**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Національний університет кораблебудування імені адмірала Макарова**

Навчально-науковий інститут комп'ютерних наук та управління проєктами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_____ проф. Приходько С.Б.

«___» _____ 2022 р.

***КВАЛІФІКАЦІЙНА РОБОТА***

**на здобуття ступеня вищої освіти «бакалавр»**

**на тему**: Розробка програмного забезпечення вебсайту для онлайн-галереї "RiArt"

Виконав: студент групи 4151

Рилов В.О.

(підпис, ПІБ)

Керівник роботи:

Викладач, к.т.н.

(посада, науковий ступень вчене звання)

Пухалевич А.В.

(підпис, ПІБ)

Миколаїв – 2022 р.

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**Національний університет кораблебудування імені адмірала Макарова**

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«___» _____ 2022 р.

***ЗАВДАННЯ***
***НА КВАЛІФІКАЦІЙНУ РОБОТУ***
**на здобуття ступеня вищої освіти «бакалавр»**

Студенту Рилову Володимиру Олександровичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: розробка програмного забезпечення вебсайту для онлайн-галереї "RiArt"

Керівник роботи викладач, к.т.н Пухалевич Андрій Володимирович

Затверджені наказом ректора №256-уч від «11» травня 2022 р.

2. Термін подання роботи: 10.06.2022 р.

3. Вихідні дані по роботі: _____

4. Перелік питань, що належать до розробки (найменування розділів) Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Опис предметної галузі та постановка задачі; Проєкт програмного забезпечення (ескізний, технічний та робочий); Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів 1. Титульний аркуш; 2. Мета та актуальність розробки; 3. Способи розробки програмного забезпечення вебсайтів; 4. Діаграма варіантів використання; 5. Діаграма класів; 6. Логічна модель даних; 7. Вибір мови програмування та СКБД; 8. Фізична модель даних; 9. Діаграма розміщення; 10. Головна сторінка; 11. Вебсторінка публікації; 12. Вебсторінка для публікації робіт; 13. Вебсторінка профілю користувача; 14. Вебсторінка редагування профілю користувача; 15. Висновки.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 4 | Гурець Н.В., ст. викладач | | |
| | | | |
| | | | |
| | | | |
| | | | |

7. Дата видачі завдання 11.05.2022 р. _____

**К А Л Е Н Д А Р Н И Й  П Л А Н**

| Номер | Назва етапів роботи | Терміни виконання | Примітка |
|---|---|---|---|
| 1. | Підготовка розділу Вступ | 24.03.2022 | ПП* |
| 2. | Опис та аналіз предметної галузі | 28.03.2022 | ПП* |
| 3. | Постановка задачі | 31.03.2022 | ПП* |
| 4. | Ескізний проєкт програмного забезпечення | 27.04.2022 | |
| 5. | Технічний проєкт програмного забезпечення | 06.05.2022 | |
| 6. | Робочий проєкт програмного забезпечення | 13.05.2022 | |
| 7. | Підготовка розділу Результати розробки програмного забезпечення | 17.05.2022 | |
| 8. | Підготовка розділу з охорони праці | 18.05.2022 | ПП* |
| 9. | Підготовка розділу Висновки | 19.05.2022 | |
| 10. | Оформлення списку використаних джерел та додатків | 20.05.2022 | |
| 11. | Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.) | 10.06.2022 | не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку) |
| 12. | Підготовка презентації та доповіді | 12.06.2022 | |
| 13. | Попередній захист роботи на засіданні кафедри ПЗАС | 14.06.2022 | |
| 14. | Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді | 24.06.2022 | |

* - за результатами переддипломної практики (ПП), яка була з 21.03.2022 до 08.05.2022 р.

**Студент** _____ Рилов В.О.
(підпис) (ПІБ)

**Керівник роботи** _____ Пухалевич А.В.
(підпис) (ПІБ)

# АНОТАЦІЯ

Кваліфікаційна робота присвячена вирішенню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці програмного забезпечення вебсайту для онлайн-галереї "RiArt".

Об’єктом розробки є програмне забезпечення вебсайту для онлайн-галереї "RiArt".

Зроблено аналіз предметної галузі онлайн-галерей та проаналізовано існуючі програмні продукти та методи розробки вебсайтів. Було розроблено проєкт програмного забезпечення, а також реалізовано програмне забезпечення для вебсайту онлайн-галереї та представлено частину коду.

Програмне забезпечення для вебсайту було виконано за допомогою середовища розробки Microsoft Visual Studio, мови програмування C# й JavaScipt, та СУБД Microsoft SQL Server.

Кваліфікаційна робота викладена на 118 сторінках друкованого тексту, містить 62 рисунки, 11 таблиць, список використаних джерел із 7 найменувань та 4 додатки.

Кваліфікаційна робота виконана українською мовою.

# ABSTRACT

Qualification work is devoted to solving the practical problem of software engineering, characterized by complexity and uncertainty of conditions, namely the website software development for the online gallery "RiArt".

The object of the development is the website software development for the online gallery "RiArt".

The analysis of the subject area of online galleries was made and the existing software products and methods of website development was analyzed. A software project was developed, as well as software for the online gallery website and part of the code was presented.

The software for the website was developed using the Microsoft Visual Studio development environment, the C # and JavaScipt programming languages, and the Microsoft SQL Server DBMS.

Qualifying work is presented on 118 pages of printed text, contains 62 figures, 11 tables, list of used sources of 7 items and 4 appendices.

Qualifying work is performed in Ukrainian.

ЗМІСТ

# ВСТУП

Фахівців з дизайну у сучасному світі з останнім часом стає все більше. Ця професія одна з найпопулярніших й одна з найбільш затребуваних в даний період.

Багато хто із сучасних спеціалістів створює електронні малюнки, моделі, тощо. У такій формі їх роботи завжди захищенні від фізичного впливу, їх набагато легше зберігати у великих кількостях. Це спрощує можливість продажу та показу робіт: покупцям, компаніям, виставкам і т.д.

Часто дизайнери використовують так звані репозиторії, де дизайнери можуть розміщувати своє резюме. Більшість із таких репозиторіїв не передбачають можливості розміщення додаткової інформації про дизайнера зручним для перегляду способом.

Цю проблему вирішують онлайн-галереї. Онлайн-галерея – це список з безліччю електронних картинок. Цілі онлайн-галерей – розповсюдження інформації про художників (дизайнерів).

Першими з онлайн-галерей були віртуальні музеї. Вони з'явилися практично одночасно з масовим поширенням Інтернету. Спершу цей тип вебсайтів був оптимізований для експозиції музейних чи художніх матеріалів на них.

За допомогою галерей відвідувачі та дизайнери об’єднуються у свого роду спільноти. Але головною перевагою онлайн-галереї є те, що матеріали, представлені в галереї, дозволяють судити про особливості творчого життя дизайнерів різних напрямків.

Вже існують аналоги ПЗ для вирішення частини цих задач і процесів. В кожного з них є свої певні недоліки. Основні з них - це те, що на існуючих онлайн-галереях не розміщуються цифрові картини чи комп’ютерні моделі. Також, більшість аналогів не мають функціоналу заповнення та показу резюме, біографії й іншої інформації про дизайнера.

Метою кваліфікаційної роботи є вирішення практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробка програмного забезпечення вебсайту для онлайн-галереї "RiArt".

Потрібно вирішити наступні завдання, для того, щоб досягнути поставленої мети:

1) провести аналіз предметної області онлайн галерей;
2) провести аналіз існуючих аналогів вебсайтів онлайн-галерей;
3) провести аналіз методів розробки вебсайтів;
4) створити проєкт програмного забезпечення;
5) провести тестування програмного забезпечення;
6) ознайомитися з питаннями охорони праці при розробці ПЗ та роботі з комп’ютером.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ВЕБСАЙТІВ ОНЛАЙН-ГАЛЕРЕЙ. ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБСАЙТУ ДЛЯ ОНЛАЙН-ГАЛЕРЕЇ “RiArt”

## 1.1 Аналіз предметної області онлайн галерей

Проблемою дизайнерів є складність розповсюдження своїх художніх досягнень та робіт. Дизайнерам незручно розміщати на одному ресурсі своє резюме, а на іншому свої роботи, тому що результат розміщення впливає на швидкість подальшого працевлаштування та розвитку.

Кваліфікацію та навички дизайнера, найчастіше, перевіряють за його портфоліо, тому процес перевірки навичок зводиться до наступного порядку:

1. Перевірка резюме та біографії;
2. Пошук посилання на ресурс, де знаходиться (якщо є) портфоліо фахівця;
3. Перехід на ресурс з портфоліо;
4. Співвідношення навичок роботи спеціаліста вказаних в портфоліо з його реальними практичними роботами.

Можливість розмістити своє резюме та художні роботи у цифровому виді на сайтах вже не в новизну. Для цього використовуються спеціалізовані вебсайти – онлайн-галереї. Онлайн-галерея – це вебсайт де зберігаються та відображаються художні роботи у цифровій формі.

Для розв’язання проблем з розміщенням інформації про дизайнера, використовують додатковий платний функціонал в онлайн-галереях, наприклад, можливість розміщення біографії та резюме на сайті. Резюме – документ який містить інформацію про навички спеціалісту, його трудову діяльність та професійні досягнення.

Найбільш популярним на даний час стали сайти онлайн-галерей, що направлені на розміщення 3D графіки, ілюстрацій і інших варіантів напрямку Digital art, які зазвичай не мають оригінальних фізичних, не оцифрованих варіантів. Популярність зумовлена досить невеликим вибором подібних онлайн-галерей.

Digital art чи цифрове мистецтво – це напрямок в дизайні, який спрямований на створення художніх робіт у цифровій формі. Для цього використовуються різні комп'ютерно-інформаційні технології, а також спеціалізоване ПЗ. В деяких випадках до Digital art відносяться відскановані художні роботи, які були модифіковані за допомогою ПЗ. Цей напрям також охоплює ілюстрації та 3D графіку і є одним з найбільш популярних напрямів у світі.

3D графіка – це ілюстрації реалістичних моделей у трьох вимірному просторі. Моделі, в основному, створюються за допомогою спеціального ПЗ, та зберігаються у цифровому виді.

За даними вебсайту "similarweb", одна з таких Digital art онлайн-галерей, яка майже не має аналогів, "ArtStation", перебуває на 1402 місці за відвідуваністю користувачів у всьому світі (рисунок 1.1) [1].


Рисунок 1.1 – Глобальне положення онлайн-галереї "ArtStation"

Головною перевагою усіх онлайн-галерей є те, що розміщати матеріали на сайті просто, швидко й доступно.

Для спрощення та автоматизації процесу розміщення цифрових робіт спеціалістів з дизайну, створення їх резюме в одному місці, та через актуальність, велику популярність та невелику кількість подібних ПЗ необхідно розробити програмне забезпечення вебсайту для онлайн-галереї "RiArt".

## 1.2 Аналіз існуючих аналогів вебсайтів галерей

Розглянемо та проаналізуємо основні недоліки та переваги вже існуючих на даний час вебсайтів онлайн-галерей.

За видом матеріалів, які можуть розміщатися в онлайн-галереях, вони діляться на музейні, художні та digital art. Музейні й художні галереї є менш популярними, й відносяться більш до біржових галерей, тому що на них безпосередньо не розміщують свої роботи, а лише їх знімки чи оцифровані варіанти. При цьому фізичні копії знаходяться окремо у власників чи в музеях.

Одними з найпопулярніших онлайн-галерей аналогів є наступні:

- “Pixiv”;
- “ArtStation”.

Pixiv – це онлайн-галерея та онлайн-сервіс який об’єднує дизайнерів й аніматорів у свого роду спільноту. Галерея створена як вебсайт для розміщення, публікації, та популяризації робіт дизайнерів, переважно країн далекого сходу.

Інтерфейс головної сторінки Pixiv представлений на рисунку 1.2.

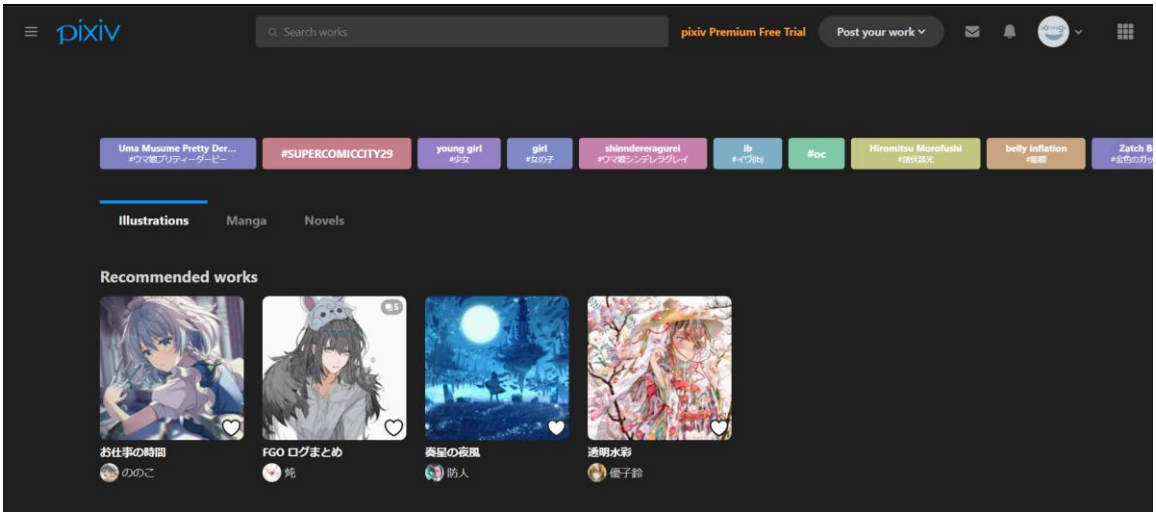


Рисунок 1.2 – Головна сторінка вебсайту Pixiv

Перелік можливостей які надає даний вебсайт:

- реєстрація та авторизація користувачів;
- публікація будь-яких цифрових робіт що відповідають вимогам сайту, окрім 3D моделей (рисунок 1.3);
- додання оцінку до опублікованих робіт користувачів;
- оформити платну підписку, для отримання доступу до додаткового функціоналу вказаному на рисунку 1.4 [2];
- пошук робіт у галереї за фільтром;
- персональна сторінка користувача з усіма його публікаціями (рисунок 1.5);
- заповнення короткої інформації про користувача (рисунок 1.6).

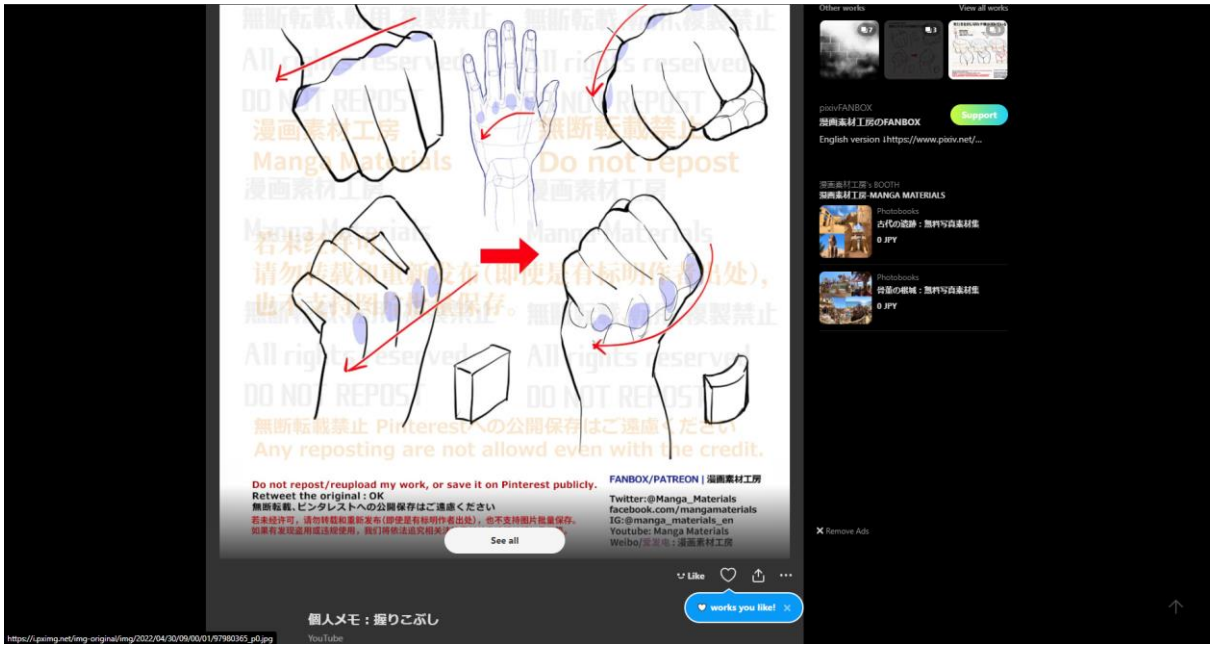


Рисунок 1.3 – Сторінка опублікованої цифрової роботи на сайті Pixiv

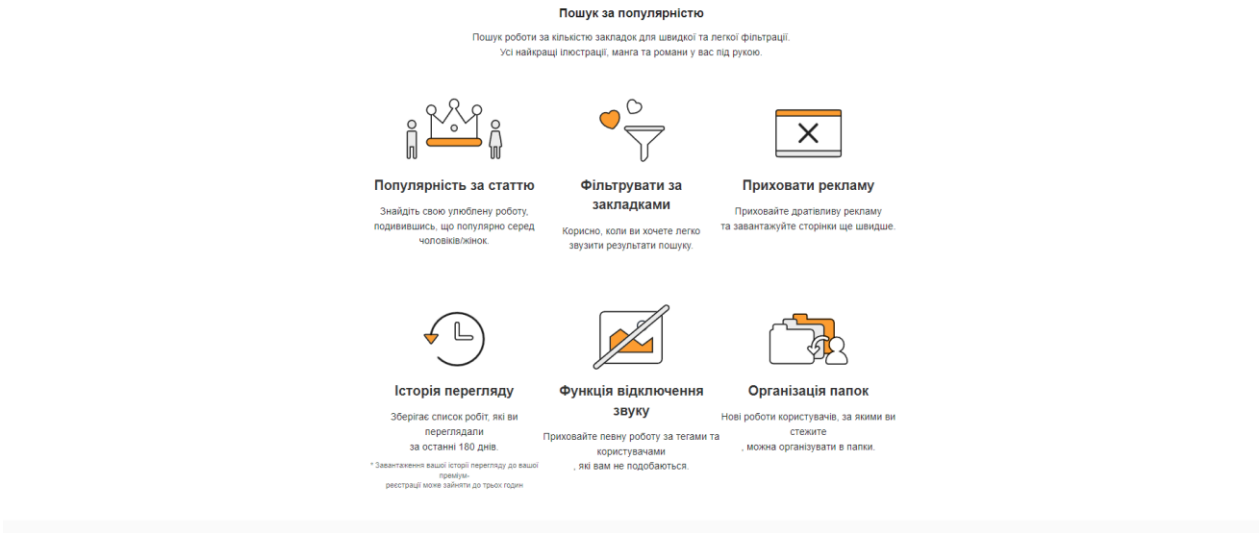


Рисунок 1.4 – Сторінка з інформацією про додатковий платний функціонал вебсайту Pixiv

