою, на екран виведеться повідомлення про помилку. На рисунку 4 зображена форма для авторизації користувача.

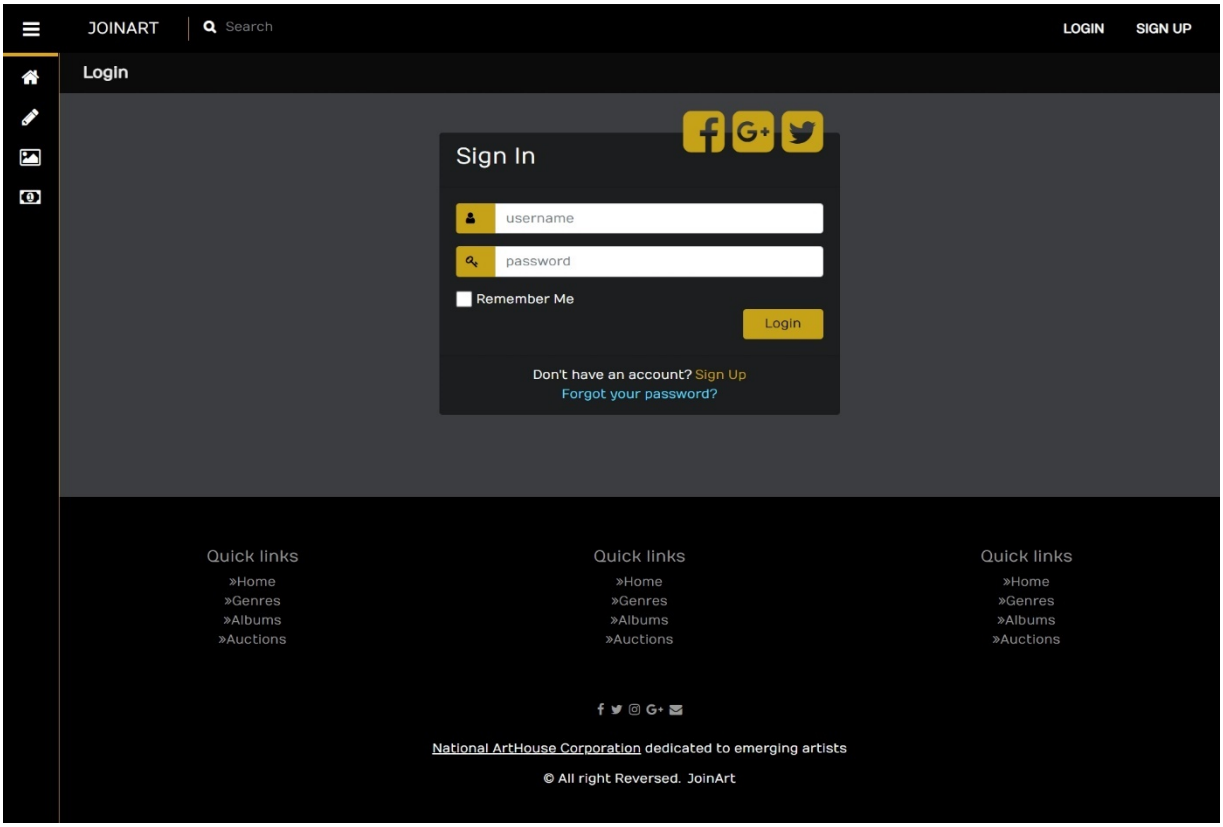

Рисунок 4 – Сторінка авторизації

В інформаційній системі «JoinArt» існує три ролі – звичайний користувач, авторизований користувач та адміністратор. Звичайний користувач має права лише переглядати картини, жанри, напрями, альбоми, а також здійснювати пошук серед картин. Натиснувши на відповідні кнопки у навігаційному меню, такий користувач може потрапити на сторінку жанрів та напрямів, де по черзі розташовані найпопулярніші жанри, найпопулярніші напрями та всі жанри та напрями. Сторінку жанрів можна побачити в додатку Ж. Сторінка альбомів

виглядає аналогічно до сторінки жанрів та напрямів, там по черзі розташовані найпопулярніші альбоми та всі альбоми.

Натиснувши на певний жанр або напрям, користувач потрапляє на сторінку жанру або напряму, де він може побачити титульне фото жанру або напряму, його опис та картини, що входять до даного жанру. Приклад сторінки жанру зображений у додатку З.

Написавши у формі пошуку потрібну назву або частину назви картини та натиснувши клавішу «Enter», користувач потрапляє на сторінку з результатами пошуку картин по назві, де йому будуть виведені всі картини, що задовольняють умові пошуку. Приклад сторінки пошуку зображений у додатку И.

Авторизований користувач має набагато розширені права, оскільки крім того, що може робити звичайний користувач, він може користуватися кабінетом користувача, додавати картини до системи, створювати альбоми та додавати до них картини, переглядати статистику, яку надає йому інформаційна система, слідкувати за оновленнями інших авторизованих користувачів, ставити вподобання та залишати коментарі, брати участь в аукціонах, купувати картини інших користувачів та продавати свої.

Натиснувши на певну картину, такий користувач потрапляє на сторінку картини. На цій сторінці він може побачити фото картини, її назву, автора, матеріали, з яких вона була написана, та опис картини. Крім того, користувач може бачити ціну картини та розпочати або долучитись до аукціону, вигравши який він зможе придбати картину. Також користувач може додати картину до будь-якого свого альбому, підписатися на оновлення користувача, який додав цю картину до системи, а також переглянути жанри та напрями, до яких відноситься картина. У разі якщо користувачу сподобалась картина, він може поставити їй вподобання, а також написати коментар будь-якого змісту, що не суперечить правилам ввічливого поводження в Інтернеті. Приклад сторінки картини зображений у додатку Ї.

Якщо користувач бажає взяти участь в аукціоні, він може у навігаційному меню натиснути на кнопку «Аукціони» та потрапити на сторінку аукціонів.

Сторінка аукціонів подібна до сторінок жанрів і напрямів та альбомів. Приклад сторінки аукціонів зображено у додатку Й.

Після переходу на сторінку аукціонів, користувач може обрати той аукціон, що йому найбільше подобається, та натиснути на нього. Таким чином він потрапить на сторінку аукціону, де може побачити фото та назву картини, яка розігрується на даному аукціоні, статус аукціону, чи він ще триває або завершився, кнопку для того, щоб розмістити свою ставку, а також історію ставок всіх користувачів, що беруть участь у цьому аукціоні. Аукціони тривають двадцять чотири години, і перемагає той користувач, який розмістив найбільшу ставку на момент завершення аукціону. Приклад аукціону зображено в додатку К.

Натиснувши на свій логін у правому верхньому куті сторінки, користувач потрапляє до кабінету користувача. У кабінеті користувача у лівій частині сторінки користувач може побачити всіх користувачів, на чиї оновлення він підписався, а також всіх користувачів, які підписалися на його оновлення. По центру сторінки зображена інформація користувача, така як логін, ім’я, стать, дата народження, телефон, адреса, країна та поштова адреса. Натиснувши на кнопку «Manage user info» користувач має змогу редагувати дану інформацію. З правого боку сторінки розташовані функціональні клавіші, які дають можливість додавати нову картину до системи, здійснювати управління вже доданими картинами, переглянути покупки користувача, переглянути альбоми користувача та переглянути аналітику, що надає інформаційна система користувачеві. Приклад сторінки користувача наведено у додатку Л.

Після того, як користувач натиснув клавішу «Add painting», йому відкривається форма для додавання нової картини до інформаційної системи. Користувач повинен заповнити поля з назвою, автором, ціною, матеріалами, описом та жанром картини, а також додати її фото. Після чого користувач має підтвердити додавання нової картини та її буде додано до інформаційної системи. Приклад форми для додавання картини зображено в додатку М.

Якщо користувач натиснув на кнопку «Manage paintings», він потрапляє на сторінку управління картинами, які він раніше додав до інформаційної системи. На цій сторінці у формі таблиці виводяться всі картини користувача, які можна відсортувати за ціною, назвою та популярністю. В таблиці, крім основної інформації про картини, такої як фото, назва, опис, ціна та кількість вподобань, також є клавіші, натиснувши на які, користувач може редагувати та видаляти свої картини з інформаційної системи. Приклад сторінки управління картинами зображено у додатку Н.

Натиснувши кнопку «Purchases», користувач потрапляє на сторінку з покупками. Сюди потрапляють всі картини, які користувач виграв на аукціонах. У вигляді таблиці тут зображена основна інформація про покупки, які здійснив користувач, а саме номер замовлення, фото картини, її назва та ціна замовлення, а також статус замовлення. Якщо користувач вже сплатив за замовлення, то в статусі замовлення написано «Sold», в іншому випадку там знаходиться кнопка «Pay», натиснувши на яку користувач може сплатити своє замовлення. Приклад сторінки покупок зображено в додатку О.

Якщо користувач натиснув на клавішу «Pay», йому відкривається вікно з чеком, де вказана основна інформація про замовлення та загальна ціна замовлення, яку необхідно сплатити, а також приємний бонус для покупця – передбачення на день. Після того, як користувач натисне кнопку «Submit» на чеку, його перекине на сторінку банку, де йому потрібно здійснити оплату за замовлення. Приклад чеку зображено на рисунку 5.

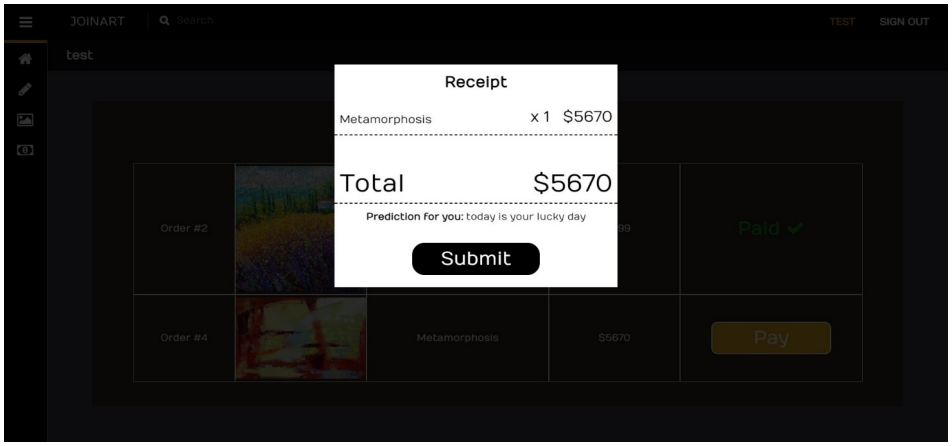

Рисунок 5 – Чек

Якщо користувач натиснув на кнопку «Albums», йому відкриється вікно з альбомами, які він створив. Натиснувши на будь-який альбом, користувач перейде на сторінку даного альбому. Крім того, біля кожного альбому розташовані клавіші, за допомогою яких користувач може або редагувати зміст альбому, або видалити альбом із системи. Також тут розташована кнопка для створення нового альбому. При натисканні на цю кнопку відкриється вікно, де, вказавши назву та опис альбому, користувач може додати новий альбом до системи. Приклад вікна з альбомами зображено на рисунку 6.