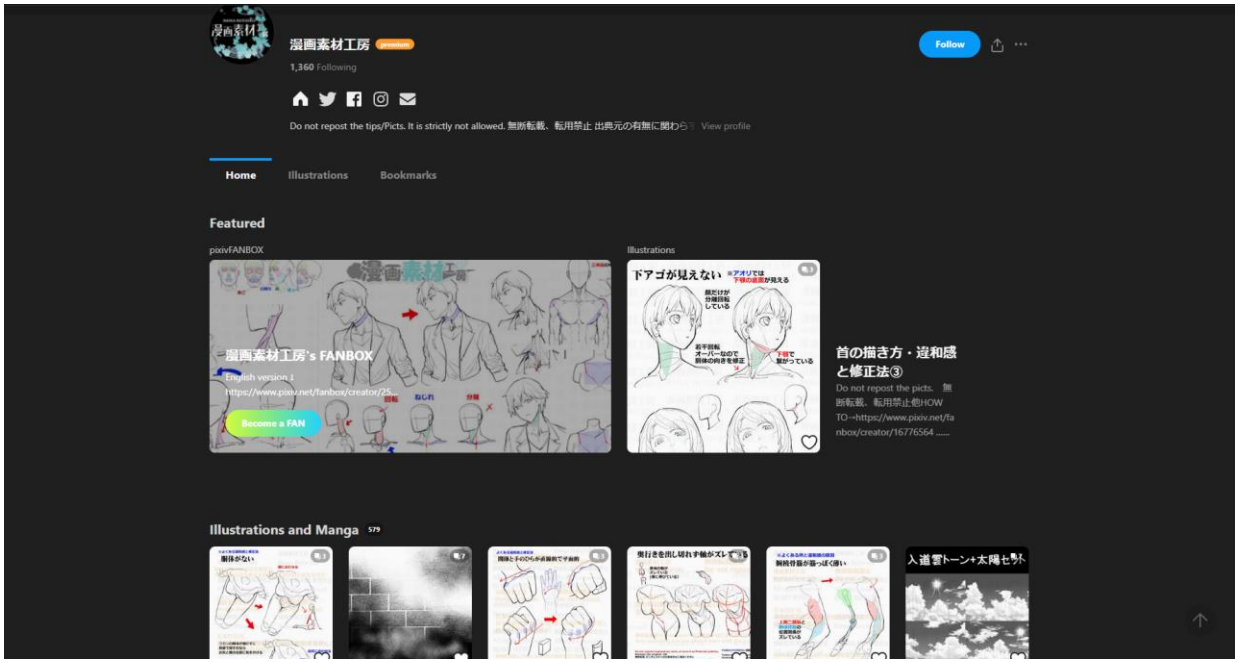


Рисунок 1.5 – Персональна сторінка користувача вебсайту Pixiv

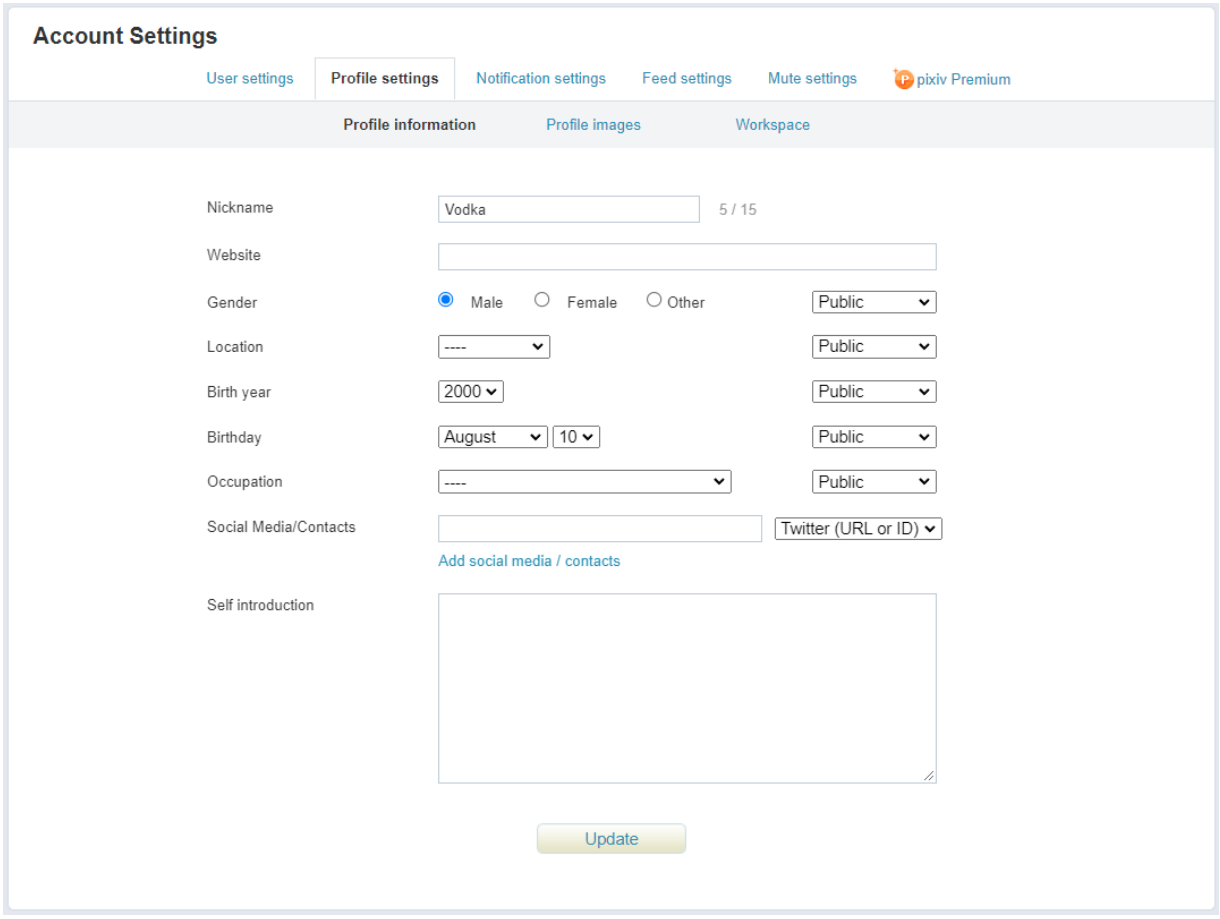


Рисунок 1.6 – Сторінка редагування інформації користувача вебсайту Pixiv

Головні переваги онлайн-галереї Pixiv [3]:

1) Наявність тегів на сайті. Теги, або ключові слова, описують роботи та допомагають користувачам швидко знаходити ілюстрації, які можуть їх зацікавити. Позначка ілюстрації з її ознаками значно полегшує пошук конкретних робіт. Кожна робота може мати до десяти тегів.

2) Можливість помічати та додавати роботи до закладок користувача. Закладки на сайті дозволяють передивитися доданні роботи в у будь-який час.

Одним з недоліків вебсайту є відсутність додаткового функціоналу для розміщення резюме та більш повної інформації про дизайнера.

Інтерфейс вебсайту виконаний в мінімалістичному стилі та нейтрально-сірих тонах. Графічний інтерфейс має свої недоліки. На головній сторінці смуга прокрутки з вибором фільтрів та тегів, не вміщується на екрані й не має додатково відступу з правого краю вебсайту, тому її досить незручно використовувати (рисунок 1.7).

Смуга прокрутки – елемент графічного інтерфейсу користувача, за допомогою якого користувач може переміщуватись по інформації, що не вміщується на екрані.

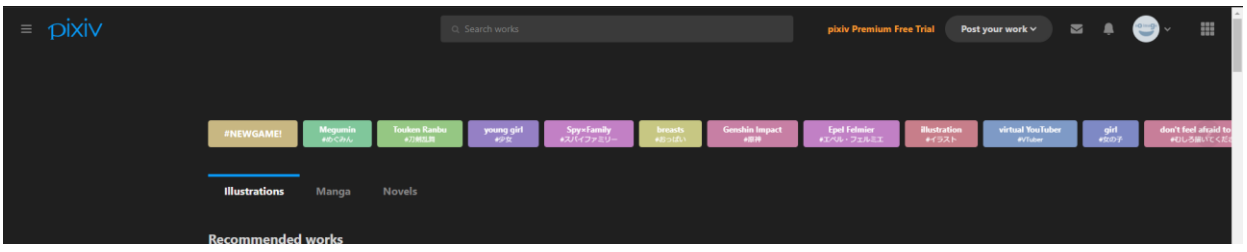


Рисунок 1.7 – Сторінка редагування інформації користувача вебсайту Pixiv

Незважаючи на впровадження у вебсайт гарних рішень, більша частина функціоналу є платною і недоступною для звичайних користувачів, тому представлена онлайн-галерея не може бути використана для пошуку саме спеціалістів зі сфери дизайну, і лише мінімально розповідає про творче життя користувача.

“ArtStation” – це онлайн галерея, біржа та міні-форум. Вона спеціалізується на більш сучасних поняттях художника як артиста та навіть продажу робіт у електронному вигляді. Тому на сайті переважають більш сучасні роботи: 3Д моделі, концепт-роботи та багато інших робіт стилістики художників північного сходу.

Декілька сторінок вебсайту зображені на рисунках 1.8 - 1.10.

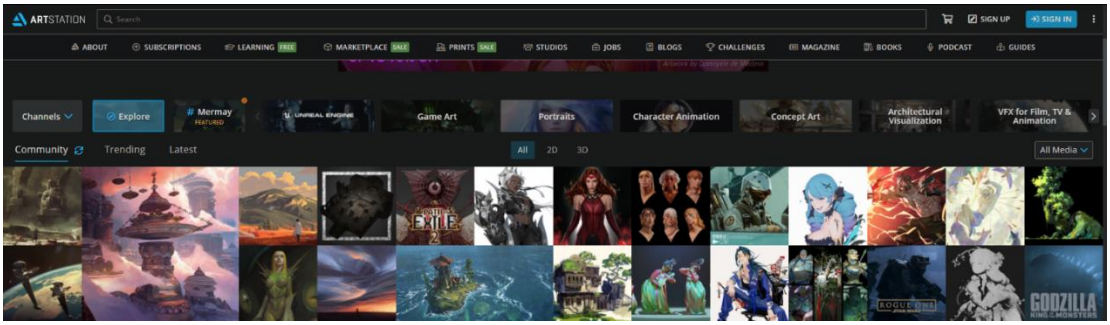


Рисунок 1.8 - Головна сторінка сайту ArtStation

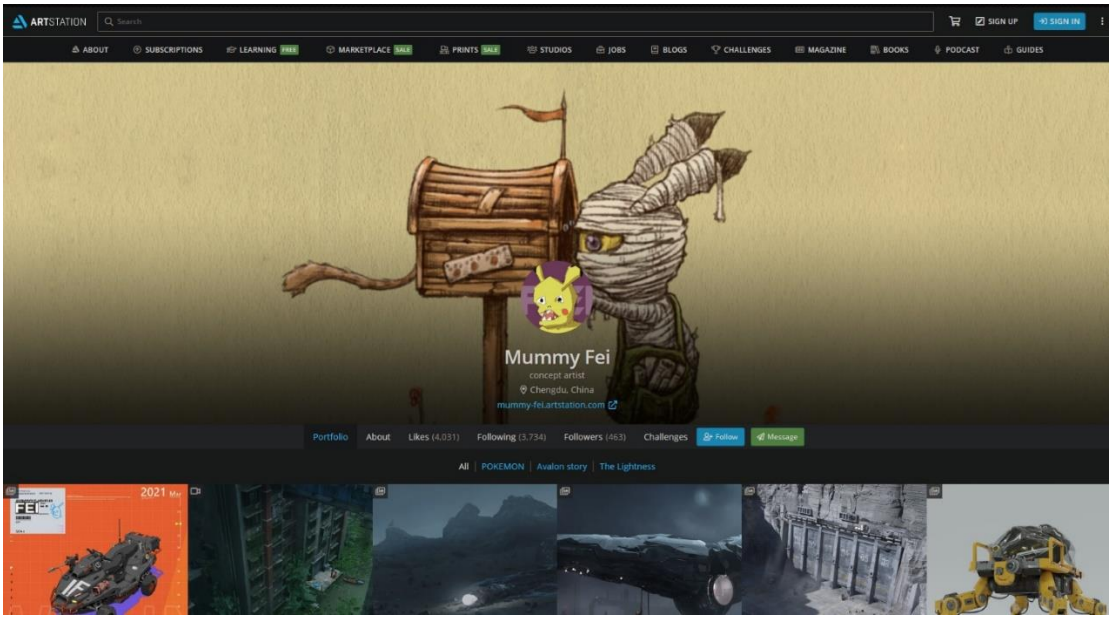


Рисунок 1.9 - Сторінка профілю користувача

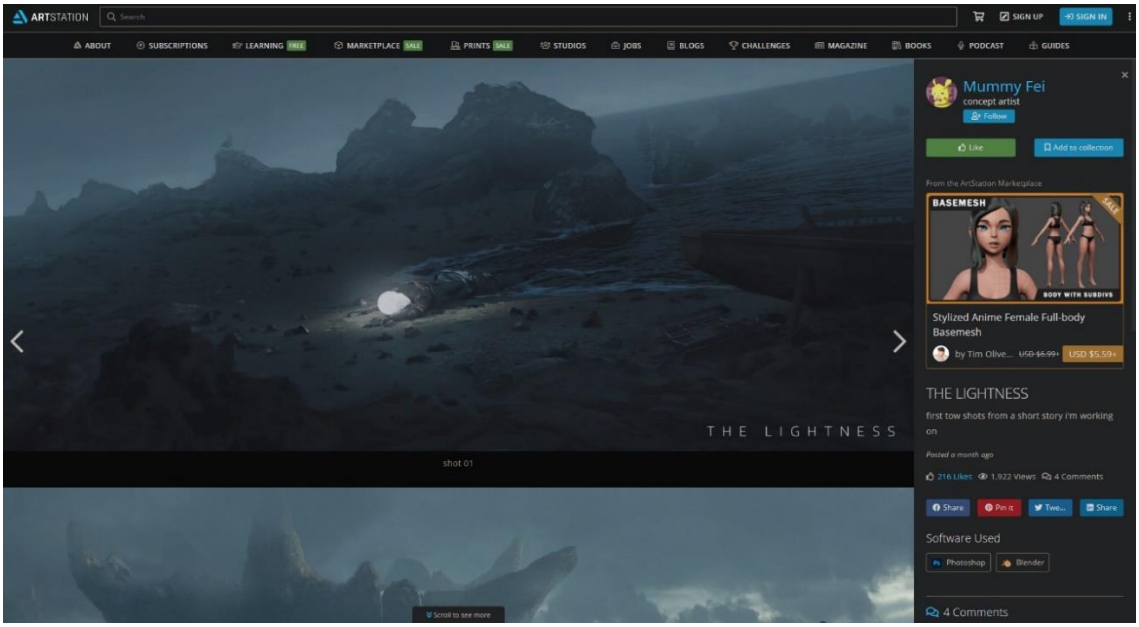


Рисунок 1.10 – Сторінка опублікованої роботи

Перелік можливостей які надає даний вебсайт:

- реєстрація та авторизація користувачів;
- публікація будь-яких цифрових робіт що відповідають вимогам сайту;
- оформлення платної підписки для отримання доступу до додаткового функціоналу;
- пошук робіт у галереї за фільтром;
- персональна сторінка користувача з усіма його публікаціями;
- заповнення короткої інформації про користувача (без можливості створення біографії);
- створення та пошук блогу;
- розміщення реклами;
- пошук та реєстрація на навчальні курси;
- магазин для продажу чи покупки художніх робіт користувачів у цифровому вигляді.

Переваги сайту:

- зручний зрозумілий інтерфейс;
- адаптивний вебсайт на кожну платформу;

- багато плагінів для браузера для перенаправлення на цей сайт;
- багато додаткового функціоналу (Блоги, Робота, Подкасти).

Недоліки сайту:

- присутня реклама;
- частина функціоналу сайту платна;
- занадто великий функціонал з великої кількістю можливостей, який перетворює онлайн-галерею у інший вебсайт такий як біржи й блог-сайти.

Можна розбити висновок що вебсайт аналог “ArtStation” можна вважати онлайн-галереєю. Переважна більшість звичайних користувачів, без підписки, буде мати обмежений функціонал, а знайти їх роботи з профілем буде дуже тяжко, через популяризації на вебсайті платної підписки та її привілеї у розміщенні робіт користувачів з нею, першими у списках. Саме фахівці-початківці з дизайну зараз є тим контингентом на який потрібно робити акцент, через їх велике число, для того щоб об’єднувати їх у більш великі групи.

## 1.3 Аналіз методів розробки вебсайтів

Існує багато різноманітних методів для розробки програмного забезпечення вебсайтів. Одним з найпоширеніших на даний час є використання онлайн конструкторів сайтів. Головною перевагою, через яку вони стали популярними, це їх доступність та простота.

Онлайн конструктори сайтів та системи керування вмістом (далі СКВ) допомагають створювати вебсайти без додаткових знань, таких як мови програмування та розмітки.

СКВ – це програмне забезпечення для створення та подальшої організації вебсайтів.

Онлайн конструкторами може бути спеціальний плагін, який доданий чи реалізований у СКВ, чи готовий реалізований сервіс на SaaS платформах.

Плагін – це програмний модуль, який підключається до програми чи вебсайту для подальшого розширення їх можливостей та функціоналу.

SaaS – це модель роботи с програмним забезпеченням, яка передбачає що технічне обслуговування, підтримка, та розробка програмної частини береться на себе постачальником послуг, з подальшим наданням користувачу доступу до використання ПЗ [4].

При роботі з СКВ чи SaaS які містять конструктори сайтів, користувачам надається свій персональний кабінет, для подальшого налаштування та вибору шаблону створюваного сайту. Дизайн сайту і його зміст розділений. Це основний принцип роботи конструкторів сайтів. Вебсайт та його оформлення, мінімально, але вже існує у вибраному шаблонні, і це і є одним із головних мінусів цих систем. Через це, ці технології зазвичай використовується для роботи з вже присутніми блоками на вебсайті, їх розміщенням та подальшим редагуванням.

Більшість конструкторів сайтів працюють повільно, тому вони не призначенні для створення багатофункціональних вебсайтів з не типовими завданнями. Однак, існують СКВ, які надають більше вбудованого функціоналу для розв’язання нетипових задач.

Конструктори сайтів часто використовують фахівці з різних творчих спеціальностей, оскільки, такий шаблон вебсайту як портфоліо чи резюме, зазвичай вбудовано в популярні СКВ чи SaaS. Наприклад наступна СКВ - WordPress, має вбудований плагін конструктору сайтів з шаблонами для створення портфоліо, однак це не є безкоштовним.

Одним з варіантів створення багатофункціональних та швидких вебсайтів, який і було обрано для створення програмного забезпечення для вебсайту онлайн-галереї “RiArt”, є розробка вебсайту з нуля. Для цього використовуються спеціалізовані мови програмування.

Зазвичай при створенні програмного забезпечення сайту з нуля, робота над ним розбивається на Frontend та Backend.

Frontend – це інтерфейс сайту, з яким працює безпосередньо користувач сайту. Під цим поняттям мають на увазі розробку інтерфейсу користувача й усього функціоналу з яким користувач буде взаємодіяти.

Backend – це серверна частина вебсайту. Серверна частина відповідає за логіку та роботу функціоналу з яким користувач буде не напряму, а через інтерфейс сайту.

Спеціалізовані мови розмітки, такі як HTML в парі з CSS та JavaScript є інструментами для розробки Frontend частини сайту.

HTML – мова гіпертекстової розмітки документів яка є основою усіх вебсайтів. Використовуються для представлення вебсторінки.

CSS – формальна мова, яка використовується для опису оформлення зовнішнього виду документів, які були створенні за допомогою мови гіпертекстової розмітки, наприклад HTML. CSS розділяє зовнішній вигляд сторінки вебсайту від її змісту.

JavaScript – мова програмування яка використовується для додання інтерактивності до сторінок вебсайту.

У Backend розробці програмного забезпечення сайту велика варіативність вибору інструментів, наприклад, мов програмування на яких буде створена серверна частина сайту. До найбільш популярних, з великою кількістю функціоналу, можна віднести наступні мови програмування:

- Java;
- PHP;
- C#;
- Python.