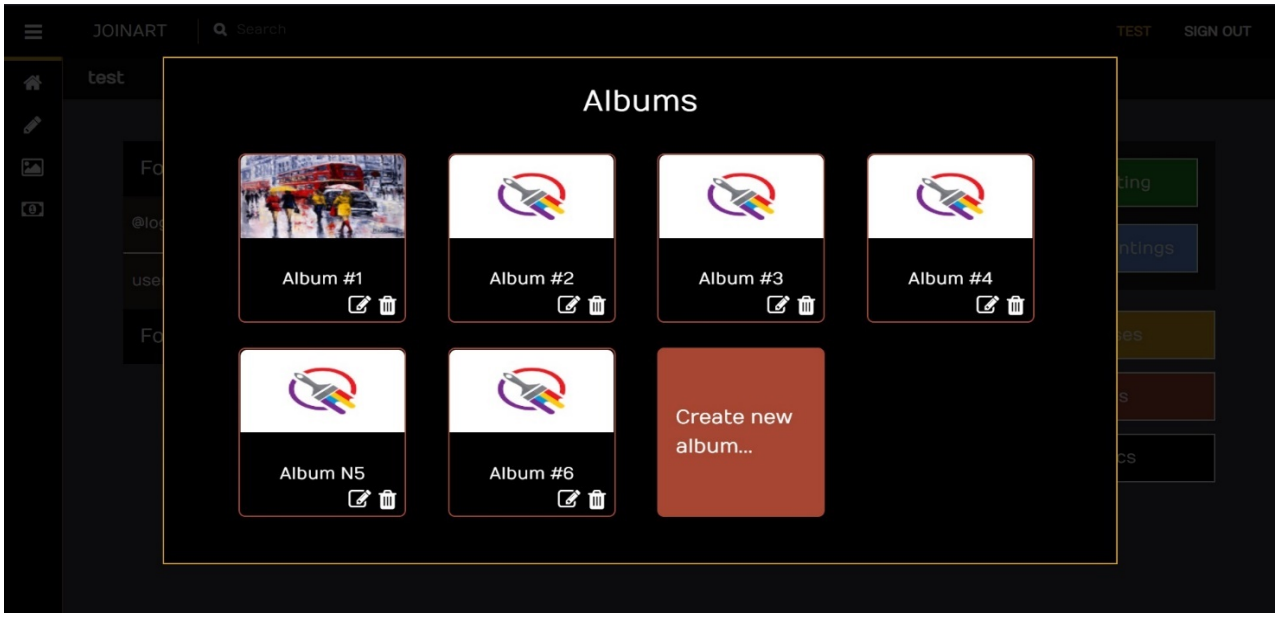

Рисунок 6 – Вікно з альбомами

Натиснувши на кнопку «Analytics», користувач перейде на сторінку з аналітикою, яку йому пропонує інформаційна система «JoinArt». Оскільки система зберігає інформацію про картини, їх перегляди та вподобання, вона може генерувати статистику, що може допомогти художникам у виборі тих напрямів та особливостей картини, які б з найбільшою ймовірністю сподобались користувачам. В системі реалізовано чотири графіки: графік найпопулярніших жанрів, графік найпопулярніших стилів, графік з рейтингом картин користувача та графік зміни переглядів картини за певний проміжок часу. Виходячи з цих даних, користувач може аналізувати, як інші користувачі приймають його картини, та на основі цієї інформації приймати рішення, які картини варто писати надалі. Приклад сторінки з аналітикою зображено на рисунку 7.

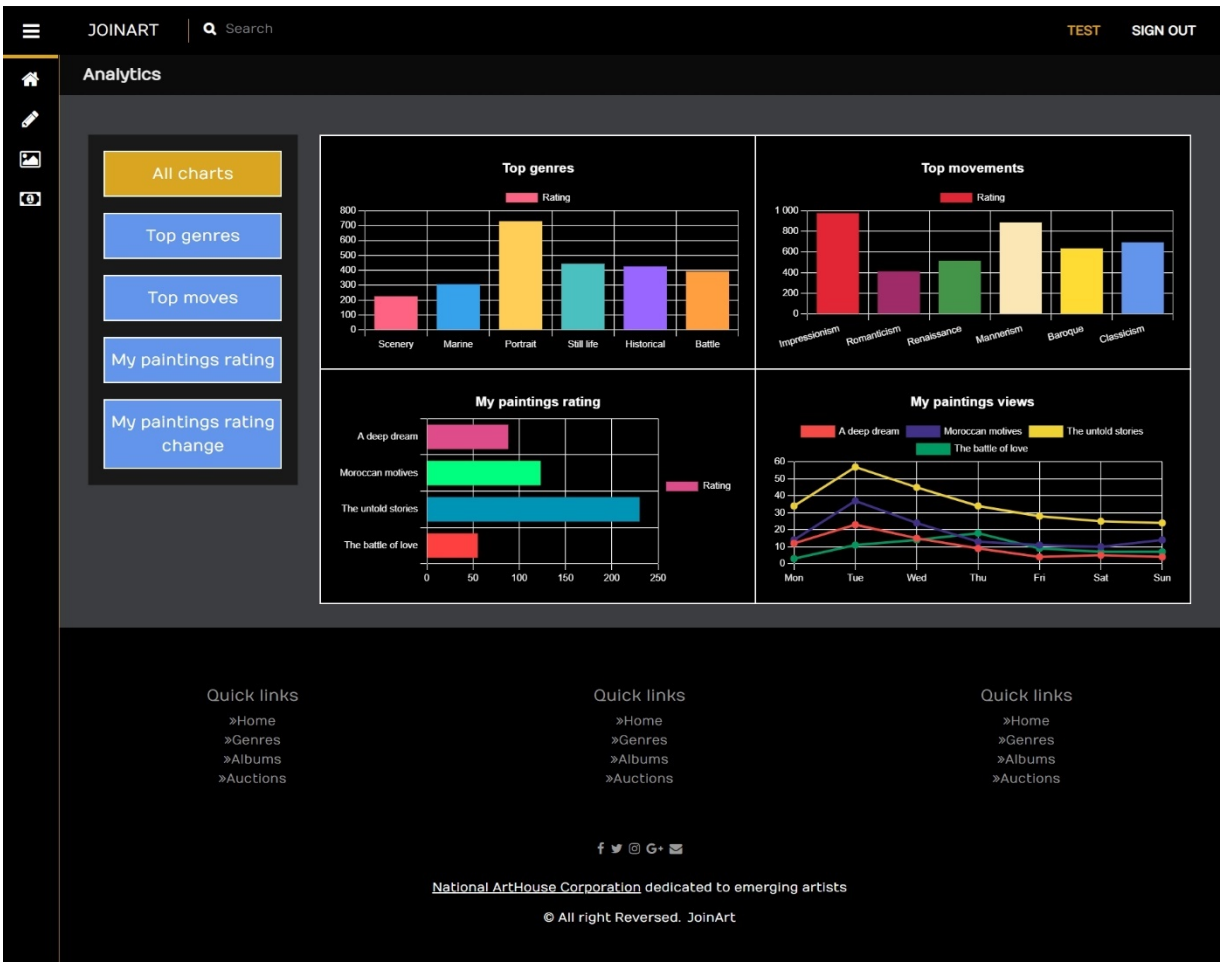

Рисунок 7 – Сторінка аналітики

Адміністратор в свою чергу має ще більше прав, ніж авторизований користувач, він має доступ до панелі адміністратора, де може здійснювати управління контентом інформаційної системи, а саме: додавати, редагувати та видаляти будь-які картини, жанри та напрями, аукціони, альбоми та інших користувачів. Перейшовши на адміністративну панель, адміністратор має змогу обрати тип даних, які він має на меті змінити, а саме: йому доступне: управління картинами, управління жанрами та напрямами, управління аукціонами, управління альбомами та управління користувачами. Перейшовши на будь-яку з цих сторінок, адміністратор має змогу переглянути у вигляді таблиці дані, додати новий об’єкт до інформаційної системи, редагувати вже існуючі або видаляти їх. Приклад адміністративної панелі, де адміністратор може здійснювати управління жанрами та напрямами, зображено у додатку П.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було виконано загальний огляд інформаційних систем та методів їх розробки, а також розроблена спеціалізована інформаційна система для художників «JoinArt», яка полегшує процес створення картин, надаючи художникам статистичну інформацію про популярність картин та жанрів, а також допомагає художникам продавати картини та збирати про них відгуки. Крім того, звичайні користувачі можуть придбати картини, які знаходяться в інформаційній системі, тим самим підтримавши художників, що їм до вподоби.

У процесі розробки інформаційної системи були пройдені всі етапи розробки, а саме: на першому етапі були сформульовані та детально проаналізовані вимоги до майбутньої інформаційної системи на основі вимог до схожих інформаційних систем та власного досвіду, щоб створений програмний продукт передбачав увесь основний функціонал, необхідний для специфіки роботи. На наступному етапі була розроблена архітектура системи, було прийнято рішення створювати додаток на базі клієнт-серверної архітектури, яка дозволяє найбільш продуктивно використовувати дану систему. Під час третього та четвертого етапу були обрані основні апаратні та програмні засоби для розробки інформаційної системи і з їхньою допомогою був розроблений додаток інформаційної системи «JoinArt». Для збереження та підтримання даних в актуальному вигляді, необхідних для функціонування інформаційної системи, було використано систему керування базами даних PostgreSQL та хмарне сховище Azure Blob Service, серверна частина, в якій зосереджена вся бізнес-логіка програми, розроблена за допомогою механізмів фреймворку ASP.NET Core та мови програмування C#, клієнтська частина була втілена за допомогою JavaScript-фреймворку React. І на останньому етапі робота системи була протестована за допомогою автоматичних unit-тестів та інтеграційних тестів, а також власного досвіду використання інформаційної системи.

Отримані на етапах розробки дані та використання зазначених засобів розробки інформаційної системи на практиці поглибило та розширило розуміння

викладеного у даній роботі теоретичного матеріалу, сприяло опануванню нових програмних технологій та покращенню навичок роботи з уже знайомими. Оскільки інформаційні системи наразі є основним функціональним рішенням проблеми пошуку, обробки, збереження та багатьох інших способів взаємодії з даними, розроблена інформаційна система та набуті в процесі розробки знання є актуальними та корисними.

Розроблена інформаційна система може бути розміщена в мережі Інтернет приватною особою або організацією для комерційного використання в наявному або вдосконаленому вигляді. Наявний в інформаційній системі аналітичний функціонал можна зробити доступним за символічну плату, а також брати деякий відсоток від продажу кожної картини, що була продана за допомогою інформаційної системи.

У перспективі функціональні можливості розробленого програмного продукту можуть бути розширені шляхом, наприклад, доповнення існуючих аналітичних можливостей системи, що надавало б користувачам ще більше інформації про популярність картин та в якому напрямку слід рухатися, а також розширити спеціалізацію на інші сфери, наприклад, дизайнерську. Також має потенціал ідея інтеграції інформаційної системи з технологією рекомендаційних систем, які б аналізували вподобання користувачів за деякими критеріями та відповідно до отриманих даних формували домашню сторінку або стрічку рекомендацій. Також було б доцільно створити мобільний додаток для більш приємного користування інформаційною системою за допомогою мобільного телефону.

Отже, наприкінці виконання кваліфікаційної роботи можна зазначити, що поставлена мета проекту – досягнута, поставлені задачі – виконані, а практична частина – реалізована. Розроблена інформаційна система «JoinArt» відповідає всім необхідним вимогам та має потенціал до розвитку та розширення.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Жданов С. А. Информационные системы / С. А. Жданов., 2015. – 13 с.
2. What Does Information System (IS) Mean? [Електронний ресурс] – Режим доступу до ресурсу: https://www.techopedia.com/definition/24142/information-system-is.
3. What Are Information Systems? - Definition & Types [Електронний ресурс] – Режим доступу до ресурсу: https://study.com/academy/lesson/what-are-information-systems-definition-types-quiz.html.
4. Stair R. Principles of Information Systems 7th edition / R. Stair, G. Reynolds., 2006. – 87 с.
5. Information Systems and Strategy Course [Електронний ресурс] // Euromed Marseille School of Management, World Med MBA Program – Режим доступу до ресурсу: http://www.chris-kimble.com/Courses/World_Med_MBA/Types-of-Information-System.html.
6. Schwalbe K. Information Technology Project Management 7th Edition / Kathy Schwalbe., 2013. – 145 с.
7. Information System [Електронний ресурс] // Fairleigh Dickinson University – Режим доступу до ресурсу: https://www.britannica.com/topic/information-system.
8. Concept of User and System Requirements [Електронний ресурс] // Collegenote – Режим доступу до ресурсу: https://collegenote.pythonanywhere.com/curriculum/software-engineering-csit/54/309/.
9. Functional requirements [Електронний ресурс] // Collegenote – Режим доступу до ресурсу: https://collegenote.pythonanywhere.com/curriculum/software-engineering-csit/54/311/.
10. Non-functional requirements [Електронний ресурс] // Collegenote – Режим доступу до ресурсу: https://collegenote.pythonanywhere.com/curriculum/software-engineering-csit/54/312/.

11. Client-Server Architecture [Електронний ресурс] // ScienceDirect – Режим доступу до ресурсу: https://www.sciencedirect.com/topics/computer-science/client-server-architecture.
12. PostgreSQL About [Електронний ресурс] // PostgreSQL – Режим доступу до ресурсу: https://www.postgresql.org/about/.
13. Azure Blob Storage [Електронний ресурс] // Microsoft Azure – Режим доступу до ресурсу: https://azure.microsoft.com/en-us/services/storage/blobs/.
14. Introduction to ASP.NET Core [Електронний ресурс] // Microsoft – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0.
15. Welcome to the Visual Studio IDE [Електронний ресурс] // Microsoft – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2019.
16. React – JavaScript-библиотека для создания пользовательских интерфейсов [Електронний ресурс] // React – Режим доступу до ресурсу: https://ru.reactjs.org/.
17. Meet WebStorm [Електронний ресурс] // JetBrains – Режим доступу до ресурсу: https://www.jetbrains.com/help/webstorm/meet-webstorm.html.

ДОДАТОК А

Процес звернення користувача до веб-сервісу з метою поставити лайк картині

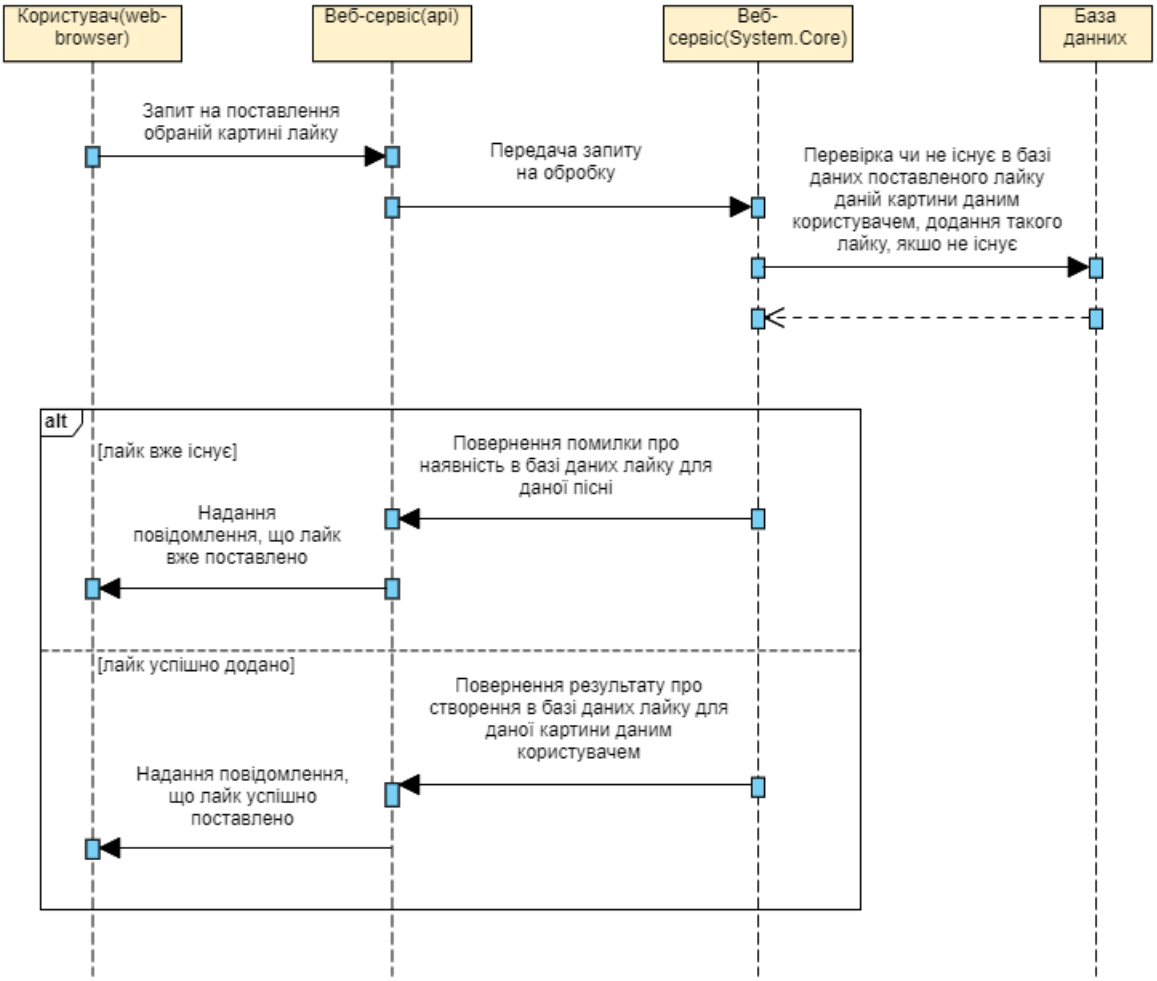

Рисунок А.1 – Діаграма послідовностей, на якій зображено процес додання лайку картині

ДОДАТОК Б

Процес звернення адміністратора до веб-сервісу з метою створити новий жанр картин

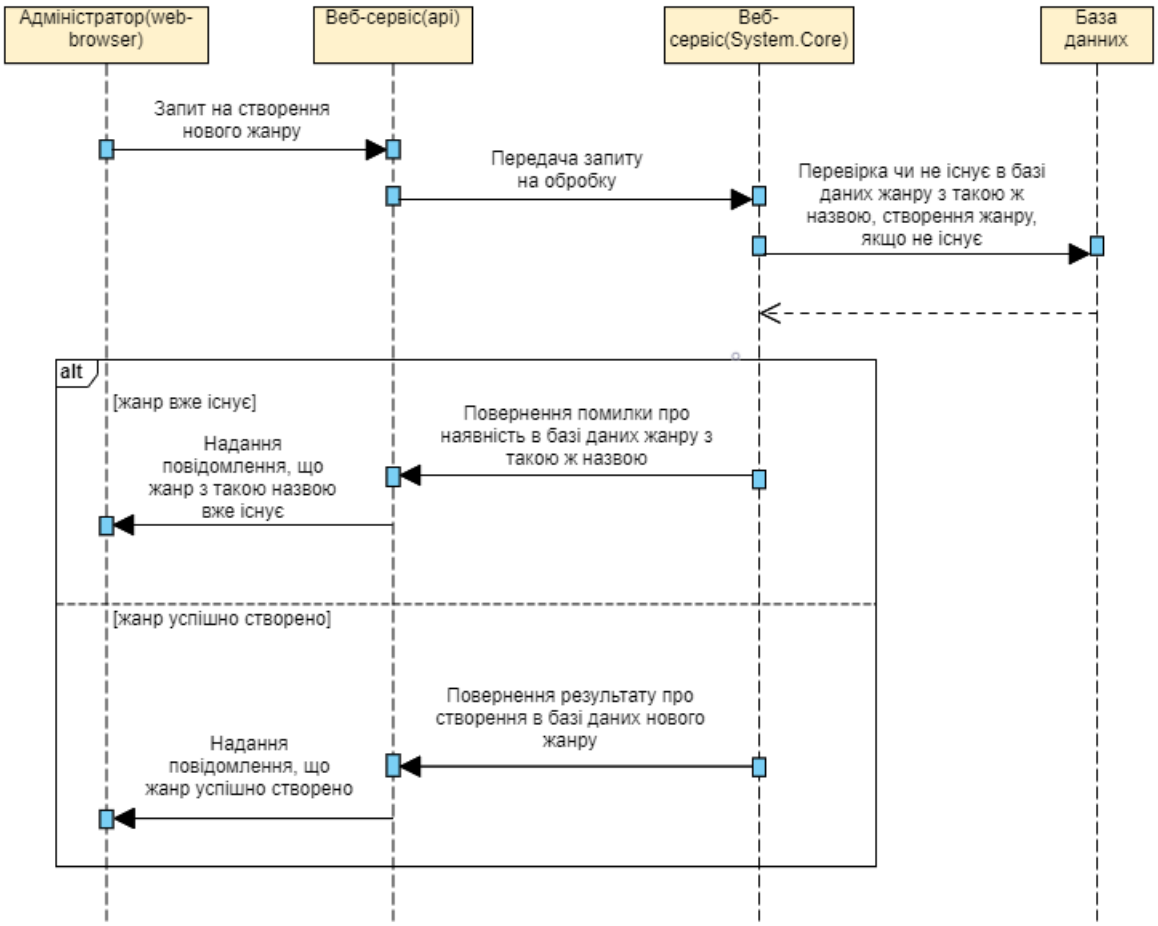

Рисунок Б.1 – Діаграма послідовностей, на якій зображено процес створення нового жанру

ДОДАТОК В

ER-модель бази даних інформаційної системи «JoinArt»