Кожна з наведених мов програмування має схожі реалізовані в них можливості для розробки Backend частини сайту, а деякі з них мають інструменти і для розробки Frontend частини, і вибір може залежати від знань спеціалісту тієї чи іншої мови програмування. Але найбільш важливим

фактором вибору є поставленні задачі які потрібно вирішити при створенні програмного забезпечення вебсайту та кількість інструментів які надає обрана мова програмування. Залежно від обраної мови програмування може варіюватися вибір інтегрованого середовища розробки та бази даних.

Для розробки програмного забезпечення вебсайту для онлайн-галереї "RiArt" було обрано варіант, який передбачає створення вебсайту з нуля.

## 1.4 Постановка задачі на розробку програмного забезпечення вебсайту для онлайн-галереї “RiArt”

У результаті проведення аналізу предметної області роботи онлайн-галерей сформулюємо наступну постановку практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов:

- розробити програмне забезпечення вебсайту для онлайн-галереї "RiArt" для автоматизації процесу розміщення та зберігання цифрових робіт й резюме спеціалістів з дизайну.

Метою розробки програмного забезпечення є

1) значне спрощення зберігання та розповсюдження робіт дизайнерів;
2) зменшення часу пошуку інформації про фахівців з дизайну;
3) розповсюдження інформації про дизайнерів.

За допомогою проведеного аналізу предметної області та аналогів онлайн-галерей, визначення їх основних можливостей та недоліків, можна зробити висновок, що онлайн-галереї є популярними на ринку ПЗ, а деякі з них, не реалізовують повного та безоплатного функціоналу створення резюме спеціалісту, що і вирішується створенням власного ПЗ.

1) Вимоги до функціональних характеристик:

Повинні бути реалізовані наступні функції у програмному забезпеченні вебсайту для онлайн галереї “RiArt”:

- авторизація користувача на сайті;
- реєстрація користувача на сайті;
- завантаження роботи на сайт;
- скачування роботи з сайту;
- публікація роботи на сайті;
- перегляд робіт на головній сторінці сайту;
- пошук робіт;
- видалення опублікованих робіт;
- блокування опублікованої роботи клієнта;
- перегляд опублікованої роботи;
- перегляд профілю зареєстрованого користувача;
- редагування профілю клієнта;
- вихід з персонального облікового запису.

2) Вимоги до вхідних та вихідних даних:

Введення вхідних текстових даних повинно бути здійснено користувачем за допомогою периферійного пристрою – комп’ютерної клавіатури. Завантаження матеріалів (зображень, моделей тощо) до сайту повинна здійснюватися за допомогою вибору потрібного матеріалу з диска комп’ютера через функціонал на сайті. Текстові данні введенні користувачем повинні зберігатися у базі даних на сервері. Матеріали завантажені на сайт повинні зберігатися на сервері.

Вхідними даними є:

- основна інформація про користувача для авторизації чи реєстрації (пароль, електронна пошта);
- додаткова інформація про користувача (ПІБ, резюме, біографія і т.д);
- інформація про опубліковану роботу (назва, коротка інформація про роботу, матеріали які представляють роботу);

- інформація для пошуку на сайті (обрані фільтр чи фільтри, назва роботи чи ПІБ користувача).

Вхідні та вихідні данні повити відображатися користувачу як при введенні інформації, так і при виведенні.

Вихідними даними є:

- список опублікованих робіт;
- опублікована робота;
- коротка інформація про опубліковані роботи та їх користувачів;
- уся доступна користувачу інформація про його профіль;
- уся доступна користувачу інформація про профіль іншого користувача;
- зображення.

Вихідні дані – це уся інформація що зберігається на сервері. Це може бути запис у базі даних, чи ресурс на сервері.

3) Вимоги до складу й параметрів технічних засобів:

Програмне забезпечення вебсайту підтримується наступними пристроями:

- Персональні комп’ютери.

4) Вимоги до інформаційної та програмної сумісності:

Програмне забезпечення вебсайту підтримується наступними операційними системами:

- Windows версії 8 та вище.

Для можливості працювати з вебсайтом потребується браузер. Далі перелічені браузери які підтримуються програмним забезпеченням:

1) Firefox версії 30 та вище;
2) Opera версії 10 та вище;
3) Chrome версії 35 та вище.

# 2 ПРОЄКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБСАЙТУ ДЛЯ ОНЛАЙН-ГАЛЕРЕЇ “RiArt”

## 2.1 Ескізний проєкт програмного забезпечення вебсайту для онлайн-галереї “RiArt”

За допомогою проведеного аналізу основних вимог було розроблено ескізний проєкт. Для створення проєкту було використано спеціальну мову моделювання UML.

Для проєктування програмного забезпечення на даному етапі було використано об’єктноорієнтовану методологію та створені наступні моделі:

- Модель варіантів використання;
- Діаграми діяльності;
- Прототип інтерфейсу користувача.

### 2.1.1 Модель варіантів використання

Було виявлено 3 потенціальних користувачі програмного забезпечення вебсайту для онлайн-галереї “RiArt”, це Адміністратор, Клієнт та Гість.

Адміністратор – це працівник галереї “RiArt”, що відповідає за адміністрування системи. Він керує правами користувачів (якщо такі є) й керує контентом інформації на сайті.

Гість – це користувач який відвідує сайт.

Клієнт – це авторизований на сайті користувач.

На основі виявлених типів користувачів, були виявленні актори системи, представленні на рисунку 2.1.

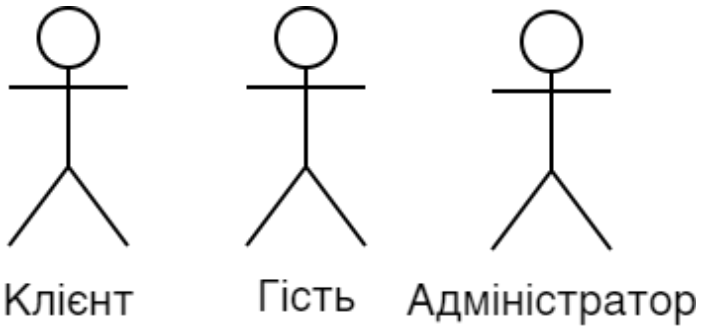


Рисунок 2.1 - Актори системи

Після аналізу функціональних характеристик та вимог програмного забезпечення було виявлено та виділено варіанти використання наведенні далі.

Варіанти використання:

- авторизація;
- реєстрація;
- завантаження роботи на сайту;
- скачування роботи з сайту;
- публікація роботи на сайті;
- перегляд публікацій на головній сторінці сайту;
- пошук робіт;
- видалення публікації;
- блокування публікації клієнта;
- перегляд опублікованої роботи;
- перегляд профілю клієнта;
- редагування профілю клієнта;
- вихід з облікового запису.

Усі варіанти використання представлені на діаграмі варіантів використання на рисунку 2.2.

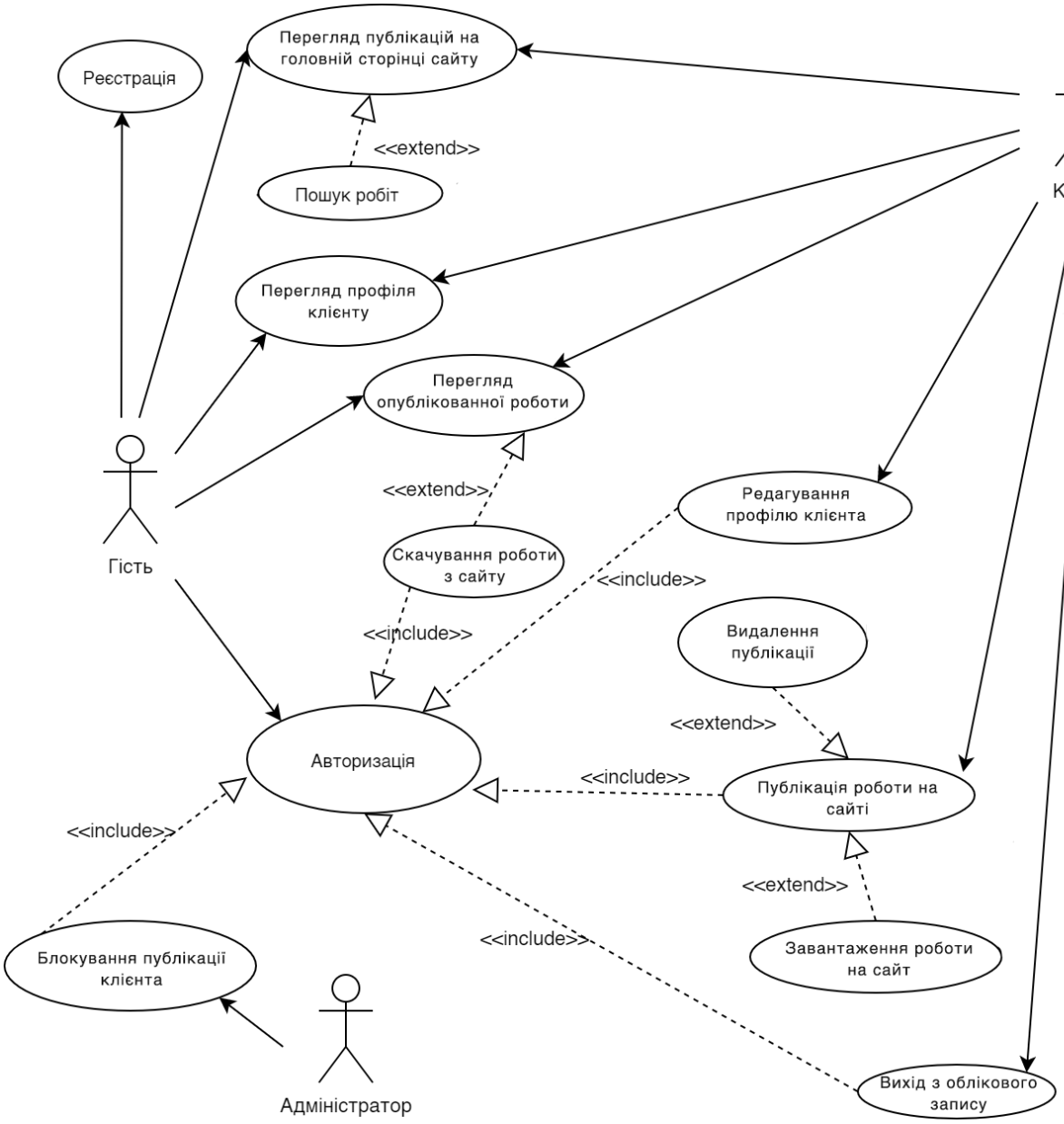


Рисунок 2.2 – Діаграма варіантів використання

### 2.1.2 Опис варіантів використання

Проаналізувавши отриманні варіанти використання та їх діаграму, проведемо більш детальний опис кожного з варіантів використання. В таблиці 2.1 представлено усі варіанти використання.

Таблиця 2.1 – Варіанти використання та їх короткий опис

| Код | Актор | Назва | Формулювання |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| А1 | Адміністратор | Авторизація | Дозволяє авторизуватися та увійти до свого профілю на сайті |
| А2 | Адміністратор | Блокування публікації клієнта | Дозволяє заблокувати публікацію з роботою будь-якого клієнта |
| К1 | Клієнт | Авторизація | Дозволяє авторизуватися у системі |
| К2 | Клієнт | Редагування профілю клієнта | Дозволяє відредагувати клієнту інформацію його персонального профілю |
| К3 | Клієнт | Завантаження роботи на сайт | Дозволяє завантажити обрану роботу на сайт |
| К4 | Клієнт | Публікація роботи на сайті | Дозволяє опублікувати роботу для відкриття публічного доступу до неї на сайті |
| К5 | Клієнт | Скачування роботи з сайту | Дозволяє скачати вибрану роботу з сайту |
| К6 | Клієнт | Вихід з облікового запису | Дозволяє вийти з системи сайту |
| К7 | Клієнт | Пошук робіт | Дозволяє знайти публікації робіт за заданими фільтрами |
| К8 | Клієнт | Перегляд публікацій на головній сторінці сайту | Дозволяє переглянути список публікацій робіт на головній сторінці сайту |

Продовження таблиці 2.1

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| К9 | Клієнт | Видалення публікації | Дозволяє видалити будь-яку з опублікованих самим клієнтом робіт |
| К10 | Клієнт | Перегляд опублікованої роботи | Дозволяє переглянути вибрану публікацію |
| К11 | Клієнт | Перегляд профілю клієнту | Дозволяє перейти до сторінки сайту з персональним профілем клієнта |
| Г1 | Гість | Реєстрація | Дозволяє створити профіль користувача на вебсайті |
| Г2 | Гість | Авторизація | Дозволяє авторизуватися та увійти до свого профілю на сайті |
| Г3 | Гість | Перегляд опублікованої роботи | Дозволяє переглянути вибрану публікацію |
| Г4 | Гість | Перегляд публікацій на головній сторінці сайту | Дозволяє переглянути список робіт на головній сторінці вебсайту |
| Г5 | Гість | Пошук робіт | Дозволяє знайти публікації робіт за заданими фільтрами |
| Г6 | Гість | Перегляд профілю клієнту | Дозволяє перейти до сторінки вебсайту з персональним профілем клієнта |

Проведемо конкретизацію основних варіантів використання:

А2 – Блокування публікації клієнта.

Основна діюча особа: Адміністратор.

Інші учасники прецеденту: зв’язок з варіантом авторизація.

Передумови, необхідні для ініціювання прецеденту: адміністратор ініціював запит на блокування.

Короткий опис

Даний варіант використання дозволяє адміністратору заблокувати вже опубліковану роботу окремо вибраного клієнта якщо вона не відповідає правилам сайту.

Г2 – Реєстрація.

Основна діюча особа: Гість.

Інші учасники прецеденту: відсутні зв’язки з іншими варіантами.

Передумови, необхідні для ініціювання прецеденту: клієнт ініціював процес реєстрації натисканням кнопки.

Короткий опис

Даний варіант використання дозволяє клієнту зареєструватися на сайті.

К3 – Завантаження роботи на сайту.

Основна діюча особа: Клієнт.

Інші учасники прецеденту: зв’язок з варіантом використання Опублікувати роботу на сайті.

Передумови, необхідні для ініціювання прецеденту: клієнт вибрав роботу для відправки на своєму пристрої.

Короткий опис

Даний варіант використання дозволяє клієнту завантажити обрану роботу на сайт.

К4 – Публікація роботи на сайті.

Основна діюча особа: Клієнт.

Інші учасники прецеденту: зв’язок з варіантами - Видалення публікації, Завантаження роботи на сайт, Авторизація.

Передумови, необхідні для ініціювання прецеденту: клієнт обрав роботу для публікації.

Короткий опис

Даний варіант використання дозволяє клієнту опублікувати вибрані роботи на сайті.

К5 – Скачування роботи з сайту.

Основна діюча особа: Клієнт.

Інші учасники прецеденту: зв’язок з варіантами Авторизація та Перегляд опублікованої роботи.

Передумови, необхідні для ініціювання прецеденту: Клієнт ініціював запит на отримання ресурсу сайту.

Короткий опис

Даний варіант використання дозволяє клієнту скачати обрану роботу з сайту на свій пристрій.

К8, Г4 – Перегляд публікацій на головній сторінці сайту.

Основні діючі особи: Клієнт, Гість.

Інші учасники прецеденту: зв’язок з варіантом Пошук робіт.

Передумови, необхідні для ініціювання прецеденту: клієнт перейшов до галереї.

Короткий опис

Даний варіант використання представляє клієнту отримати список робіт на головній сторінці сайту.

К9 – Видалення публікації.

Основна діюча особа: Клієнт.

Інші учасники прецеденту: зв’язок з варіантом Публікація роботи на сайті.

Передумови, необхідні для ініціювання прецеденту: клієнт ініціював видалення вибраної роботи.

Короткий опис

Даний варіант використання дозволяє клієнту видалити обрану їм роботу яка була опублікована самим клієнтом.

К10, Г3 – Перегляд опублікованої роботи.

Основні діючі особи: Клієнт, Гість.

Інші учасники прецеденту: зв’язок з варіантом - Скачування роботи з сайту.

Передумови, необхідні для ініціювання прецеденту: клієнт натиснув на роботу на сайті.

Короткий опис

Даний варіант використання дозволяє перейти на вебсторінку для перегляду обраної роботи.

Г6 – Перегляд профілю клієнту.

Основні діючі особи: Гість, Клієнт.

Інші учасники прецеденту: відсутні зв’язки з іншими варіантами.

Передумови, необхідні для ініціювання прецеденту: актор обрав потрібного користувача на сайті.

Короткий опис

Даний варіант використання дозволяє перейти на сторінку для перегляду профілю зареєстрованого користувача (клієнта) сайту.

К10 – Редагування профілю клієнта.

Основна діюча особа: Клієнт.

Інші учасники прецеденту: включає варіант Авторизація.

Передумови, необхідні для ініціювання прецеденту: клієнт ініціював редагування свого профілю.

Короткий опис

Даний варіант використання дозволяє клієнту редагувати чи доповнити інформацію власного профілю користувача на сайті.

Авторизація акторів:

1) Адміністратор

Адміністратор має свій власний обліковий запис. Цей акаунт має права адміністратора. Завдяки цьому адміністратор може редагувати деяку інформацію у базі даних.

2) Гість

Гість може авторизуватися до сайту, для цього йому треба вказати свою електрону пошту, та пароль заданий при реєстрації.

Завдяки цьому користувач зможе повноцінно користуватися можливостями сайту.

### 2.1.3 Опис сценаріїв варіантів використання

Для формалізації сценаріїв варіантів використання було створено декілька основних діаграм діяльності, на основі детального опису основних варіантів використання.

Детальний опис сценаріїв використання наведено у таблицях 2.2 – 2.4.

Таблиця 2.2 – Сценарій використання «Реєстрація»

| **Назва прецеденту** | **Реєстрація** |
|---|---|
| Опис прецеденту | Гість сайту заповнюючи спеціальну форму реєструється на вебсайті |

Продовження таблиці 2.2

| **Назва прецеденту** | **Реєстрація** |
|---|---|
| Дійові особи прецеденту | Гість |
| Зв’язок з іншими варіантами використання | Відсутні зв'язки |
| Базовий потік | Користувач ввів логін та пароль у поля форми реєстрації, та натиснув кнопку Реєстрація |
| Альтернативні потоки | Якщо користувач з таким логіном вже існує, логін чи пароль має заборонені символи, логін чи пароль не відповідає мінімальним та максимальним розмірам, або користувач не ввів усю інформацію в обов’язкові для заповнення поля йому видається сповіщення про помилку |
| Передумови | Завантажена форма реєстрації на вебсторінці |
| Пост умови | Користувач створює свій аккаунт, та одночасно входить до нього |
| Особливості | Автоматична аутентифікація та авторизація користувача |

Таблиця 2.3 – Сценарій використання «Авторизація» та «Вихід з облікового запису»

| **Назва прецедентів** | **Авторизація та вихід з облікового запису** |
|---|---|
| Опис прецеденту | Будь-який користувач, який попередньо зареєстрований, може авторизуватися або вийти зі свого облікового запису на сайті. |
| Дійові особи прецеденту | Вихід з облікового запису – Адміністратор, Клієнт. Авторизація (аутентифікація) – Гість. |

Продовження таблиці 2.3

| **Назва прецеденту** | **Авторизація та вихід з облікового запису** |
|---|---|
| Зв’язок з іншими варіантами використання | Вийти з облікового запису – включає варіант авторизацію.<br>Варіанти використання: Опублікувати роботу на сайті, Редагування профілю клієнта, Заблокувати опубліковану роботу клієнту, Скачування роботи з сайту включають обов'язковим варіантом авторизацію (аутентифікацію). |
| Базовий потік | Гість заповнивши поля форми авторизації, такі як логін та пароль, зайшов до свого облікового запису, та пройшовши аутентифікацію наділяється функціоналом клієнту чи адміністратору. |
| Альтернативні потоки | Клієнт чи адміністратор вийшовши з облікового запису, наділяються функціоналом гостя. |
| Передумови | Користувач натиснув кнопку вийти з профілю/авторизація. |
| Пост умови | Авторизація – Користувач входить до власного облікового запису, чи виводиться помилка результату авторизації.<br>Вихід з облікового запису – Користувач виходить з власного облікового запису. |
| Виняткові ситуації | Користувача не вдалося авторизувати. |

Таблиця 2.4 – Сценарій використання «Пошук робіт»

| **Назва прецеденту** | **Пошук Робіт** |
|---|---|
| Опис прецеденту | Користувач, використовуючи головну сторінку галереї, ініціює пошук робіт. |
| Дійові особи прецеденту | Гість, Клієнт. |

Продовження таблиці 2.4

| **Назва прецеденту** | **Пошук Робіт** |
|---|---|
| Зв’язок з іншими варіантами використання | Пошук робіт розширює варіант використання Перегляд публікацій на головній сторінці сайту. |
| Базовий потік | Користувач вибирає пункт головного фільтра - пошук за Автором роботи чи за Назвою роботи, вводить в строку пошуку данні за якими буде відбуватися пошук у базі даних. |
| Альтернативні потоки | Користувач додатково вибирає пункти головних фільтрів Авторів робіт, чи Назв Робіт, перед ініціюванням пошуку. |
| Передумови | Користувач перейшов до головної сторінки галереї |
| Пост умови | Користувачу представляється результат пошуку у вигляді списку відфільтрованих робіт представлених на головній вебсторінці онлайн-галереї за запитом пошуку, або повідомлення про помилку пошуку. |
| Виняткові ситуації | За запитом користувача не було знайдено жодної роботи. |

Діаграми основних варіантів діяльності наведенні на рисунках 2.3 - 2.5.

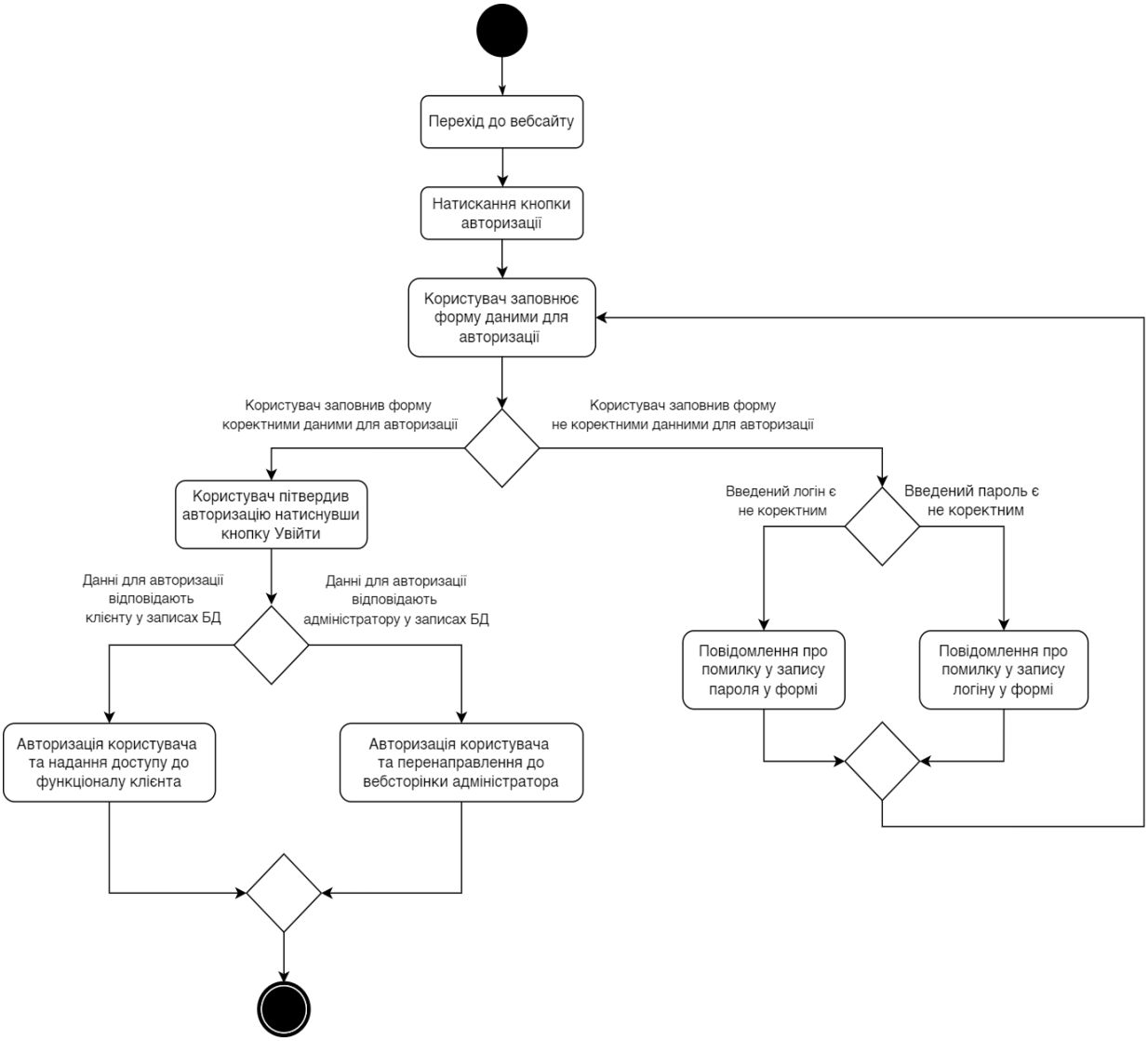


Рисунок 2.3 – Діаграма діяльності для варіанту використання “Авторизація”

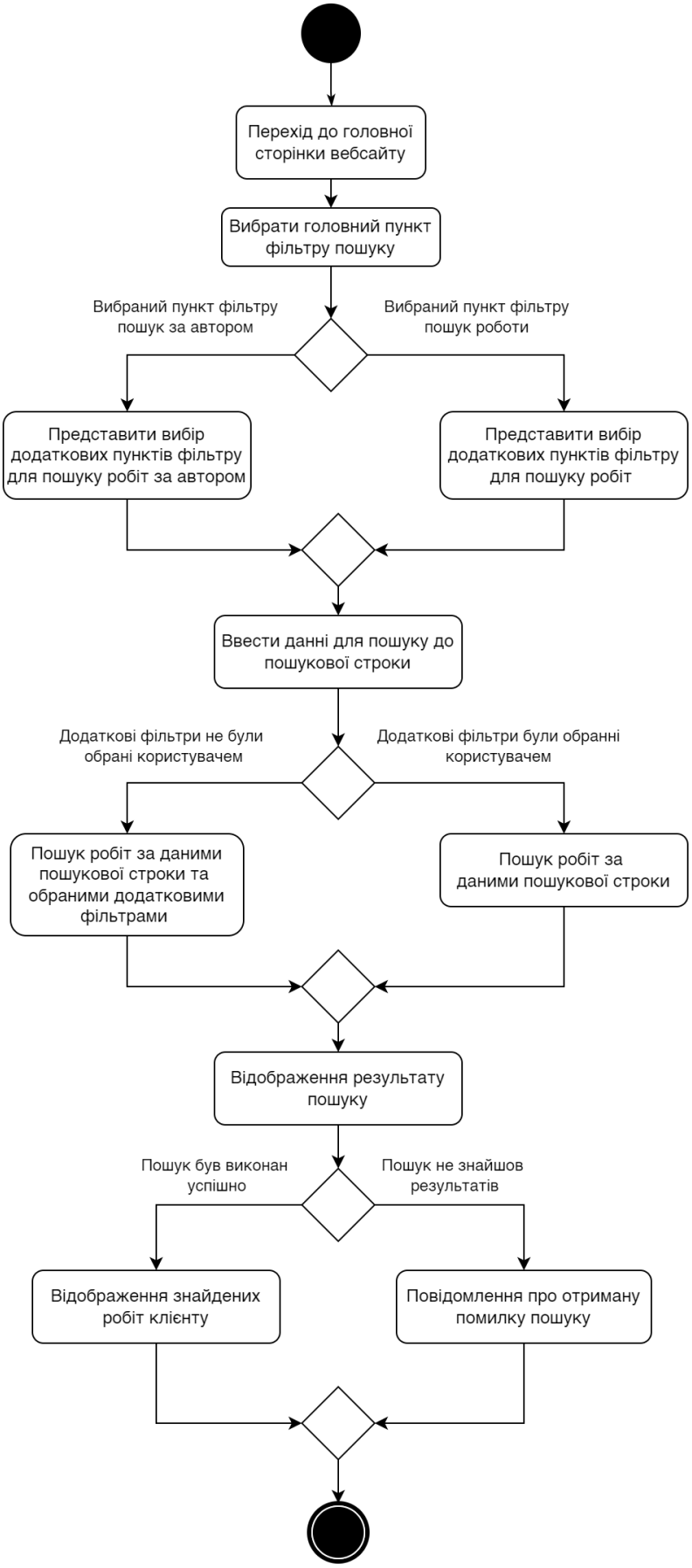


Рисунок 2.4 - Діаграма діяльності для варіанту використання “Пошук робіт”

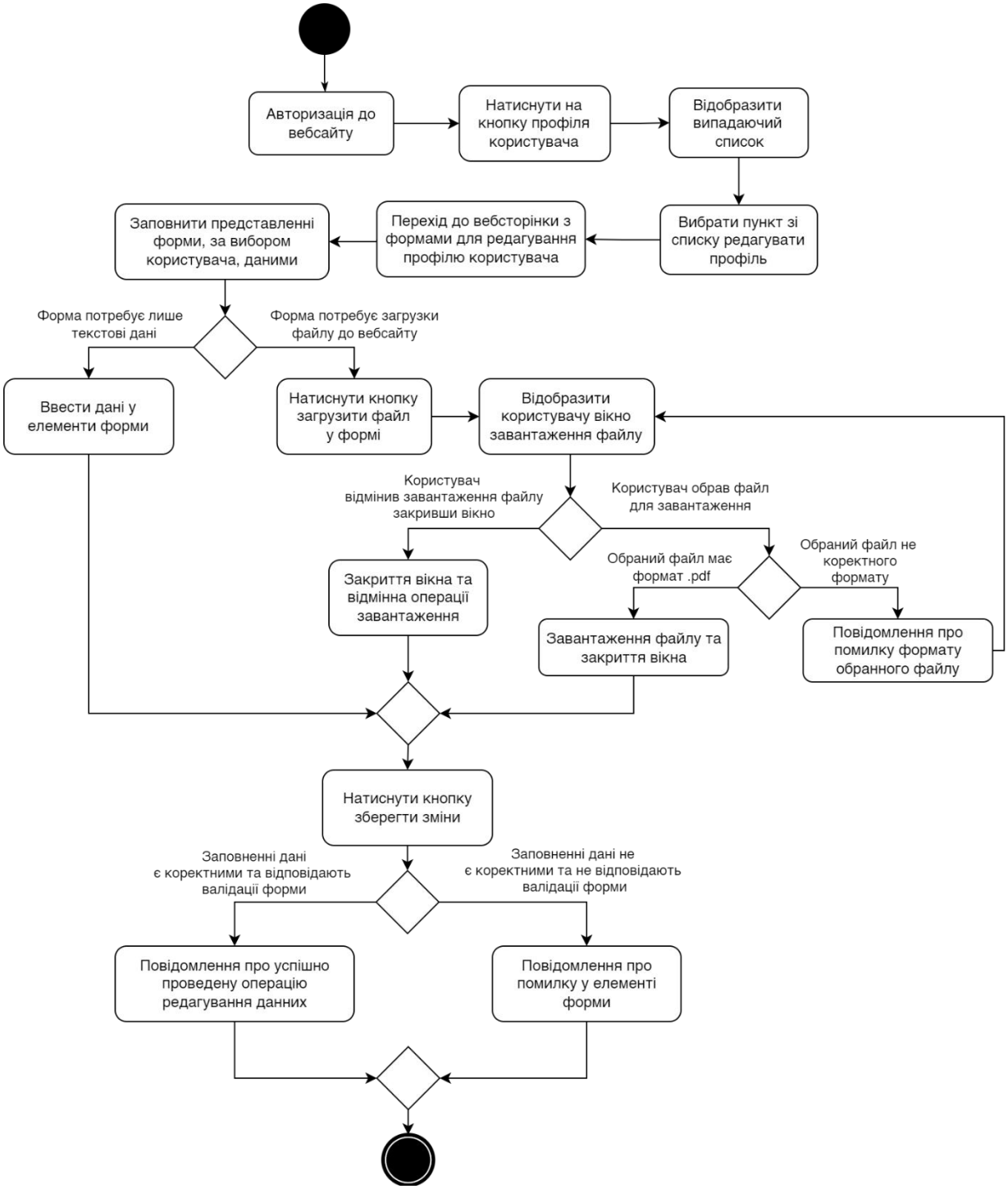


Рисунок 2.5 - Діаграма діяльності для варіанту використання “Редагування профілю клієнта”

### 2.1.4 Прототип інтерфейсу користувача

Для уточнення кінцевого інтерфейсу програмного забезпечення вебсайту для онлайн-галереї “RiArt”, аналізу елементів інтерфейсу, необхідних для виконання варіантів використання акторами, та їх зв’язку між собою, представлено схеми інтерфейсу взаємодії користувача з вебсайтом.

Зазвичай, вебсторінки розділяють на три рівні для розміщення інформації у кожному блоку:

- голова сторінки, де зазвичай розміщується навігація по сайту, кнопки для реєстрації чи авторизації та пошукові форми;
- контент сторінки чи тіло, де розміщується найважливіша інформація, яку часто буде переглядати користувач, чи функціональна частина сайту з якою буде часто взаємодіяти користувач, наприклад фільтри;
- нижній колонтитул сторінки, куди виносять інформацію про контактну інформацію підтримки вебсайту, розробників, представників компанії і т.д. а також посилання на вебсторінку з інформацією про сайт чи значок копірайту.

Користувач зазвичай зчитує інформацію зліва-направо, тому першим елементом було вирішено розмістити назву сайту у голові сторінки. Після, головний функціонал - авторизація та реєстрації користувача.

Головна сторінка сайту повинна чітко давати представлення що це галерея. Розмістивши на головній сторінці назву сайту, та додавши уточнення що це галерея, допоможе відвідувачу зрозуміти куди він потрапив. А розміщений список робіт, допоможе ознайомитись з популярними роботами на сайті.

Головна сторінка, яку побачить користувач при переході на вебсайт, зображена на рисунку 2.6

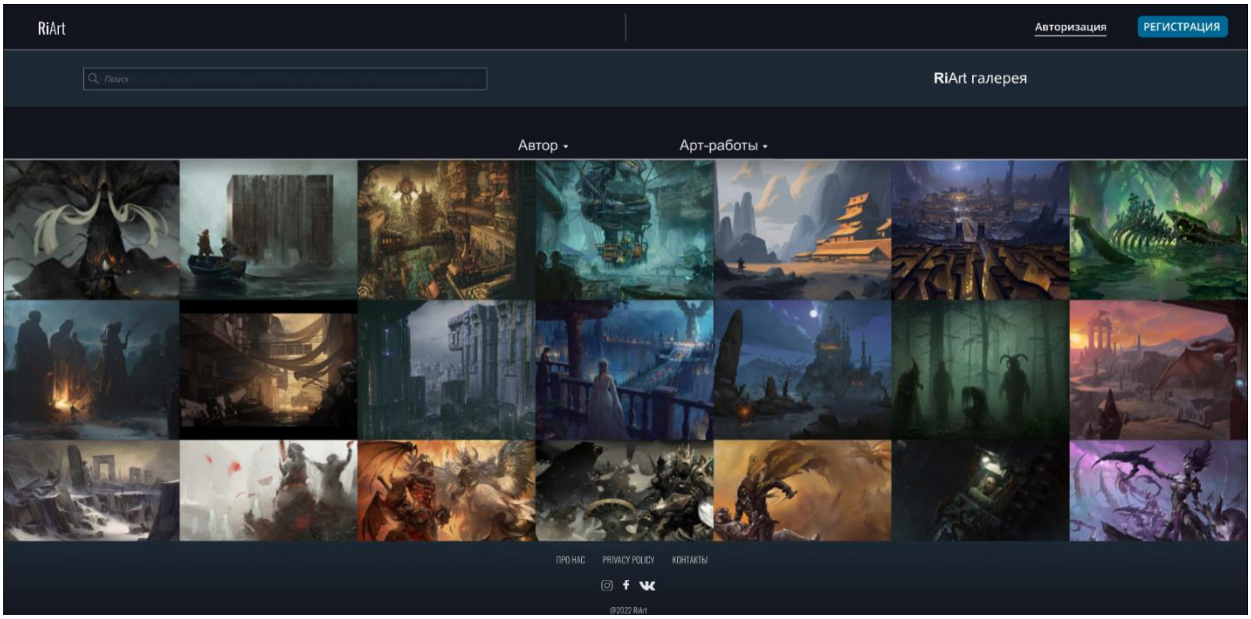


Рисунок 2.6 – Головна сторінка вебсайту

Для доступу до усіх основних функцій програмного забезпечення вебсайту потрібно авторизуватися до облікового запису. Для цього потрібно попередньо мати зареєстрований обліковий запис.

Вікно реєстрації було вирішено відкривати на тій же самій вебсторінці, за допомогою відкриття користувачу вікна з формою реєстрації поверх головної сторінки.

Вікно реєстрація користувача зображено на рисунку 2.7.

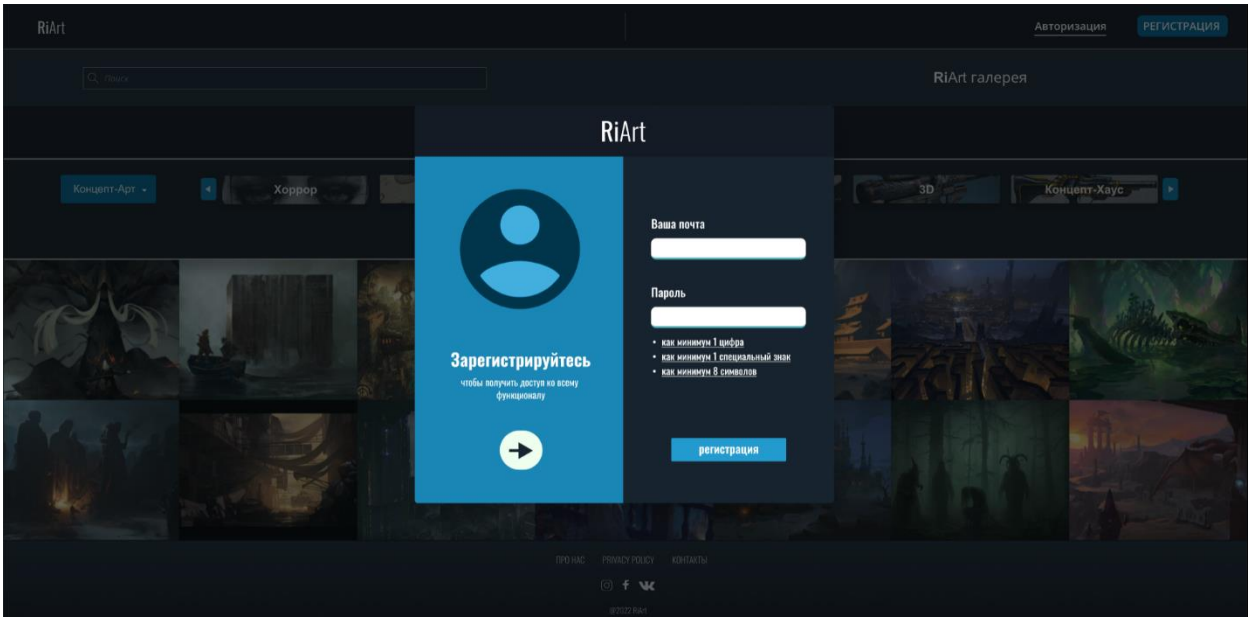


Рисунок 2.7 – вікно реєстрації користувача

Вікно авторизації також було вирішено зобразити по тому ж самому принципу як і вікно реєстрації, для забезпечення схожої структурності сайту для користувача. Вікно з формою авторизації зображено на рисунку 2.8.