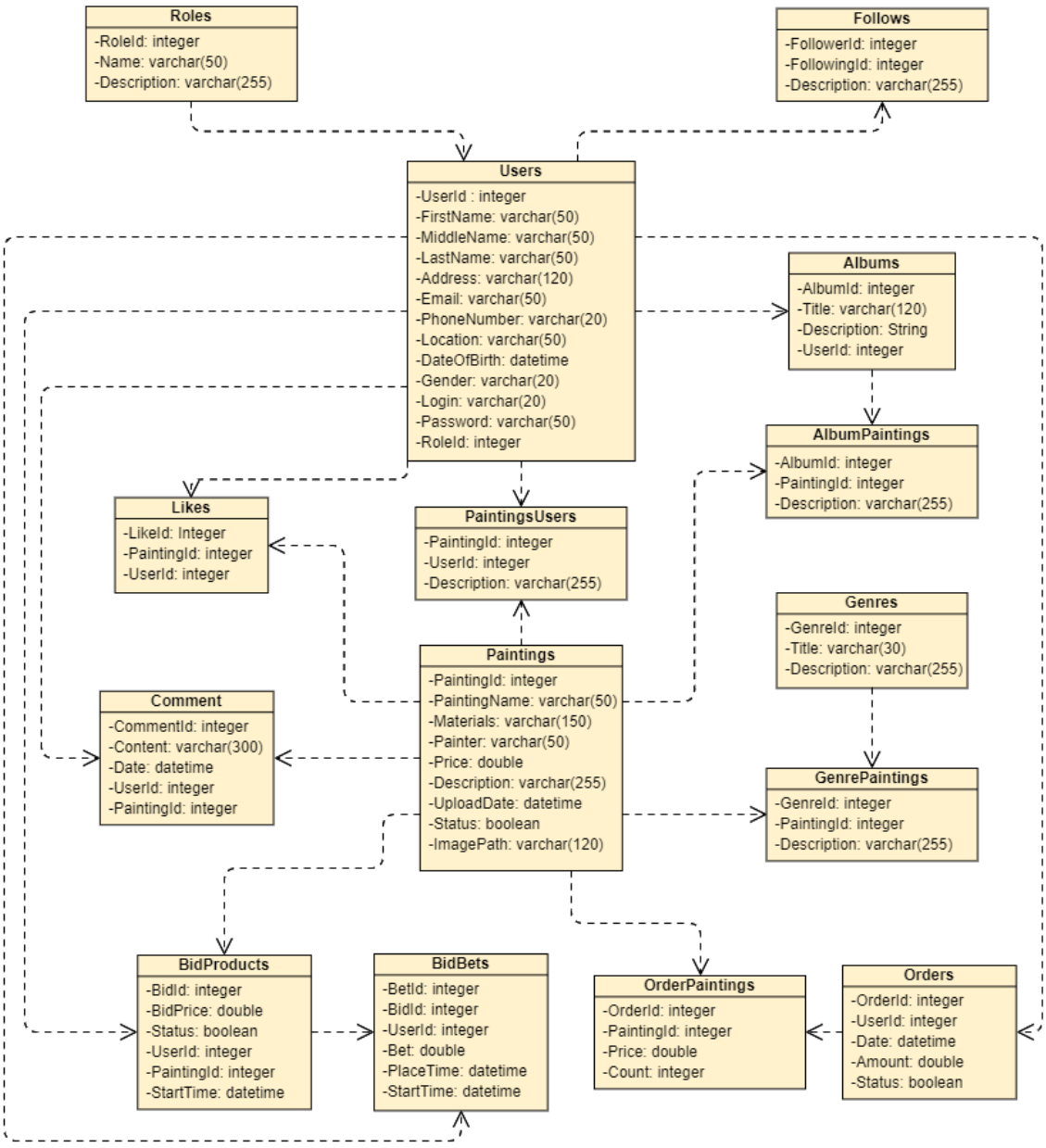

Рисунок В.1 – ER-модель бази даних інформаційної системи «JoinArt»

ДОДАТОК Г

Моделювання управління системи «JoinArt»

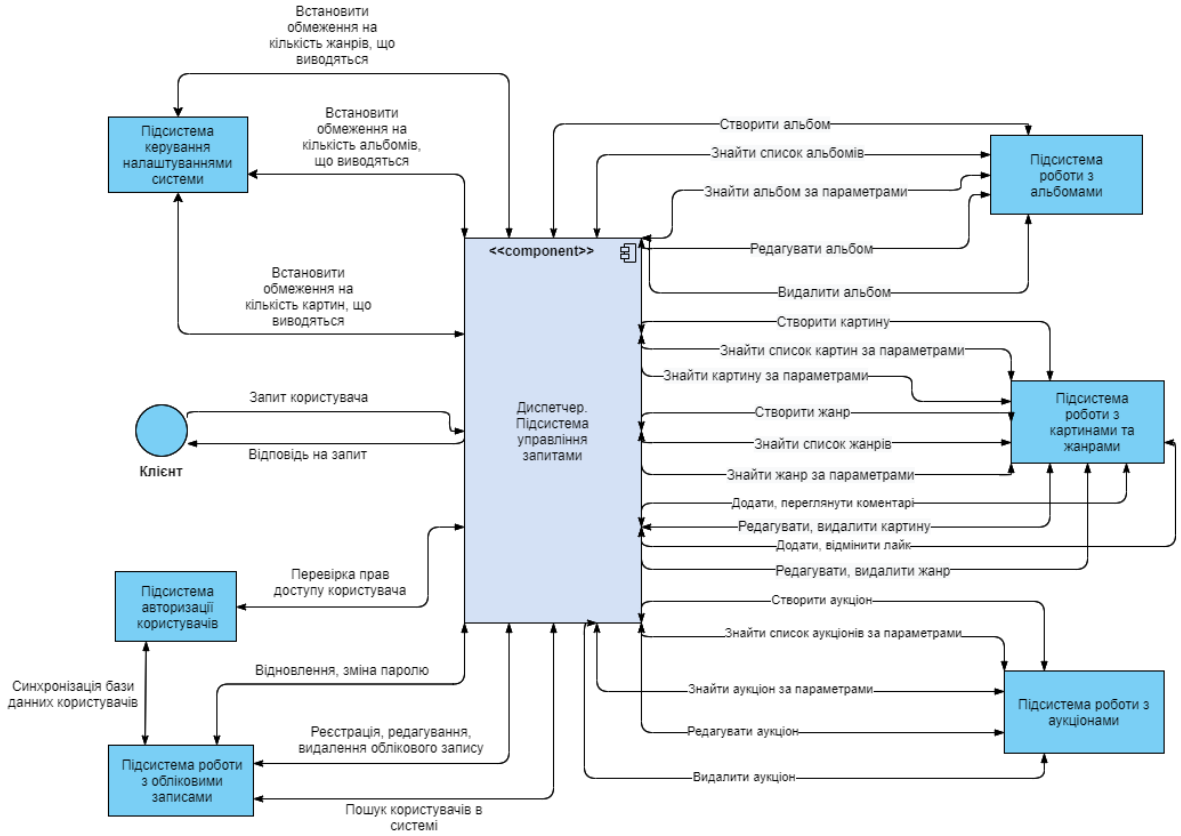

Рисунок Г.1 – Діаграма компонентів, на якій зображено моделювання управління системи «JoinArt»

ДОДАТОК Ґ

Модульна декомпозиція системи «JoinArt»

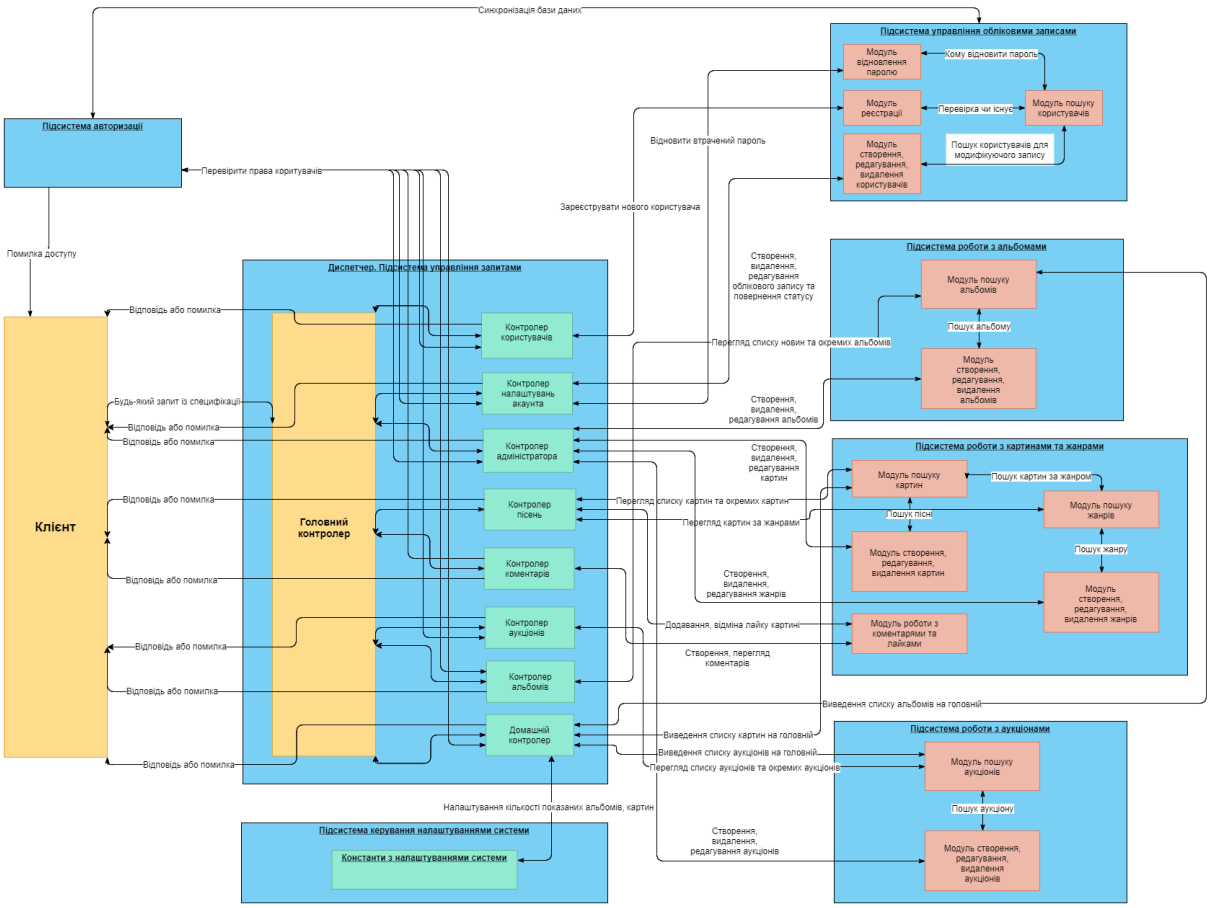

Рисунок Ґ.1 – Діаграма компонентів, на якій зображено модульну декомпозицію системи «JoinArt»

ДОДАТОК Д

Приклад використання стилів із фреймворку Bootstrap

```
<div className="row footer-mrg-0">
    <div className="col-xs-12 col-sm-12 col-md-12 mt-2 mt-sm-5 textcol">
        <ul className="list-unstyled list-inline social text-center">
            <li className="list-inline-item">
                <a href="https://www.facebook.com/joinart">
                    <i className="fa fa-facebook" />
                </a>
            </li>
            <li className="list-inline-item">
                <a href="https://www.twitter.com/joinart">
                    <i className="fa fa-twitter" />
                </a>
            </li>
            <li className="list-inline-item">
                <a href="https://www.instagram.com/joinart">
                    <i className="fa fa-instagram" />
                </a>
            </li>
            <li className="list-inline-item">
                <a href="https://www.google.com/joinart">
                    <i className="fa fa-google-plus" />
                </a>
            </li>
        </ul>
    </div>
    <hr/>
</div>
```

Лістинг Д.1 – Використання стилів із CSS-фреймворку Bootstrap

ДОДАТОК Е

Сторінки та їх роути, які доступні у розробленому додатку

```
<Switch>
    <Route exact path="/" component={Home} />
    <Route exact path="/genres" component={Genres} />
    <Route exact path="/login" component={Login} />
    <Route exact path="/signup" component={SignUp} />
    <Route exact path="/painting/:id" component={PaintingPage} />
    <Route exact path="/genre/:id" component={GenrePage} />
    <Route exact path="/albums" component={Albums} />
    <Route exact path="/album/:id" component={AlbumPage} />
    <Route exact path="/search" component={SearchPage} />
    <Route exact path="/auctions" component={Auctions} />
    <Route exact path="/auction/:id" component={AuctionPage} />
    <Route exact path="/user" component={UserPage} />
    <Route exact path="/user/manage-paintings" component={ManagePaintings} />
    <Route exact path="/user/purchases" component={Purchases} />
    <Route exact path="/user-paintings/:userId" component={UserPaintingsPage} />
    <Route exact path="/user/analytics" component={Analytics} />
    <Route exact path="/admin" component={Admin} />
</Switch>
```

Лістинг Е.1 – Використання JavaScript-бібліотеки React Router для надання кожній сторінці свого роуту