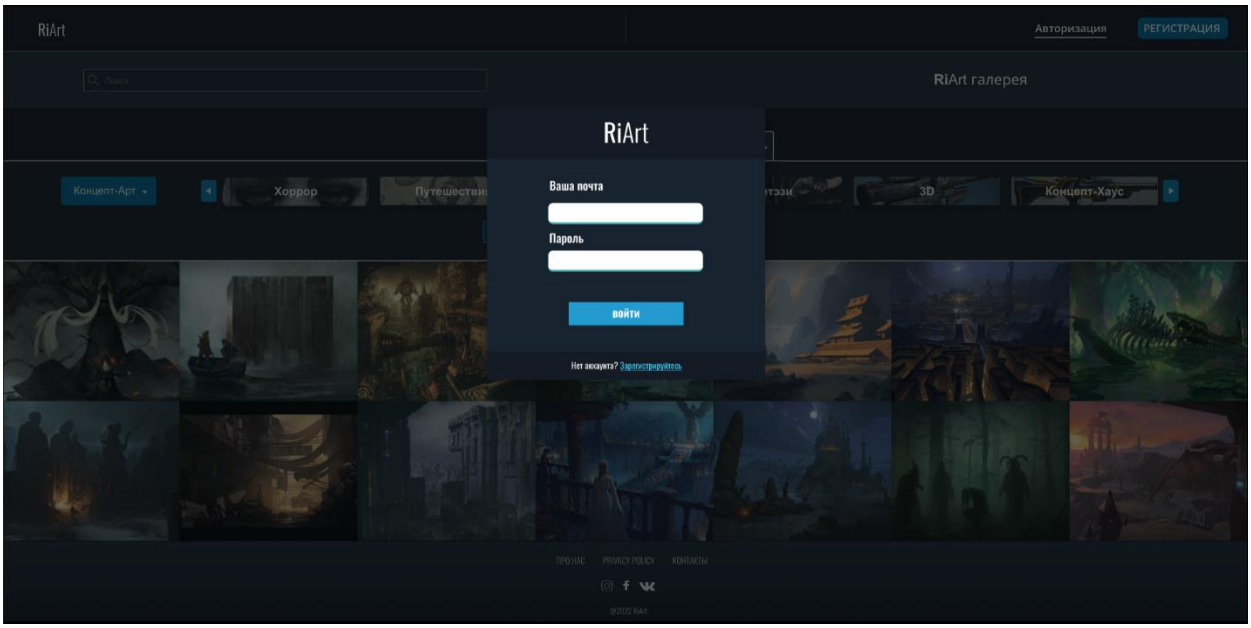


Рисунок 2.8 – вікно авторизації користувача

Для пошуку робіт на сайті було вирішено додати строку пошуку, розміщену у верхній лівій частині екрану. Було вирішено розмістити фільтри під полем пошуку, а для того щоб на вебсторінці сайту було менше непотрібних деталей, і більше місця для головної інформації на сайті, робіт у галереї, фільтри було розроблено як елемент який можна розкрити чи сховати вибравши кнопку головного фільтру (пошук за автором, пошук за роботами).

Приклад вебсторінки пошуку, з вибраними фільтрами зображено на рисунку 2.9.

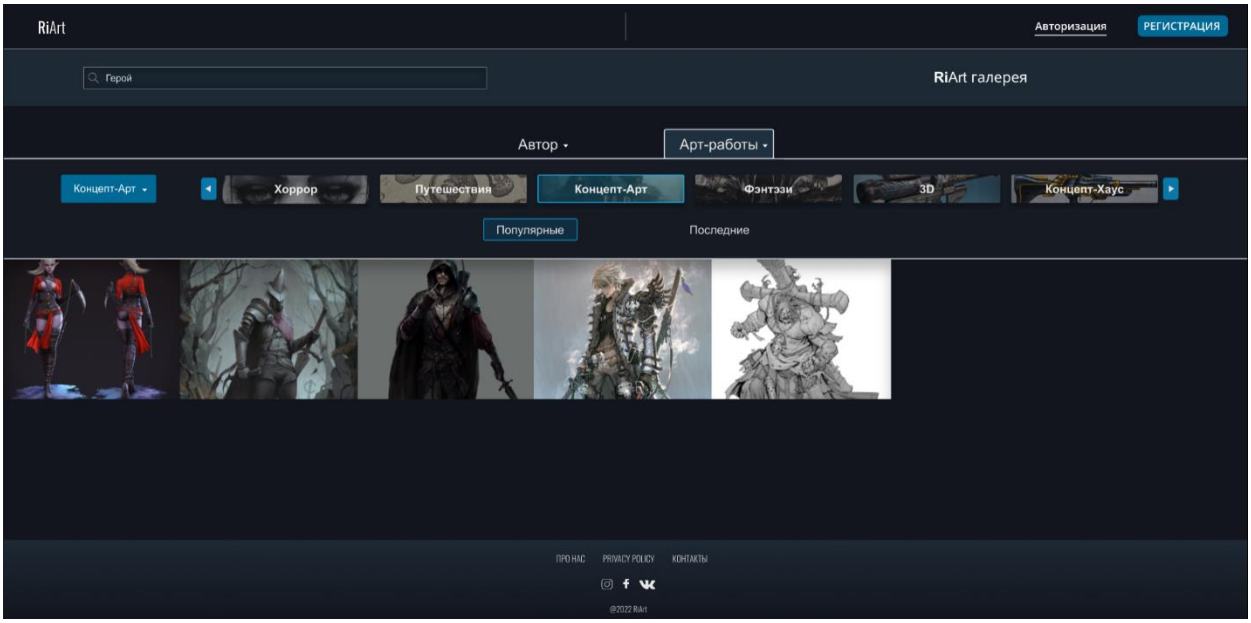


Рисунок 2.9 – приклад пошуку роботи на сайті

Після авторизації чи реєстрації, користувача автоматично перенаправить до його персонального профілю. Для навігації по персональному профілю було вирішено розмістити навігаційне меню під логіном користувача.

Персональний профіль користувача зображено на рисунку 2.10.

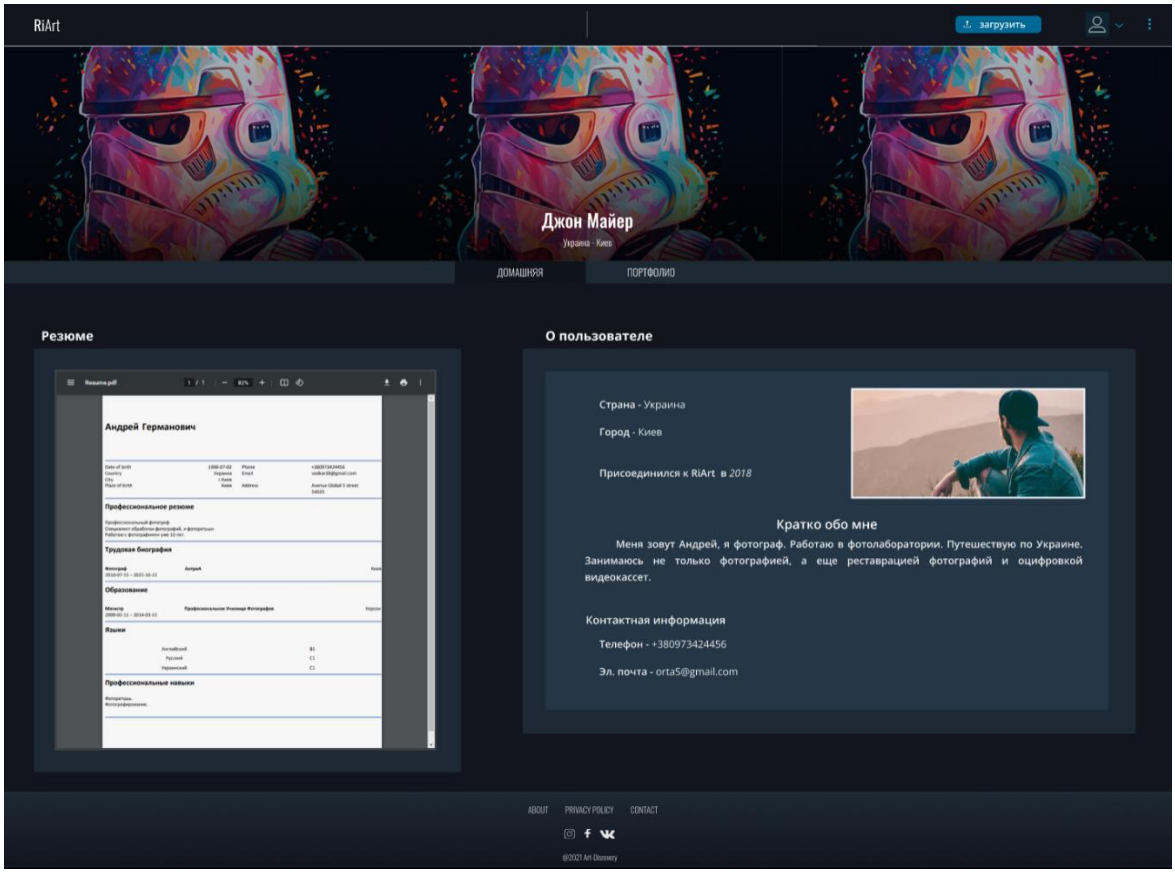


Рисунок 2.10 – головна сторінка персонального профілю користувача сайту

Портфоліо користувача, також знаходиться на його сторінці профілю. Портфоліо представлено на рисунку 2.11.

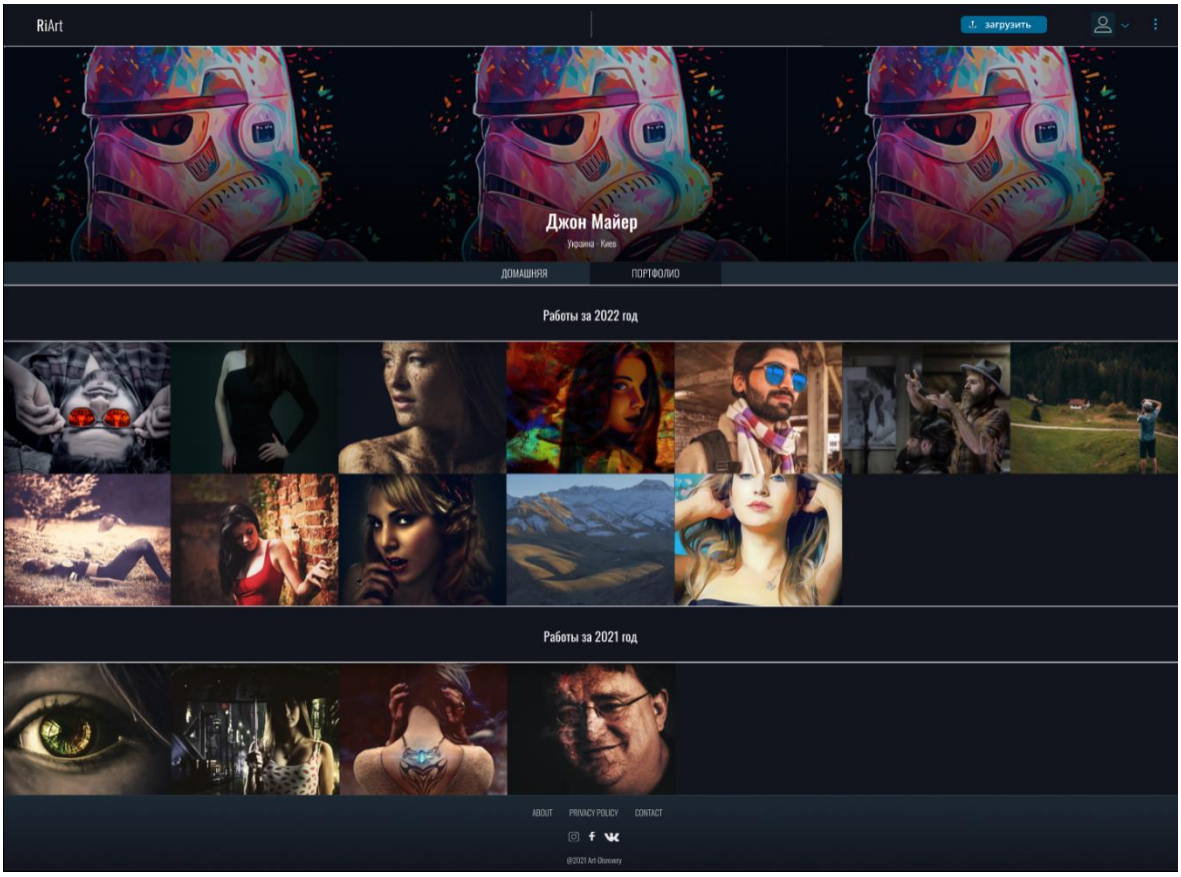


Рисунок 2.11 – портфоліо користувача

Для редагування персонального профілю користувача потрібно перейти до вебсторінки редагування облікового запису. Меню було вирішено

розмістити у голові сайту, бо меню навігації дуже важливий елемент для будь-якого вебсайту.

Меню для редагування профілю зображено на рисунку 2.12.

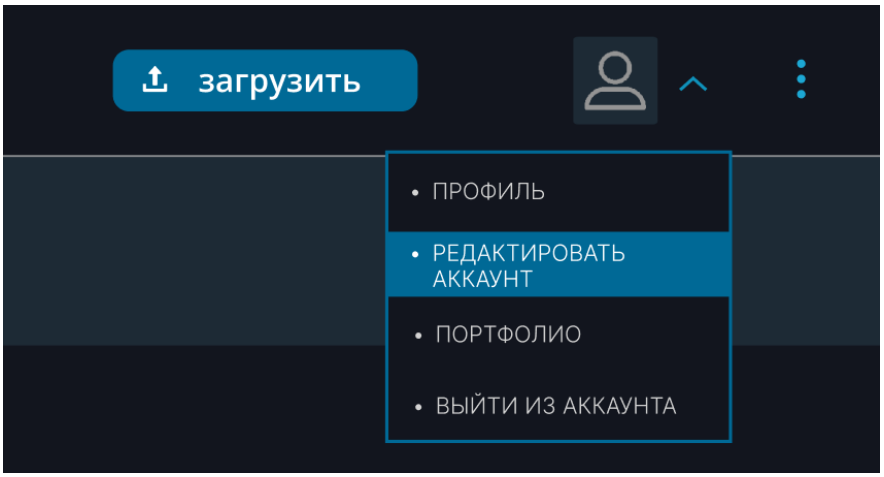


Рисунок 2.12 – меню для роботи з профілем користувача

Вебсторінка з редагуванням профілю користувача зображена на рисунку 2.13.

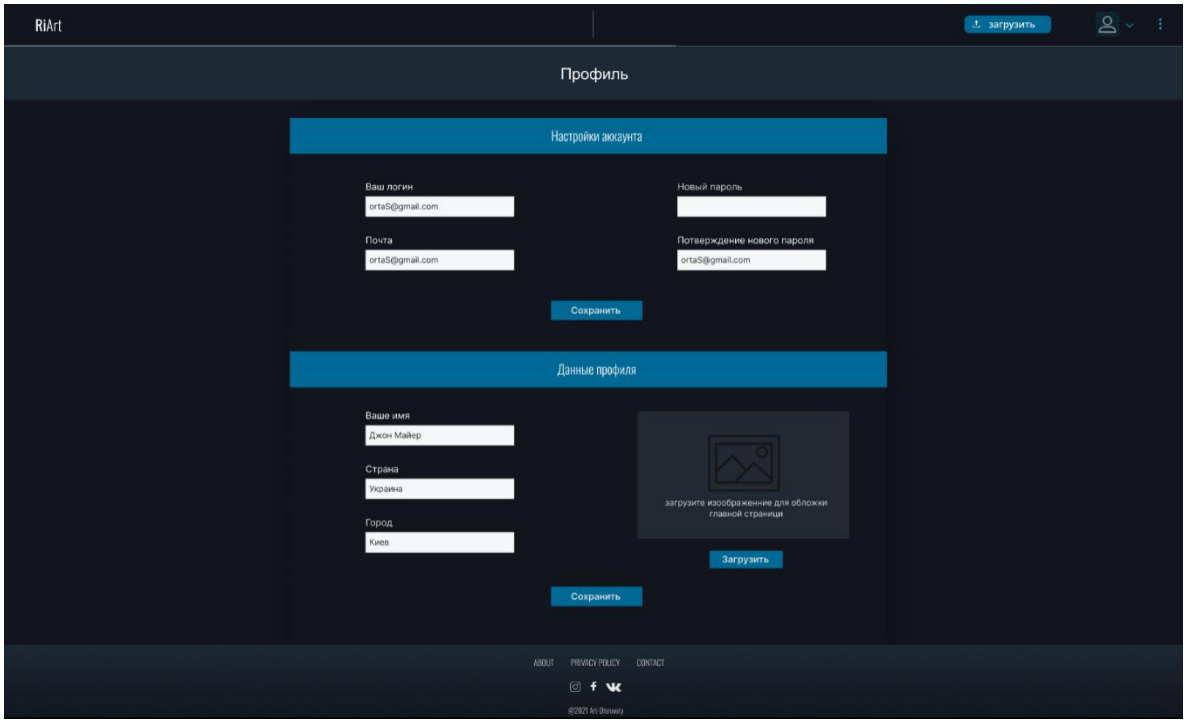


Рисунок 2.13 – вебсторінка редагування профілю користувача

Для перегляду робіт потрібно натиснути на роботу у галереї, що зацікавила користувача. Відкриється вебсторінка для перегляду роботи що зображена на рисунку 2.14.