ДОДАТОК Є

Головна сторінка інформаційної системи «JoinArt»

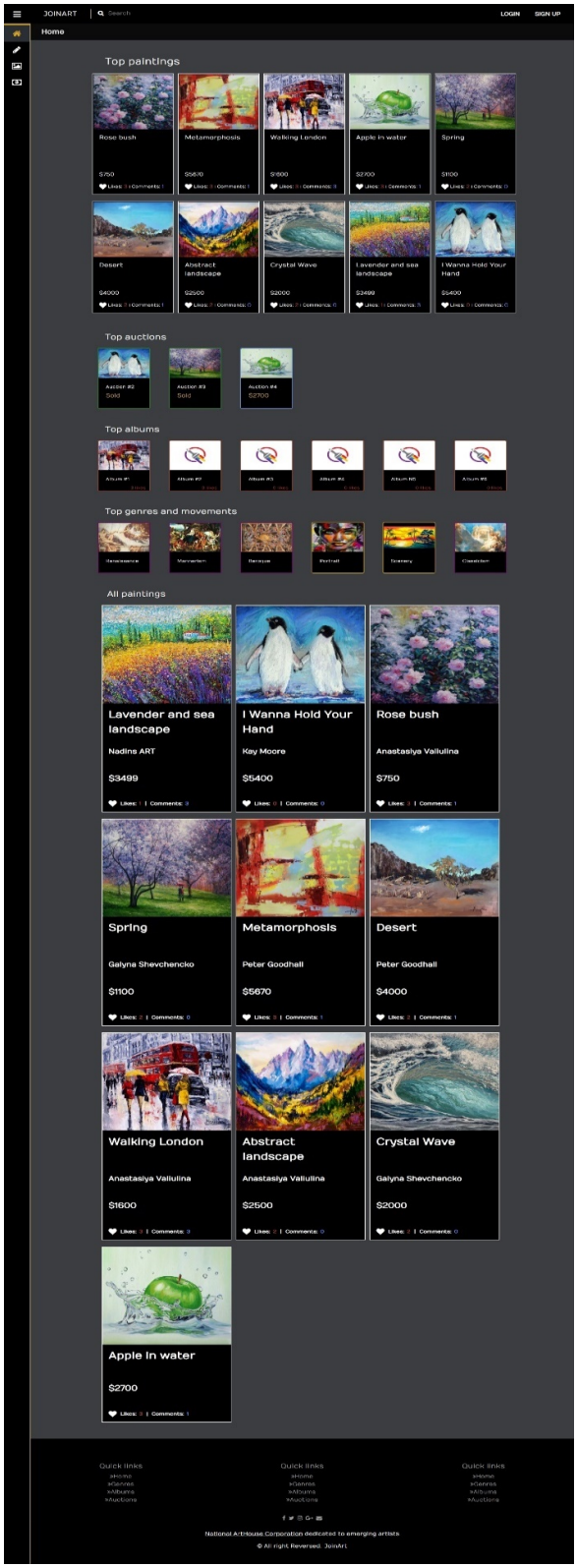

Рисунок Є.1 – Скріншот головної сторінки інформаційної системи «JoinArt»

ДОДАТОК Ж

Сторінка жанрів

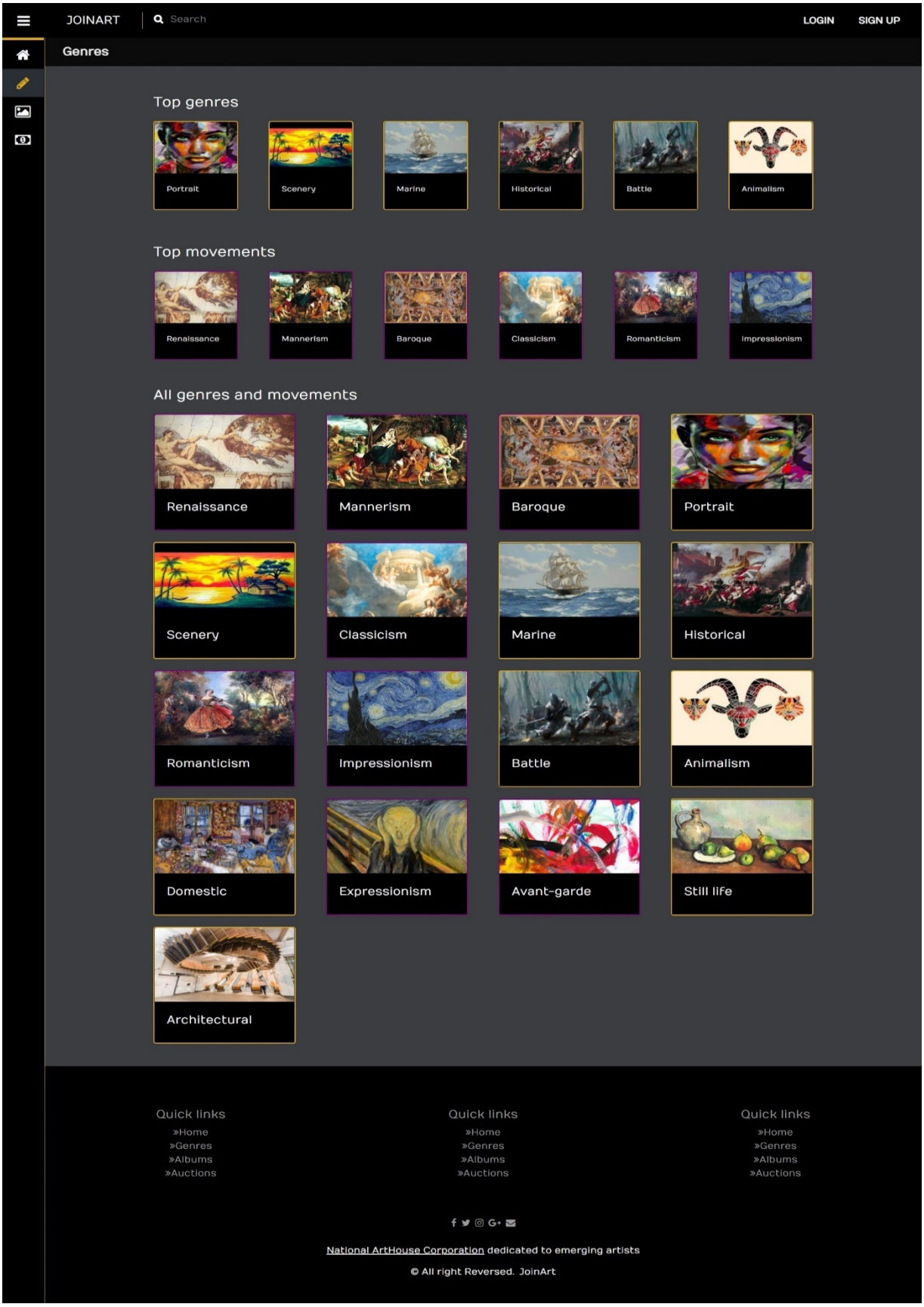

Рисунок Ж.1 – Скріншот сторінки жанрів інформаційної системи «JoinArt»

ДОДАТОК З

Сторінка жанру

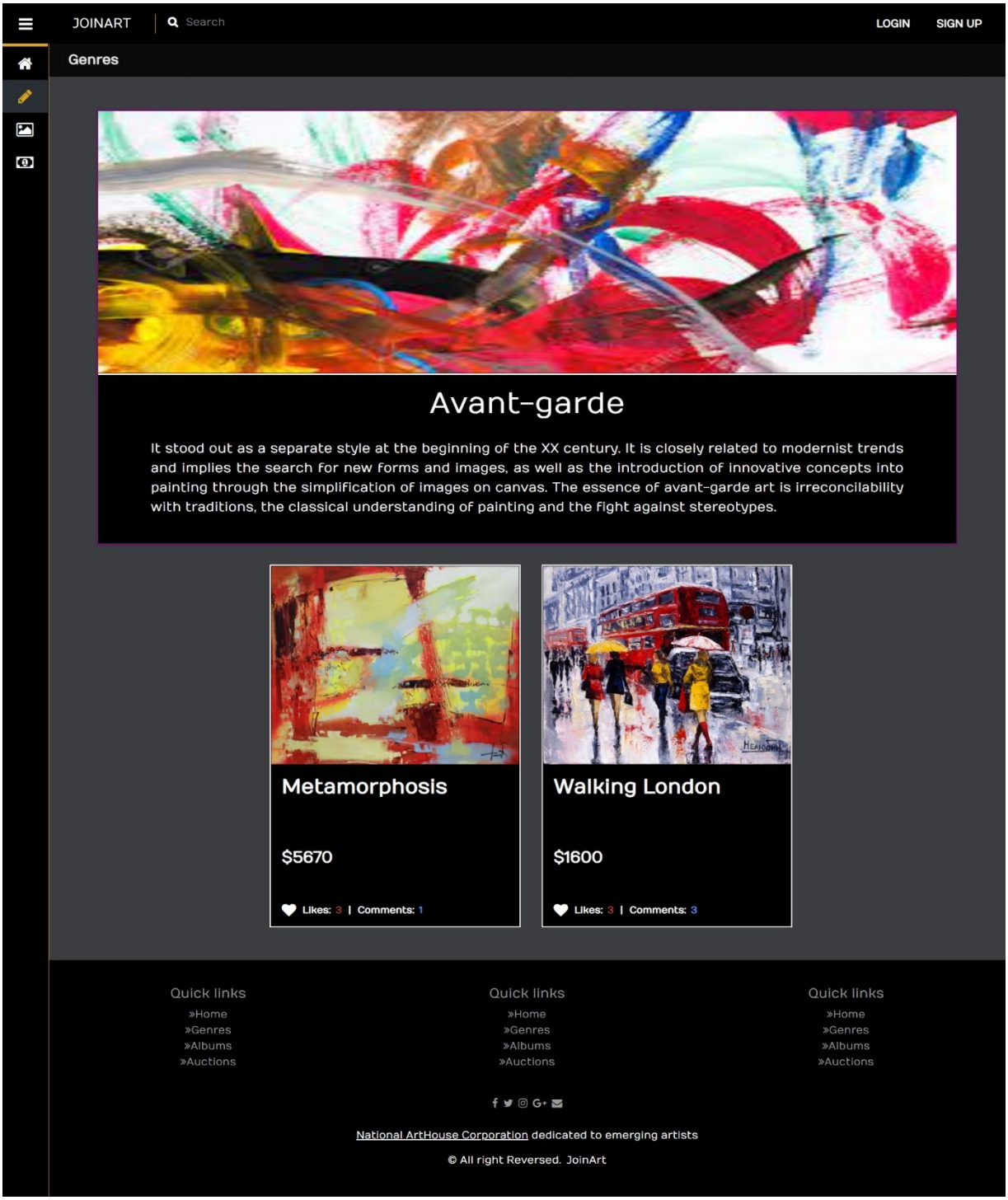

Рисунок 3.1 – Скріншот сторінки жанру з назвою «Avant-garde» інформаційної системи «JoinArt»

ДОДАТОК И

Сторінка результатів пошуку

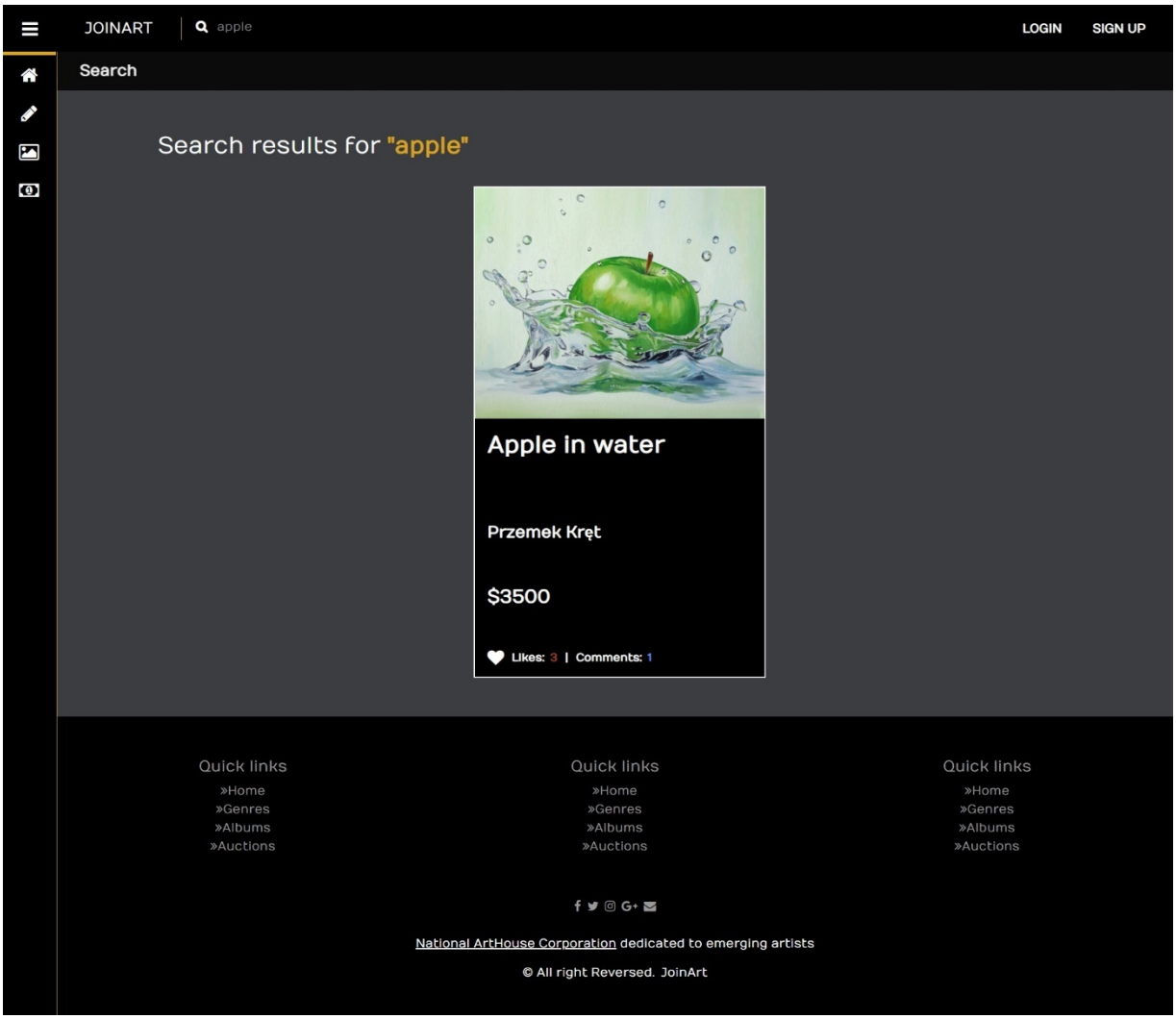

Рисунок И.1 – Скріншот сторінки результатів пошуку за ключовим словом «apple» інформаційної системи «JoinArt»

ДОДАТОК Ї

Сторінка картини

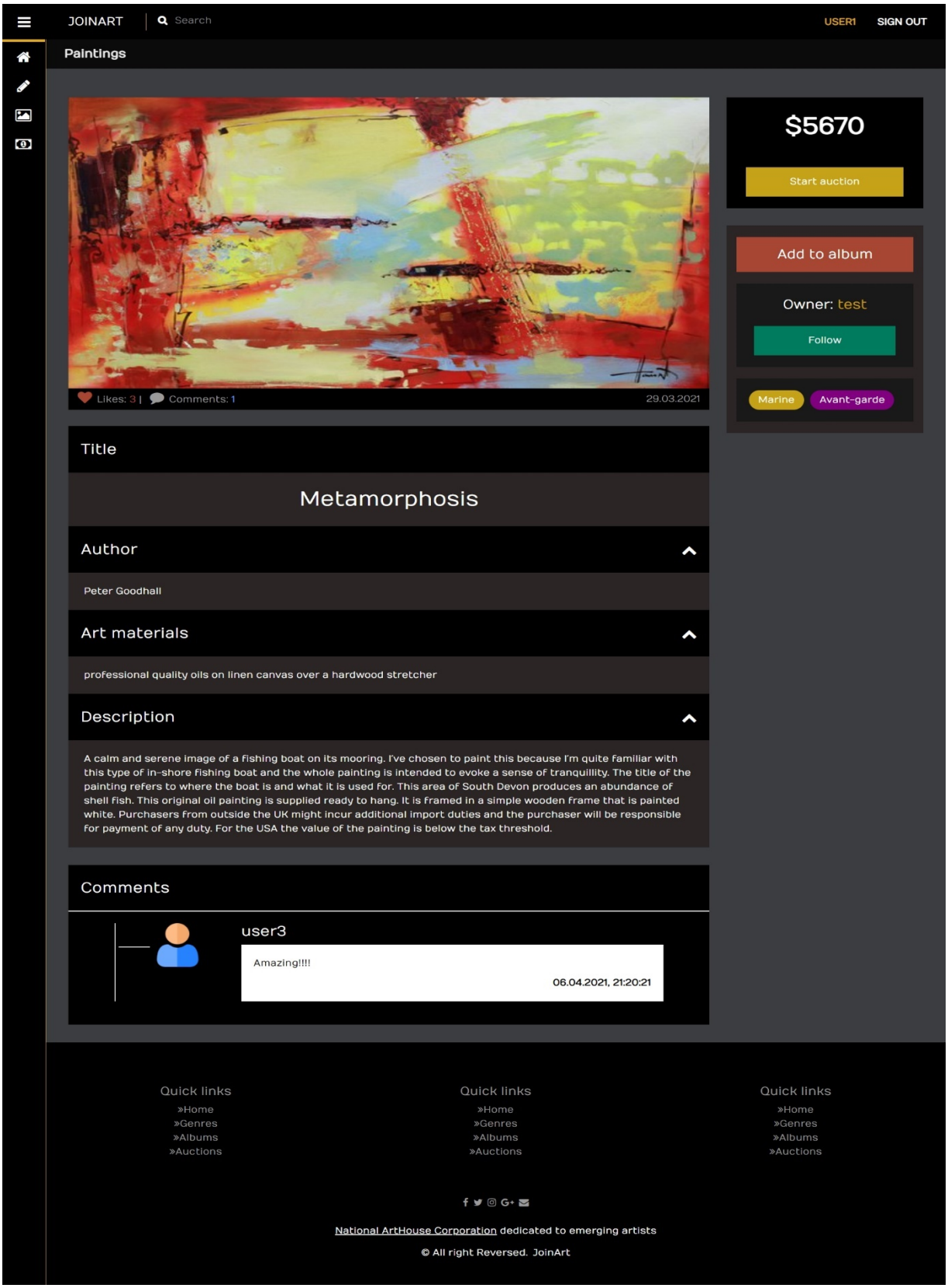

Рисунок Ї.1 – Скріншот сторінки картини з назвою «Metamorphosis» інформаційної системи «JoinArt»

ДОДАТОК Й

Сторінка аукціонів

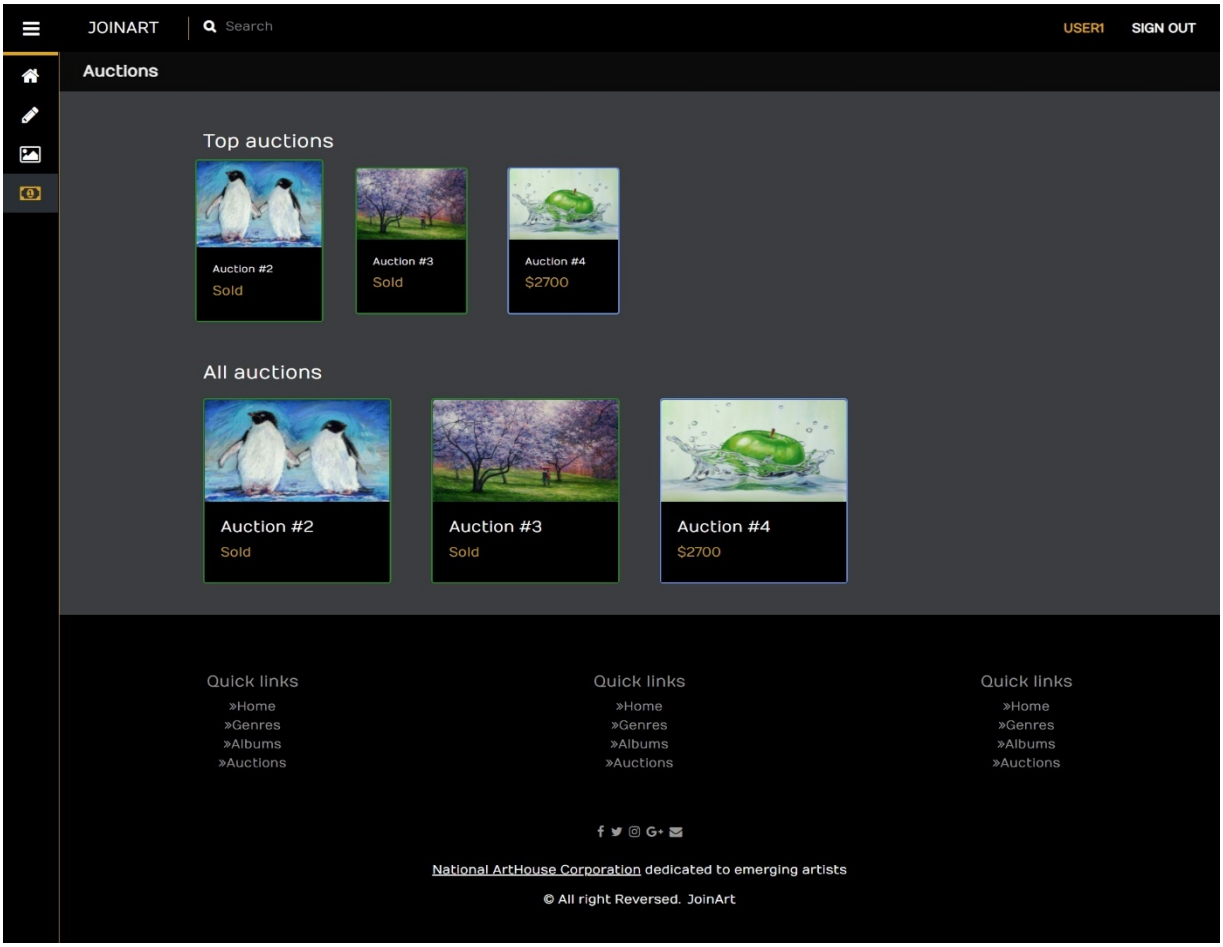

Рисунок Й.1 – Скріншот сторінки аукціонів інформаційної системи «JoinArt»