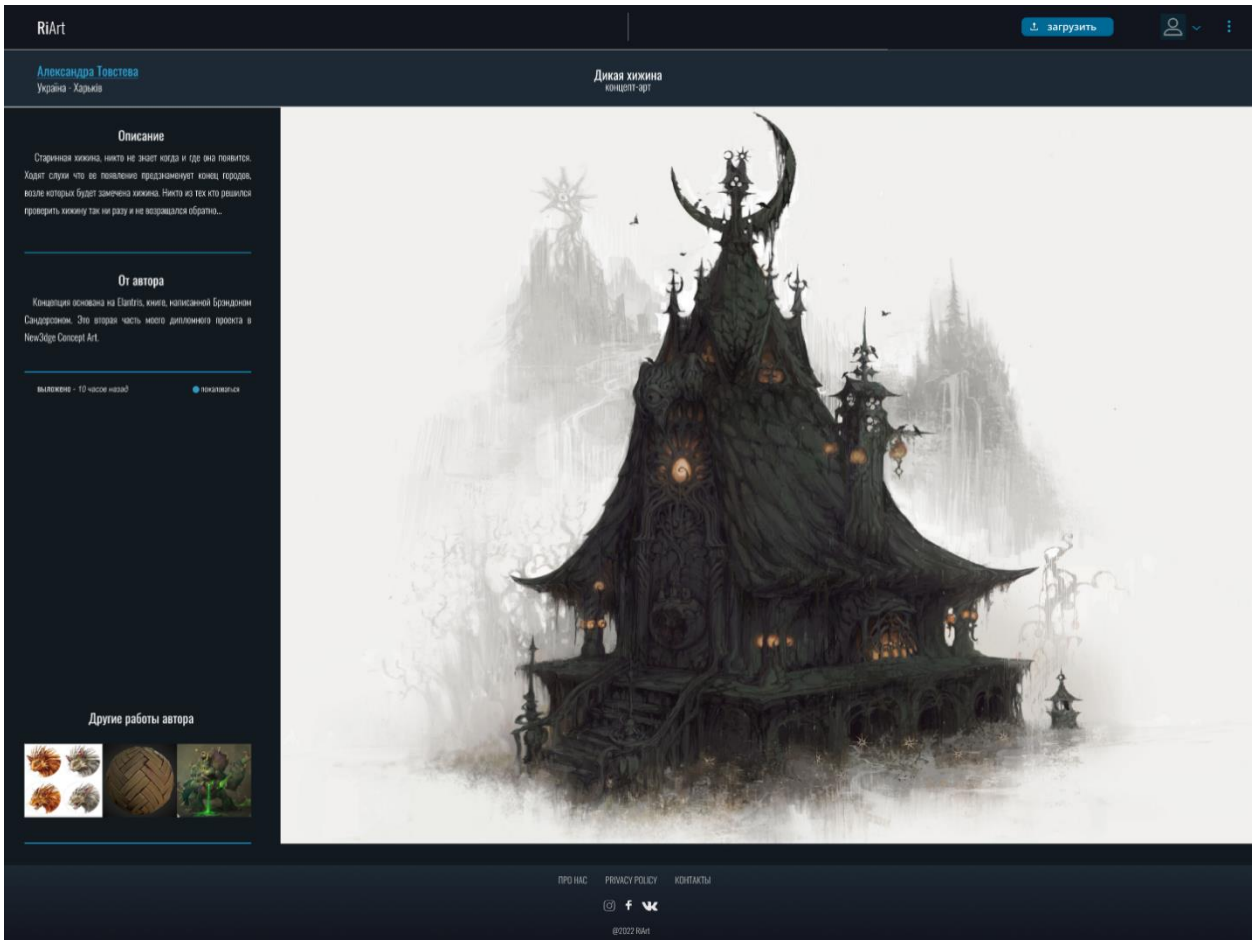


Рисунок 2.14 – вебсторінка для перегляду роботи

Для публікації роботи та автоматичного додання до портфоліо, користувачу буде представлена спеціальна вебсторінка для публікації яка зображена на рисунку 2.15.

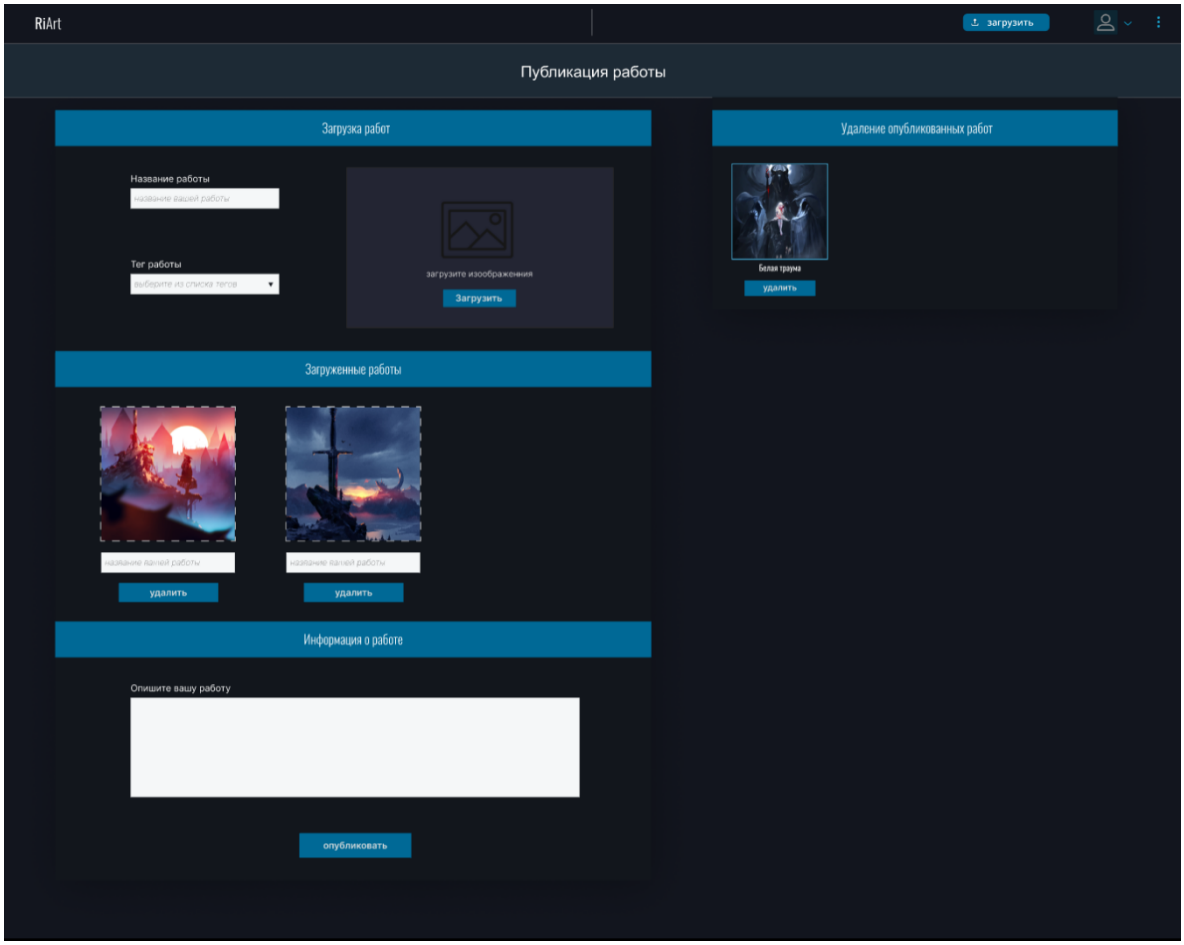


Рисунок 2.15 – вебсторінка для загрузки та публікації роботи

## 2.2 Технічний проєкт програмного забезпечення вебсайту для онлайн-галереї “RiArt”

На основі розробки ескізного проєкту, було створено технічний проєкт для програмного забезпечення вебсайту.

При розробці були створені статична й динамічна модель варіантів використання та логічна модель бази даних. Після аналізу класів програмного забезпечення представлено специфікацію класів.

### 2.2.1 Статична модель варіантів використання

Статична модель у вигляді діаграми класів представлена на рисунку 2.16.

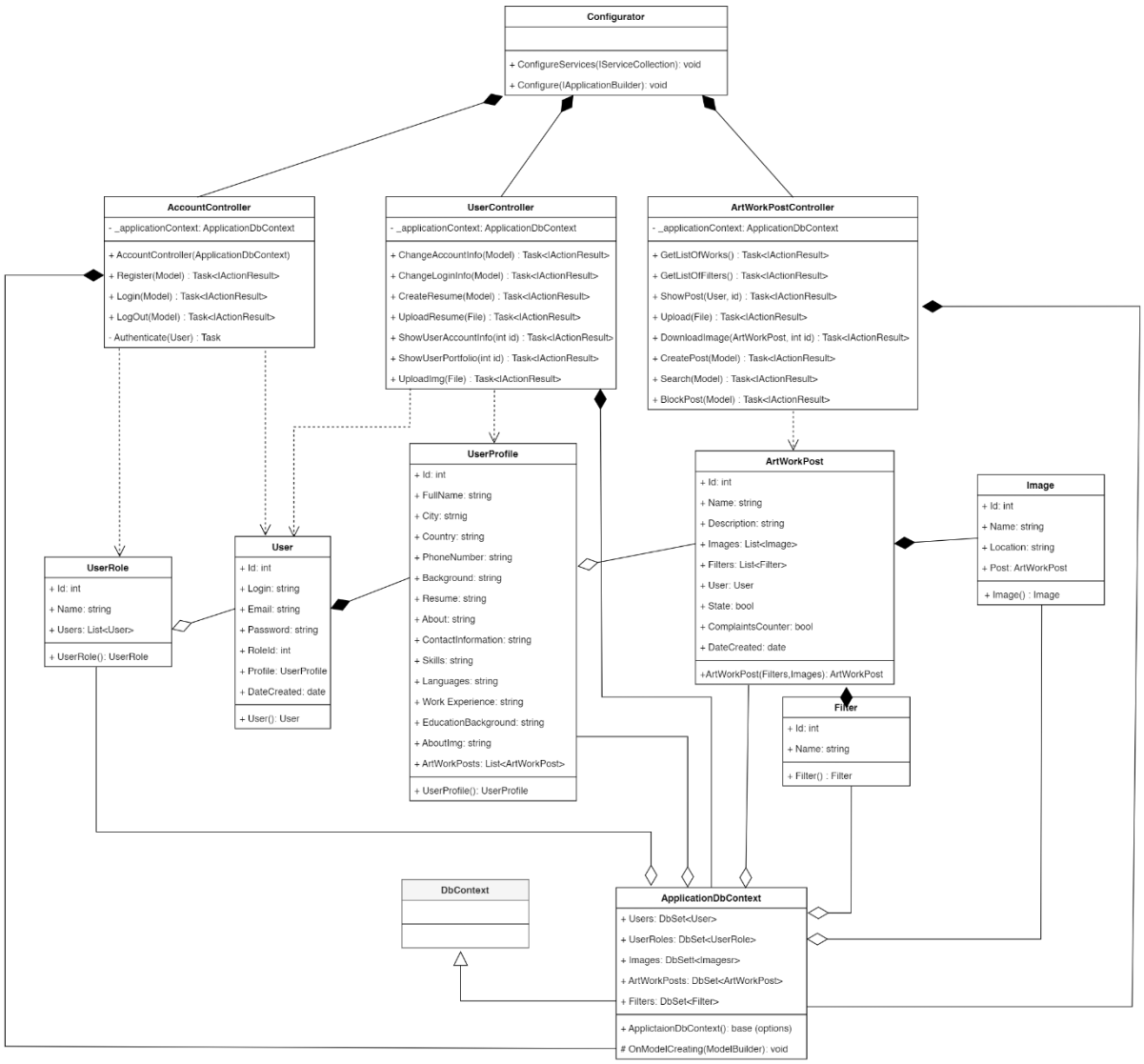


Рисунок 2.16 – Статична модель у вигляді діаграми класів

### 2.2.2 Динамічна модель варіантів використання

Після визначення сценаріїв варіантів використання та розробки діаграми класів було розроблено динамічну модель.

Діаграму послідовності для варіанту використання Авторизація, представлено на рисунку 2.17.

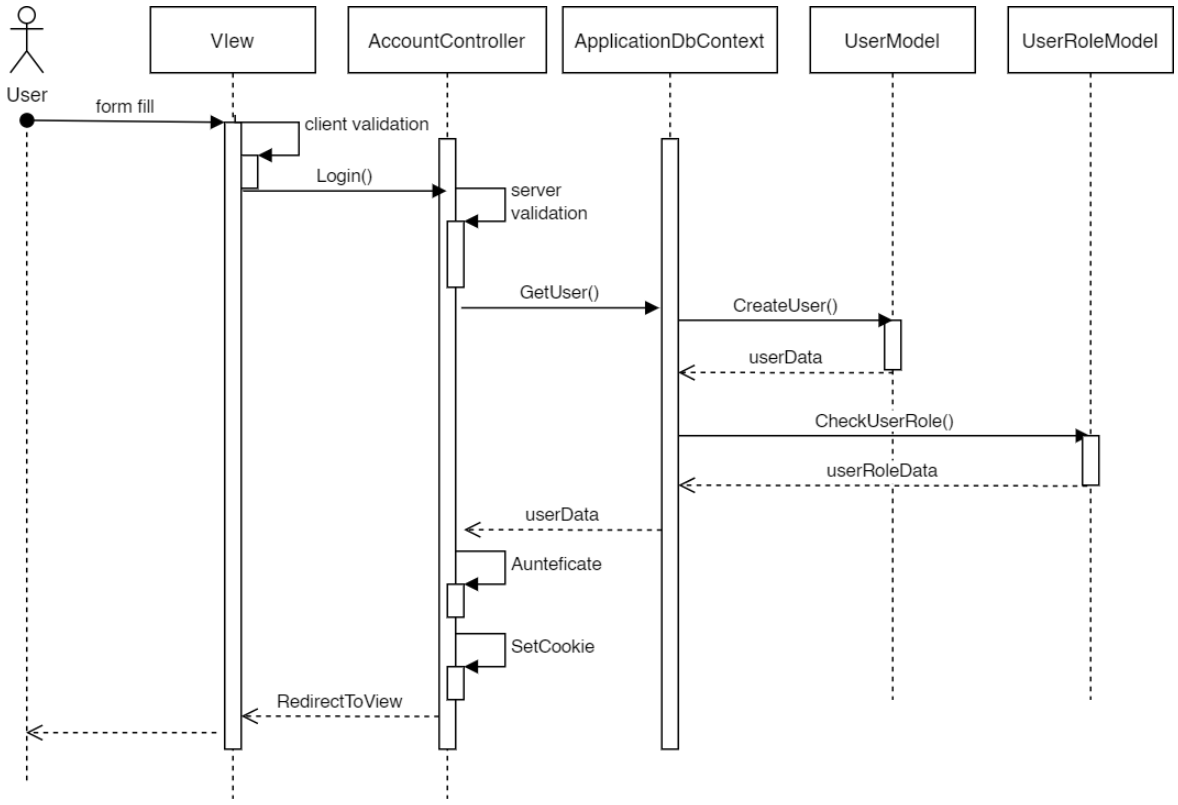


Рисунок 2.17 – Діаграма послідовності для Авторизації

### 2.2.3 Специфікація класів

Після аналізу класів у вигляді створення статичної та динамічної моделі, можемо проаналізувати склад класів, створивши їх специфікації.

Специфікація класів програмного забезпечення вебсайту для онлайн-галереї “RiArt” приведена нижче.

Назва: Configurator.

Відповідальність класу: Головний клас програмного забезпечення який відповідає за встановлення маршрутизації на вебсайті, та підключення сервісів.

Функції які надає клас:

- ConfigureServices() – під’єднати та налаштувати сервіси;
- Congifure() – налаштування та встановлення маршрутів.

Назва: UserController.

Відповідальність класу: Клас відповідає за логіку роботи з користувачами.

Атрибути:

- _applicationContext ().

Функції які надає клас:

- ChangeAccountInfo() – редагувати данні профілю;
- ChangeLoginInfo() – редагувати дані для входу;
- CreateResume() - створити;
- UploadResume() – загрузити резюме;
- ShowUserAccountInfo() – відобразити інформацію про профіль користувача;
- ShowUserPortfolio() – відобразити портфоліо користувача;
- UploadImg() – загрузити зображення до профілю користувача.

Назва: AccountHandler.

Відповідальність класу: Клас відповідає за авторизацію та реєстрацію користувачів.

Атрибути:

- _applicationContext ().

Функції які надає клас:

- AccountController() – конструктор класу ;
- Register() – реєстрації користувача;

- Login() – авторизації користувача;
- LogOut() – вихід з облікового запису користувача;
- Authenticate() – аутентифікація користувача.

Назва:ArtWorkPost.

Відповідальність класу: Клас відповідає за логіку роботи з постами.

Атрибути:

- _applicationContext.

Функції які надає клас:

- GetListOfWorks() – отримати список постів;
- GetLsitOfFilters() – отримати список фільтрів;
- ShowPost() – відобразити пост;
- Upload () – загрузити зображення роботи;
- DownloadImage() – скачати зображення роботи;
- CreatePost() – опублікувати пост;
- Search() – знайти пости за запитом;
- BlockPost() – заблокувати пост користувача.

Назва: ApplicationDbContext.

Відповідальність класу: Клас є допоміжним для реалізації міграції, а також відповідає за роботу з базою даних.

Атрибути:

- Users;
- UserRoles;
- Images;
- ArtWorkPosts;
- Filters.

Функції які надає клас:

- ApplicationDbContext() – конструктор класу;
- OnModelCreating() – перевизначений метод суперкласу.

Назва: User.

Відповідальність класу: Клас є моделлю зареєстрованого користувача, яка представляє таблицю в базі даних.

Атрибути:

- Id;
- Login;
- Email;
- RoleId;
- DateCreated;
- Profile.

Функції які надає клас:

- User() - конструктор класу.

Назва:UserRoles.

Відповідальність класу: Клас є моделлю ролей користувачів, яка представляє таблицю в базі даних.

Атрибути:

- Id;
- Name;
- Users.

Функції які надає клас:

- User() - конструктор класу.

Назва: UserProfile.

Відповідальність класу: Клас є моделлю профілю де зберігається уся додаткова інформація користувача, яка представляє таблицю в базі даних.

Атрибути:

- Id;
- FullName;
- City;

- Country;
- PhoneNumber;
- Background;
- Resume;
- About;
- ContactInfromation;
- Skills;
- Languages;
- WorkExperience;
- EducationBackground;
- AboutImg;
- ArtWorkPosts.

Функції які надає клас:

- UserProfile() - конструктор класу.

Назва: ArtWorkPost.

Відповідальність класу: Клас є моделлю публікації, яка представляє таблицю в базі даних.

Атрибути:

- Id;
- Name;
- Description;
- Images;
- Filters;
- User;
- State;
- DateCreated;
- ComplaintsCounter.

Функції які надає клас:

- ArtWorkPost() - конструктор класу.

Назва: Filter.

Відповідальність класу: Клас є моделлю фільтрів, яка представляє таблицю в базі даних.

Атрибути:

- Id;
- Name.

Функції які надає клас:

- Filter() - конструктор класу.

Назва: Image.

Відповідальність класу: Клас є моделлю художніх робіт, та представляє таблицю в базі даних.

Атрибути:

- Id;
- Name;
- Location;
- Post.

Функції які надає клас:

- Image() - конструктор класу.

### 2.2.4 Логічна модель даних

Для відображення структури бази даних незалежно від використовуваної СУБД було побудовано Логічну модель.

Модель представлено на рисунку 2.18.

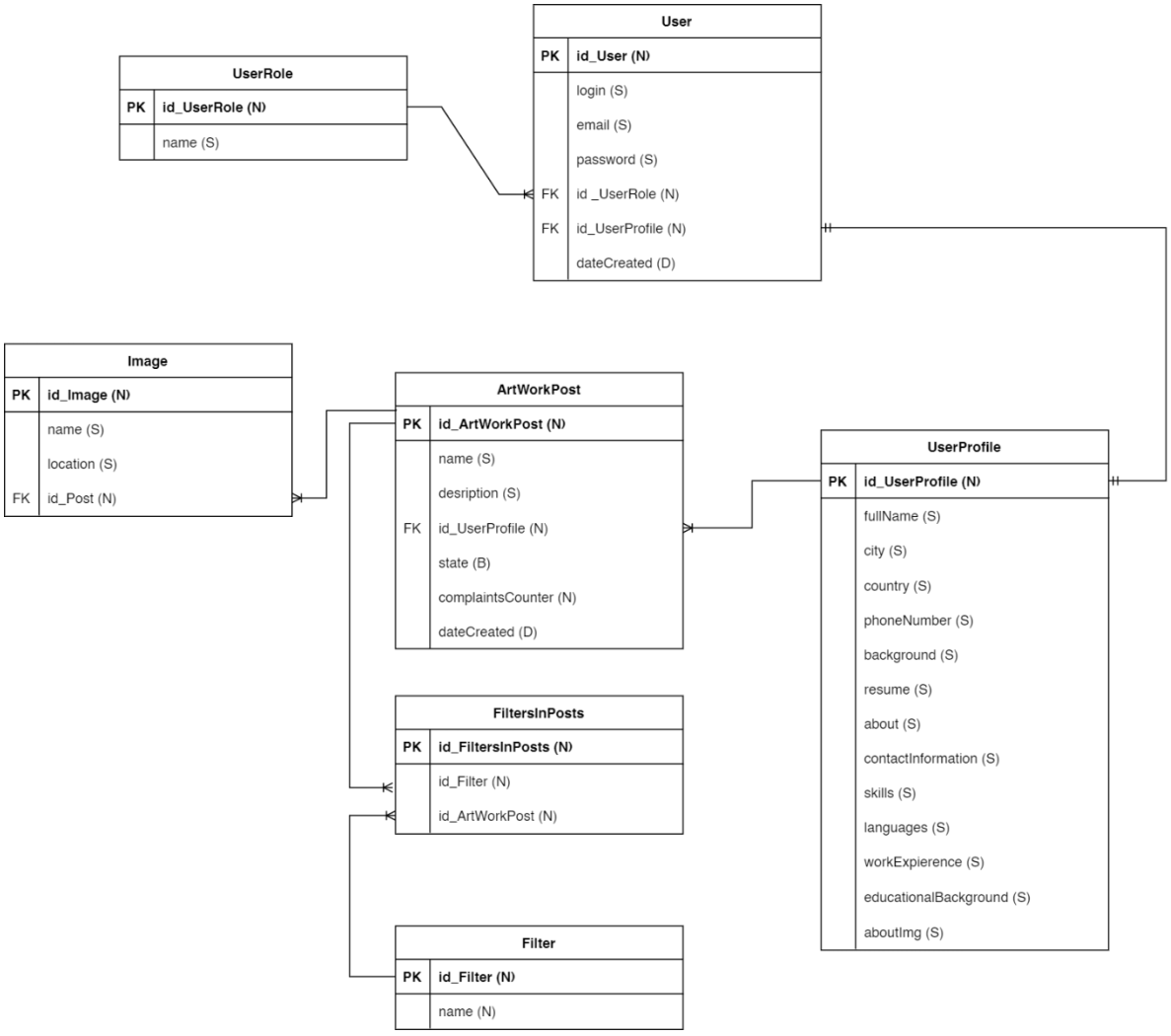


Рисунок 2.18 – Логічна модель бази даних

## 2.3 Робочий проєкт програмного забезпечення програмного забезпечення вебсайту для онлайн-галереї “RiArt”

Після аналізу вимог до програмного забезпечення, були вибрані конкретні інструменти для реалізації вебсайту. Підрозділ присвячений опису переваг та недоліків вибраних мов програмування та СУБД, проведенню тестів варіантів використання та випробовування роботи програмного забезпечення вебсайту для онлайн-галереї “RiArt”.

### 2.3.1 Вибір мови програмування

При розробці будь-якого програмного забезпечення виникає задача вибору мови програмування та технологій для розробки. Для розробки програмного забезпечення вебсайтів, існують різні способи та архітектури розробки.

Архітектурою розробки було обрано MVC.

Більшість мов програмування має достатньо технологій та функціональних можливостей для створення програмного забезпечення на MVC архітектурі. Для даного проєкту потрібно вибрати мову програмування з підтримкою спрощеної роботи с СУБД для скорочення часу потрібного на розробку, а саме, мову яка має фреймворки з ORM-технологією. Технологією яка надає можливість представити дані таблиці в БД як об’єкт та працювати з ними як з об’єктом класу. А також на вибір впливає можливість реалізації шаблону MVC.

В результаті було обрано об’єктноорієнтовану мову програмування C#. На основі цієї мови програмування побудовано фреймворк ASP.NET core, який створений за допомогою платформи .NET.

MVC у ASP.Net виступає спеціальним фреймворком та передбачає застосування моделей (Model), представлень (View), та Контролерів (Controller), як головних компонентів.

Модель – потрібна для опису даних які будуть використовуватися в програмному забезпеченні та їх логіку. Часто моделі являють собою звичайний клас з атрибутами та їх валідацією, і не можуть мати логіку взаємодії з користувачем через користувацький інтерфейс. Працювати з моделями найбільш ефективно якраз з використанням ORM фреймворків для бази даних.

Представлення – відповідає за користувацький інтерфейс, через який користувач вже може взаємодіяти з програмним забезпеченням. Представлення не повинне мати майже ніякої логіки для управління даними, окрім їх відображення.

Контролер – це головний компонент у схемі MVC. Він відповідає за зв'язок між програмним забезпеченням, користувачем, моделлю та представленням. Отже, контролер містить усі логіку, яка потрібна для обробки користувацьких запитів. Зазвичай він дає можливість обробки даних, та відображення результату у вигляду представлення, яке заповнюється даними отриманими з моделей.

Головною особливістю фреймворку ASP.NET core є простота побудови програмного забезпечення для вебсайтів, відкритий код, а також підтримка MVC, у вигляді вбудованого до стандартних пакетів сервісу. Перевагою ж мови C# є швидкість. Загальномовне виконуюче середовище C# та .NET, CLR (Common Language Runtime), що є віртуальною машиною, є у кожній операційній системі Windows, які, в свою чергу, є найбільш популярними. Для того щоб код зміг виконатися CLR, код написаний на C# повинен спочатку бути скомпільований [5].

Компіляція розділяється на 2 етапи:

1). Компіляція вихідного коду Microsoft Intermediate Language (IL);

2). Компіляція IL у специфічний для платформи код за допомогою CLR.

Фреймворки C# використовуються для побудови програмних рішень націлених на Windows платформи, однак з виходом фреймворку .Net core, з'явилась багатоплатформність.

Для роботи з C# існує 2 найбільш популярних IDE (середовища розробки) це – Microsoft Visual Studio та JetBrains Rider. Зазвичай різниці в вибору тієї чи іншої IDE немає, окрім їх доступності, та деяких додаткових функцій, тому було обрано середовище розробки Microsoft Visual Studio 2019, яке є більш доступним.

Основні переваги у використанні мови програмування C# та фреймворку Asp.Net core:

- швидкість роботи;
- простота розробки;
- наявність популярного ORM фреймворку для роботи с СУБД Entity Framework core;
- можливість створення програмного забезпечення вебсайту за архітектурами REST, WebApi або MVC;
- можливість використовувати мову C# прямо у html файлах розмітки за допомогою технології Razor Pages;
- можливість використовувати технологію LINQ запитів, для спрощення рутинної роботи та значного підвищення швидкості розробки.

Web API – це спосіб створення програмного забезпечення в Asp.Net core, який спеціально створений для роботи в стилі REST архітектури. Такий спосіб розробки дозволяє розробити Backend частину на Asp.Net core, а Frontend писати за допомогою Single Page Application, які використовують фреймворки мови програмування javascript, наприклад jQuery.

Razor Pages – спеціальний формат html файлів, який дозволяє використовувати мову програмування C# напряму у файлі.

Asp.Net core буде виступати у ролі Backend частини сайту. Для Frontend частини було обрано html, css, js та фреймворк jQuery, що є стандартним набором інструментів при розробці на Asp.net core.

### 2.3.2 Вибір системи управління базами даних

Для роботи з БД використовують СУБД. Існує багато різноманітних СУБД та БД, і цей підрозділ присвячений аргументуванню вибору СУБД для програмного забезпечення вебсайту онлайн-галереї “RiArt”.

Під реляційними базами даних мають на увазі ті, які використовують мову запитів SQL. Дані в таких базах даних зберігаються у вигляді таблиць й рядків у них, які можливо поєднувати між собою, створюючи зв’язки, за допомогою ключів. Це дозволяє додати унікальності для елемента в таблиці.

Нереляційні бази даних (NoSQL) не мають такої структурованості як у реляційних, а інформація в них зберігається у спеціальних документах формату JSON, як пара ключ-значення.

Обидві бази даних мають свої недоліки та переваги, однак, для роботи з обраною мовою програмування та технологіями, найважливішим для розроблюваного програмного забезпечення є підтримка вибраного фреймворку Entity Framework та робота з структурованими даними з чіткими вимогами.

Проаналізувавши реляційні та нереляційні бази даних, було вирішено обрати реляційну базу даних.

Для реляційних баз даних існують різні СУБД. Найпопулярнішими з них є MySQL, PostgreSQL та Microsoft SQL Server. Обраний фреймворк для роботи з СУБД Entity Framework дозволяє працювати з будь-якими СУБД, тому вибір повинен бути обумовленим на можливостях які надають ці СУБД.

PostgreSQL краще застосовувати у великих проєктах, через дуже широкий функціонал з високим рівнем захисту. Використання PostgreSQL не є доцільним вибором у цьому випадку, бо більша частина роботи буде виконана фреймворком, тому велика кількість вбудованих у СУБД функцій, не потрібна. Також через занадто великий функціонал й розташуванням на

одному сервері, PostgreSQL не виділяється швидкістю, якщо порівнювати з іншими приведеними СУБД.

MySQL – це досить проста та швидкісна СУБД. В ній реалізовані найбільш потрібні для розробки програмних забезпечень вебсайтів функції. Вона містить вже реалізовані та вбудовані функції захисту, працює з багатьма мовами програмування та технологіями. Однак, навіть маючи увесь функціонал призначений для веброзробки, вона повноцінно не підтримує технологію LINQ запитів у фреймворку .NET, яка використовується в розроблюваному програмному забезпеченні вебсайту, без додаткового встановлення сторонніх інструментів, що підвищить час розробки.

Microsoft SQL Server містить майже усі визначенні плюси MySQL СУБД, має простий й зрозумілий інтерфейс, може бути розміщена на декількох серверах й підтримує LINQ запити. Одним з головних мінусів це вартість її розміщення на хостингу через невеликий вибір доступних сервісів. Однак, є безоплатні варіанти розміщення, з умовою використання лише основного функціоналу представленого у СУБД.

Сайти хостинги – це сайти які надають сервер з публічним доменом для розміщення програмного забезпечення вебсайтів, з виходом до інтернету.

Тому провівши аналіз існуючих СУБД та БД, було вирішено використовувати у розробці програмного забезпечення вебсайту онлайн-галереї “RiArt” СУБД Microsoft SQL Server.

### 2.3.3 Фізична модель даних

Фізичну модель бази даних наведено у виді таблиць (таблиці 2.5 – 2.11).

Таблиця 2.5 – Структура таблиці UserRole

| **Номер** | **Назва поля** | **Ключ** | **Тип Даних** |
|---|---|---|---|
| 1 | id_userRole | PK | TINYINT |
| 2 | name | | NCHAR(21) |

Таблиця 2.6 – Структура таблиці User

| Номер | Назва поля | Ключ | Тип Даних |
|---|---|---|---|
| 1 | id_user | PK | INT |
| 2 | login | | NCHAR(30) |
| 3 | email | | NCHAR(36) |
| 4 | dateCreated | | DATETIME |
| 5 | id_userRole | FK | TINYINT |
| 6 | id_userProfile | FK | INT |

Таблиця 2.7 – Структура таблиці UserProfile

| Номер | Назва поля | Ключ | Тип Даних |
|---|---|---|---|
| 1 | id_userProfile | PK | INT |
| 2 | fullName | | NCHAR(40) |
| 3 | city | | NCHAR(30) |
| 4 | country | | NCHAR(30) |
| 5 | phoneNumber | | INT |
| 6 | background | | NCHAR(100) |
| 7 | resume | | NCHAR(100) |
| 8 | about | | NCHAR(450) |
| 9 | contactInformation | | NCHAR(100) |
| 10 | skills | | NCHAR(300) |
| 11 | languages | | NCHAR(200) |
| 12 | workExpierence | | NCHAR(350) |
| 13 | educationalBackground | | NCHAR(350) |
| 14 | aboutImg | | NCHAR(100) |

Таблиця 2.8 – Структура таблиці ArtWorkPost

| Номер | Назва поля | Ключ | Тип Даних |
|---|---|---|---|
| 1 | id_artWorkPost | PK | INT |
| 2 | name | | NCHAR(50) |
| 3 | description | | NCHAR(400) |
| 4 | id_userProfile | FK | INT |
| 5 | state | | BIT |
| 6 | complaintsCounter | | INT |
| 7 | dateCreated | | DATETIME |

Таблиця 2.9 – Структура таблиці FiltersInPosts

| Номер | Назва поля | Ключ | Тип Даних |
|---|---|---|---|
| 1 | id_filtersInPosts | PK | INT |
| 2 | id_filter | FK | INT |
| 3 | id_artWorkPost | FK | INT |

Таблиця 2.10 – Структура таблиці Filter

| Номер | Назва поля | Ключ | Тип Даних |
|---|---|---|---|
| 1 | id_filter | PK | INT |
| 2 | name | | NCHAR(20) |

Таблиця 2.11 – Структура таблиці Image

| Номер | Назва поля | Ключ | Тип Даних |
|---|---|---|---|
| 1 | id_image | PK | INT |
| 2 | name | | NCHAR(50) |
| 3 | location | | NCHAR(100) |
| 4 | id_post | FK | INT |

### 2.3.4 Діаграма розміщення

Діаграма розміщення є фізичною моделлю проєктування для представлення архітектури програмного забезпечення. За допомогою цієї діаграми відображають програмні чи апаратні вузли виконання, з проміжним програмним забезпеченням що з'єднує їх, тобто стає можливим розуміння того як система буде фізично розгорнута.

Діаграма наведена на рисунку 2.19.

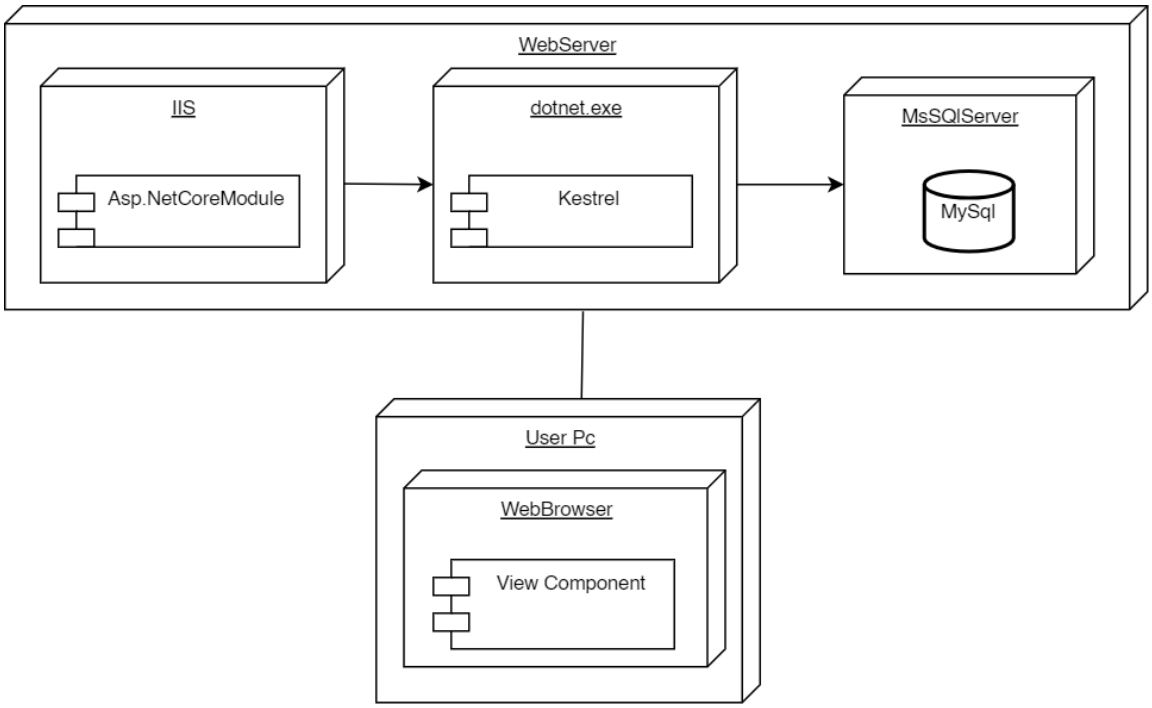


Рисунок 2.19 – Діаграма розміщення

### 2.3.4 Тестування варіантів використання

Проведені тестування були виконані за принципом чорної скриньки. Для тестування були обранні основні варіанти використання. Метою кожного

тестування є перевірка на помилки в роботі програмного забезпечення вебсайту онлайн-галереї “RiArt”.

Тестування варіантів використання.

У ході першого тестування був протестований варіант використання “Авторизація”. Проведених тестів 3.

Вхідними даними є:

- Введена електрона пошта гостем;
- Введений пароль гостем.

Тести:

1) Авторизація гостя як клієнта.

Для того, щоб гість міг авторизуватися, йому потрібно на головній сторінці сайту, у правій верхній частині, натиснути кнопку “Авторизація”, та заповнити форму вхідними даними для входу до свого профілю. Натиснувши “Увійти” очікуваним результатом є те, що програмне забезпечення повинно перевірити введені гостем дані (рисунок 2.20) та авторизувати гостя як клієнта, якщо введені данні є коректними. Результат тесту представлений на рисунку 2.21.

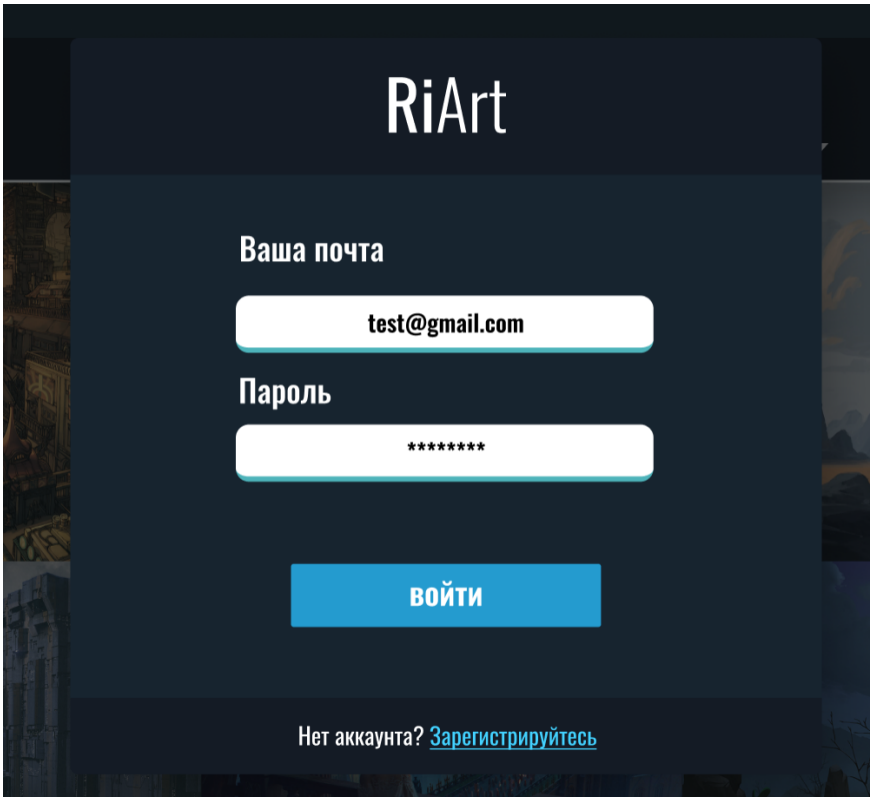


Рисунок 2.20 - Введення даних користувачем

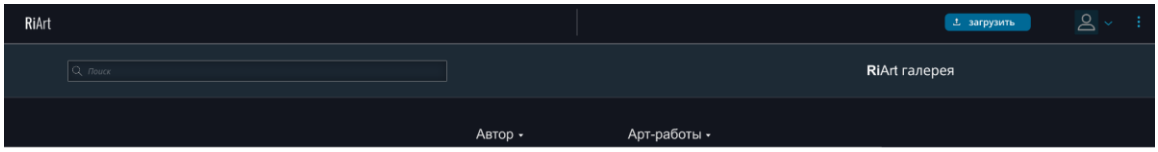


Рисунок 2.21 - Результат тесту авторизації як клієнту

2) Авторизація гостя як адміністратора.