ДОДАТОК К

Сторінка аукціону

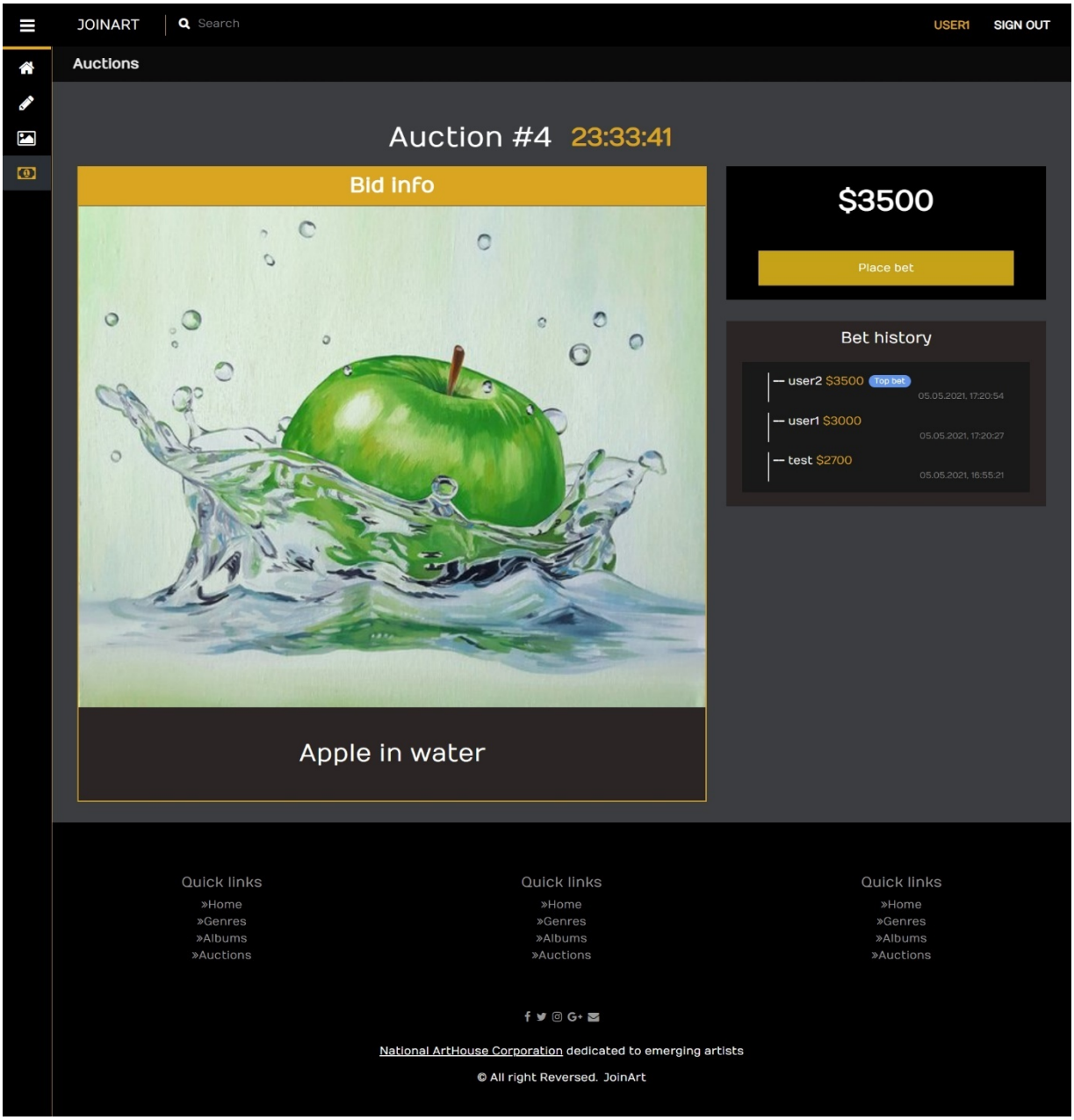

Рисунок К.1 – Скріншот сторінки аукціону, в якому розігрується картина з назвою «Apple in water»

ДОДАТОК Л

Сторінка кабінету користувача

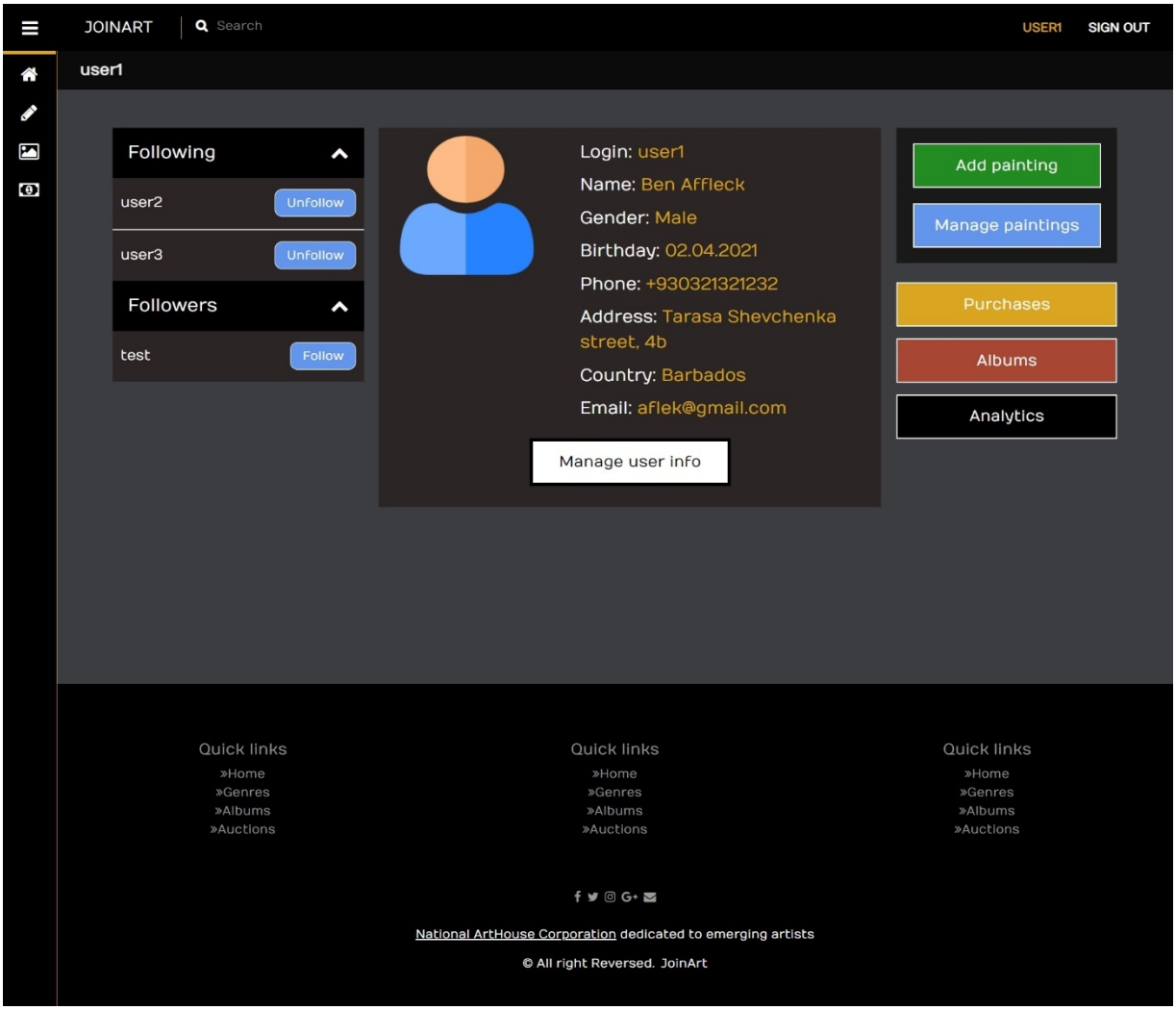

Рисунок Л.1 – Скріншот сторінки кабінету користувача інформаційної системи «JoinArt»

ДОДАТОК М

Форма для додавання картини

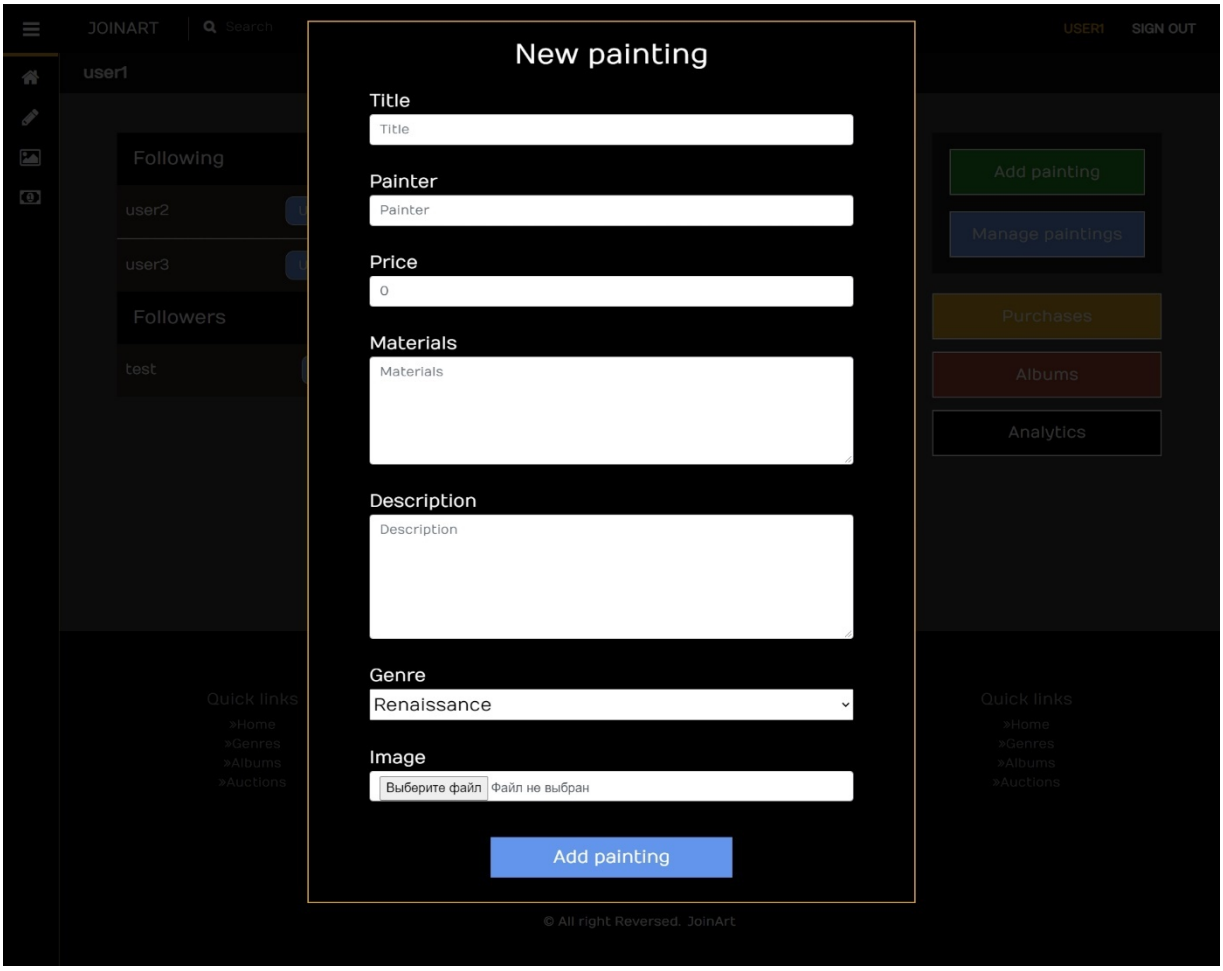

Рисунок М.1 – Скріншот форми для додавання нової картини до інформаційної системи «JoinArt»

ДОДАТОК Н

Сторінка управління картинами

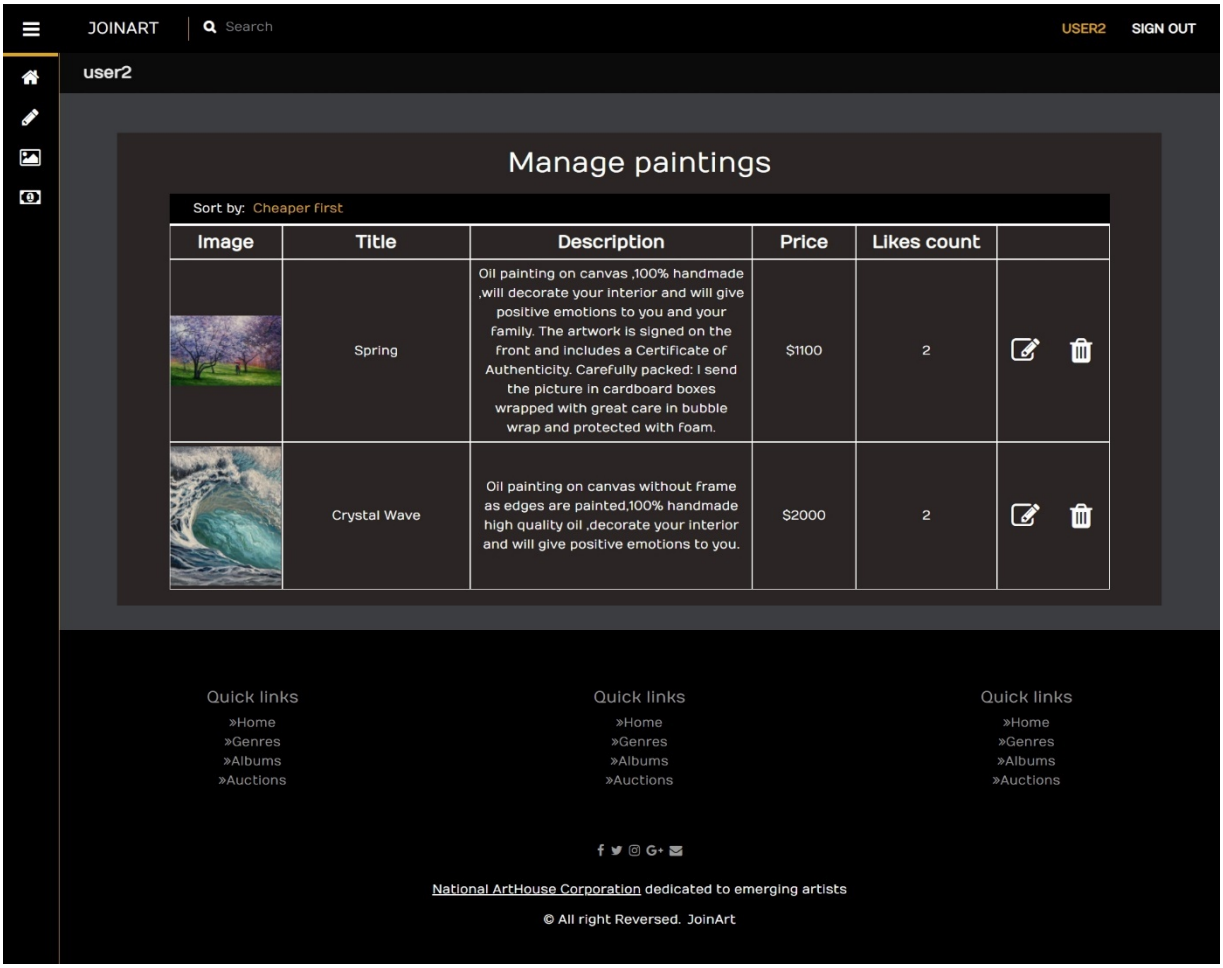

Рисунок Н.1 – Скріншот сторінки управління картинами користувача інформаційної системи «JoinArt»

ДОДАТОК О

Сторінка покупок

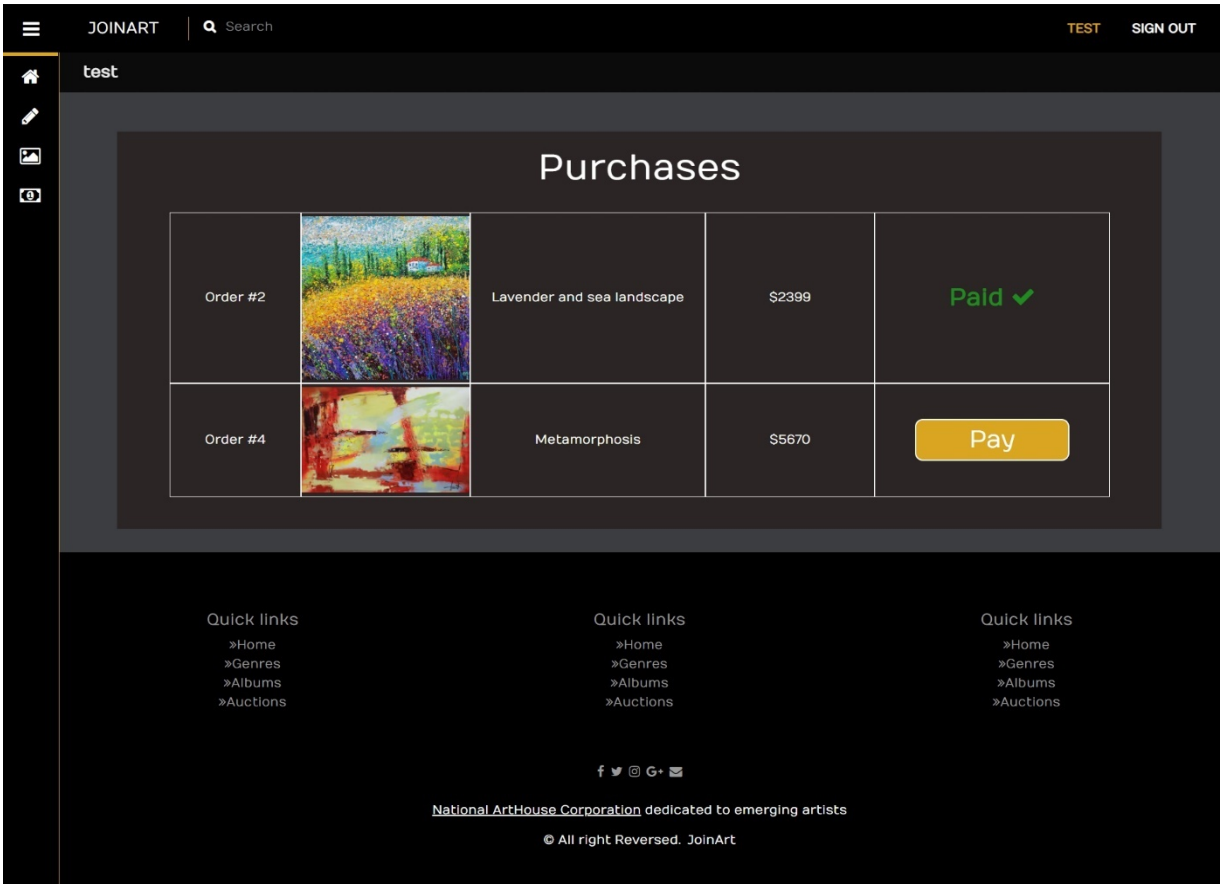

Рисунок О.1 – Скріншот сторінки покупок користувача інформаційної системи «JoinArt»

ДОДАТОК П

Панель адміністратора

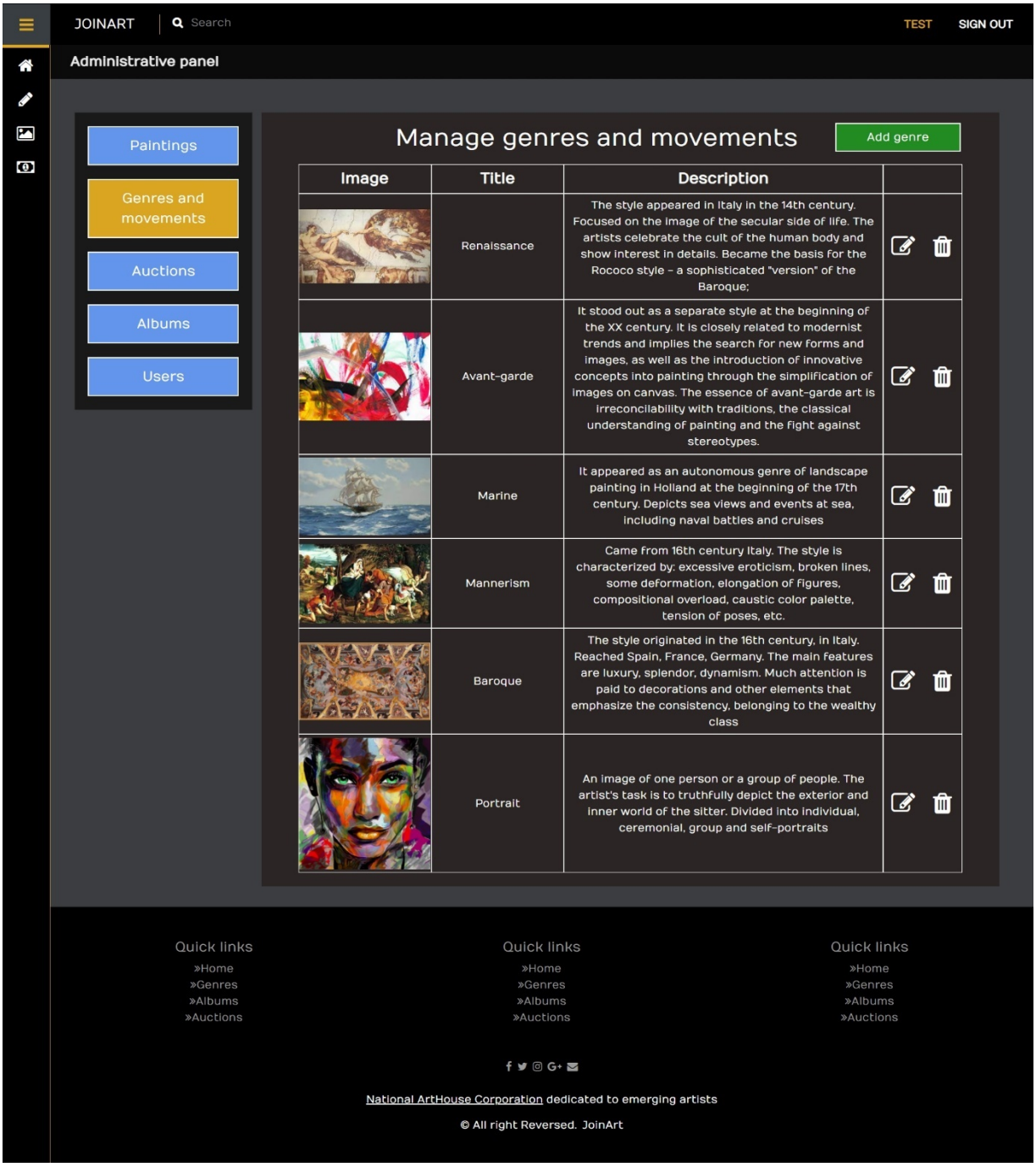

Рисунок П.1 – Скріншот панелі адміністратора інформаційної системи «JoinArt»