Для того, щоб гість міг авторизуватися, йому потрібно на головній сторінці сайту, у правій верхній частині, натиснути кнопку “Авторизація”, та заповнити форму вхідними даними для входу до свого профілю. Натиснувши “Увійти” очікуваним результатом є те, що програмне забезпечення повинно перевірити введені гостем дані (рисунок 2.22) та авторизувати гостя як адміністратора, якщо введені данні є коректними.

Результат тесту представлений на рисунку 2.23.

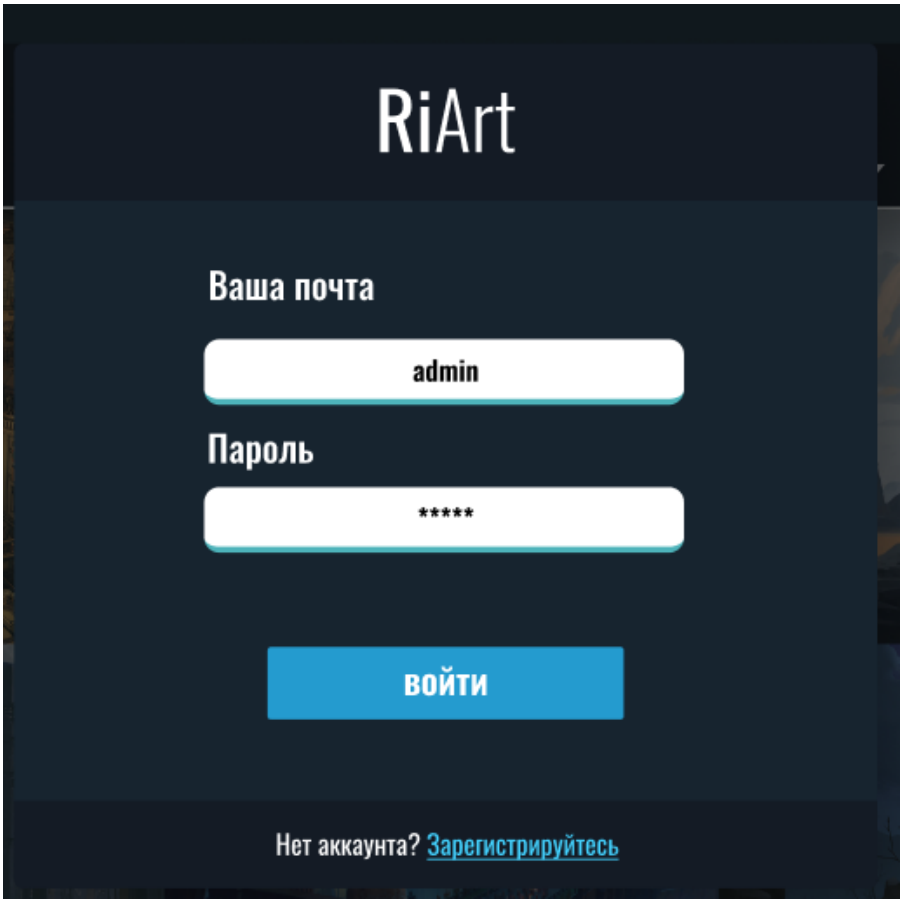


Рисунок 2.22 - Введення даних користувачем

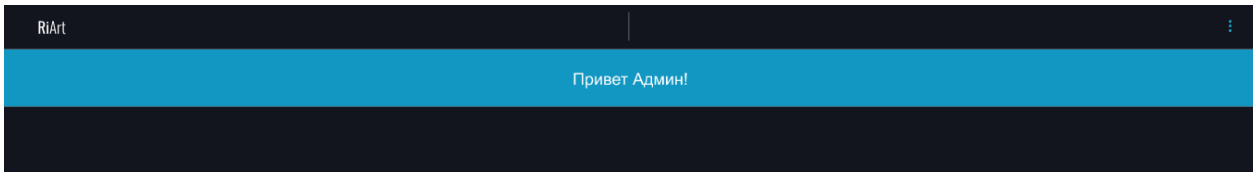


Рисунок 2.23 - Результат тесту авторизації як адміністратора

3) Авторизація з помилкою у процесі

Якщо гість ввів некоректні вхідні дані, чи не заповнив усі елементи у формі, після натискання кнопки “Увійти” очікуваним результатом повинна бути помилка видана програмним забезпеченням з вказаною причиною її виникнення. На рисунках 2.24 та 2.25 зображені результати тестування.

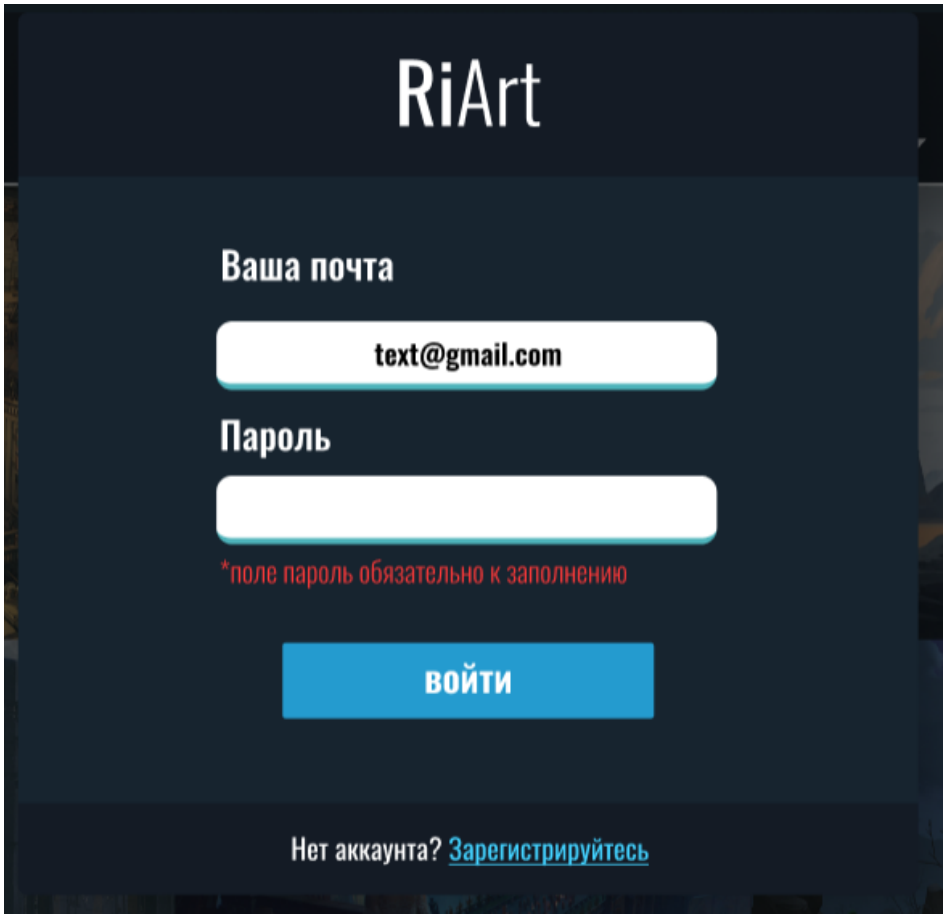


Рисунок 2.24 - Перший варіант результату тесту авторизації з помилкою

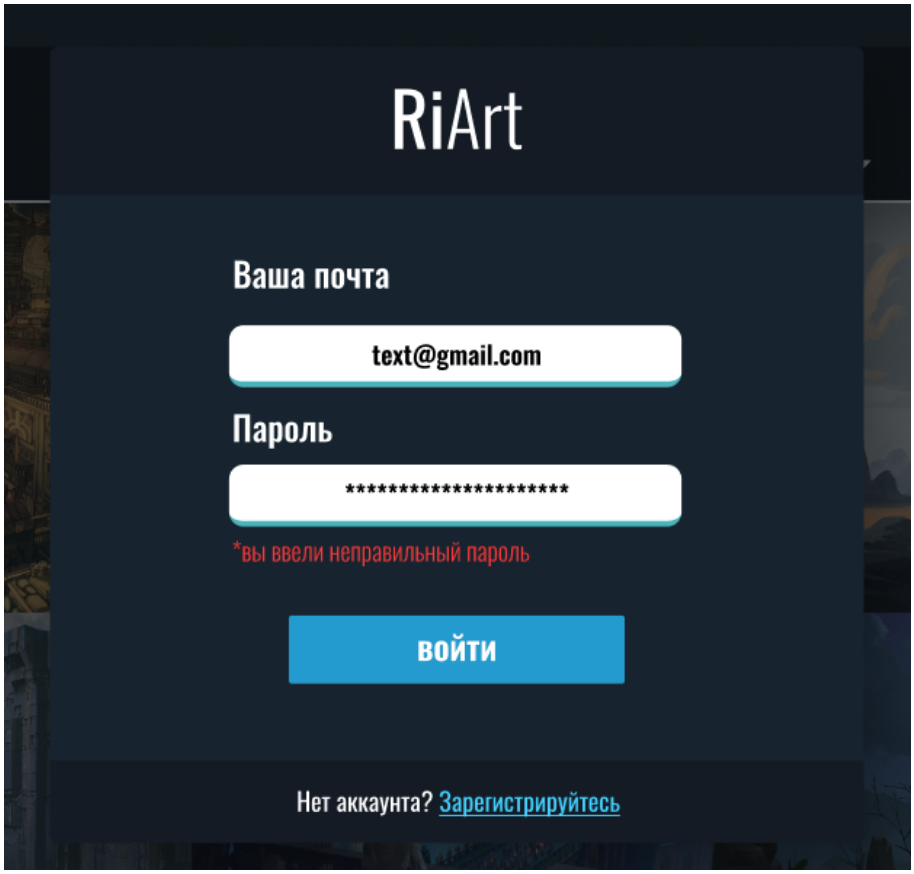


Рисунок 2.25 - Другий варіант результату тесту авторизації з помилкою

В ході тестування Авторизації помилок у роботі програмного забезпечення не було виявлено.

У ході другого тестування був протестований варіант використання “Пошук робіт”. Проведених тестів 2.

Тести:

1) Пошук робіт за фільтрами

Для ініціалізації пошуку за фільтрами, користувачу потрібно вибрати один із типів фільтрів натиснувши на “Автор” чи “Арт Роботи”, та обрати потрібний фільтр зі списку, чи скористатись списком з горизонтальною смугою прокрутки. Після виборі фільтра ввести до строки пошуку потрібні вхідні дані для пошуку, та натиснути на клавішу Enter чи на значок лупи біля строки пошуку.

Вхідними даними є:

- Дані уведені до строки пошуку;

- Обранні фільтри пошуку із представлених на вебсторінці.

Очікуваним результатом на етапі вибору типу фільтра є відкриття панелі с списком фільтрів для заданого типу фільтрування (рисунок 2.26).

Очікуваним результатом на етапі вибору фільтра є виділення обраного фільтра, та автоматичне виділення аналогічного фільтра в альтернативному способі вибору (рисунок 2.27).

Очікуваним результатом на етапі ініціювання пошуку за запитом у пошуковій строчці є представлення користувачу результату пошуку у вигляді знайдених публікацій робіт, чи повідомлення про неможливість знайти публікації за даними критеріями пошуку (рисунок 2.28 та рисунок 2.29).

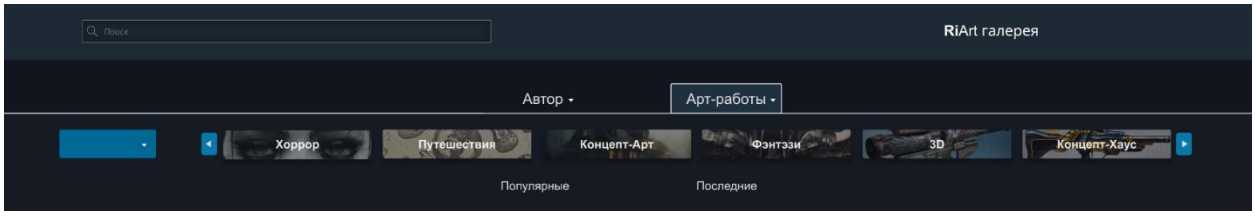


Рисунок 2.26 - Тестування етапу вибору типу фільтра

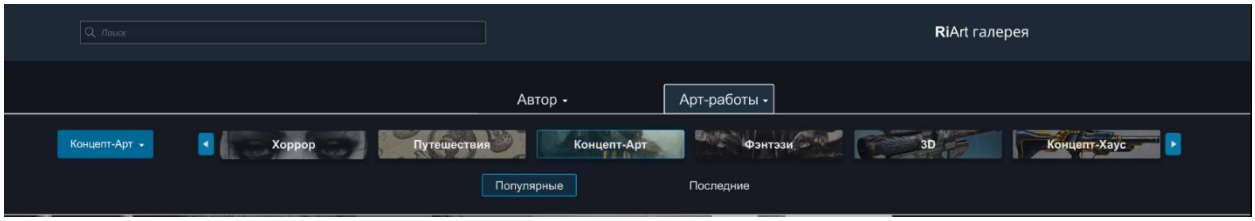


Рисунок 2.27 - Тестування етапу вибору фільтра

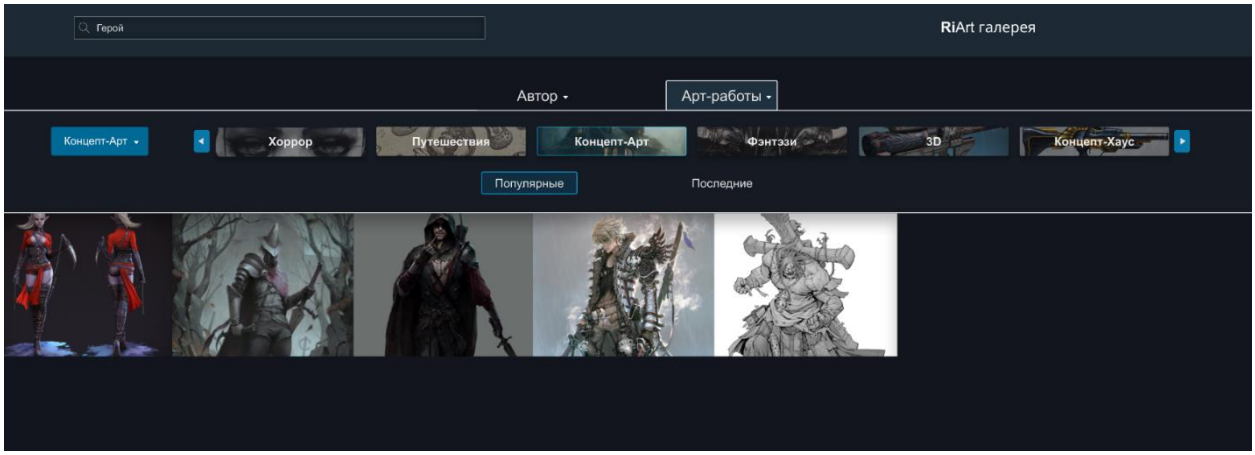


Рисунок 2.28 - Тестування результату пошуку

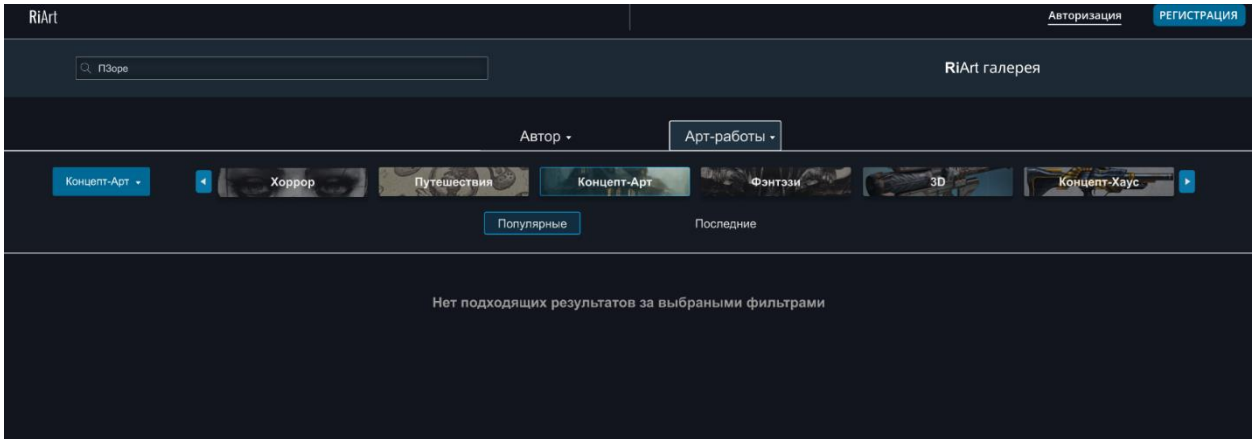


Рисунок 2.29 - Тестування результату пошуку

2) Пошук робіт без фільтра

Для ініціалізації пошуку робіт без фільтрів, користувачу потрібно ввести потрібні дані для пошуку, до пошукової строки, та натиснути на Enter чи на значок лупи біля строки пошуку.

На цьому етапі, програмне забезпечення вебсайту повинне надати результати пошуку лише за вхідними даними пошукової строки.

Результат тестування зображено на рисунку 2.30.

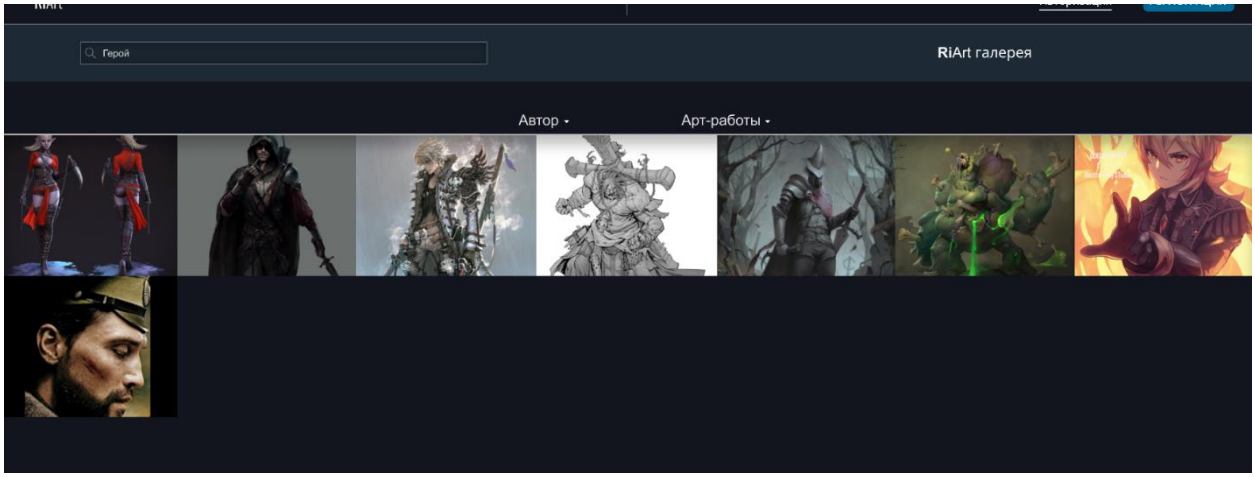


Рисунок 2.30 - Результат тестування пошуку

В ході тестування пошуку помилок у роботі програмного забезпечення не було виявлено.

### 2.3.5 Випробування роботи програмного забезпечення

Методику випробувань для розробленого програмного забезпечення вебсайту наведено в Додатку В. На основі додатка можемо провести випробування роботи програмного забезпечення.

Для початку роботи, користувачу потрібно перейти до вебсайту за допомогою одного з браузерів наведених у пункті “Постанова задачі”. Перед користувачем завантажиться головна сторінка вебсайту, яка зображена на рисунку 2.31.

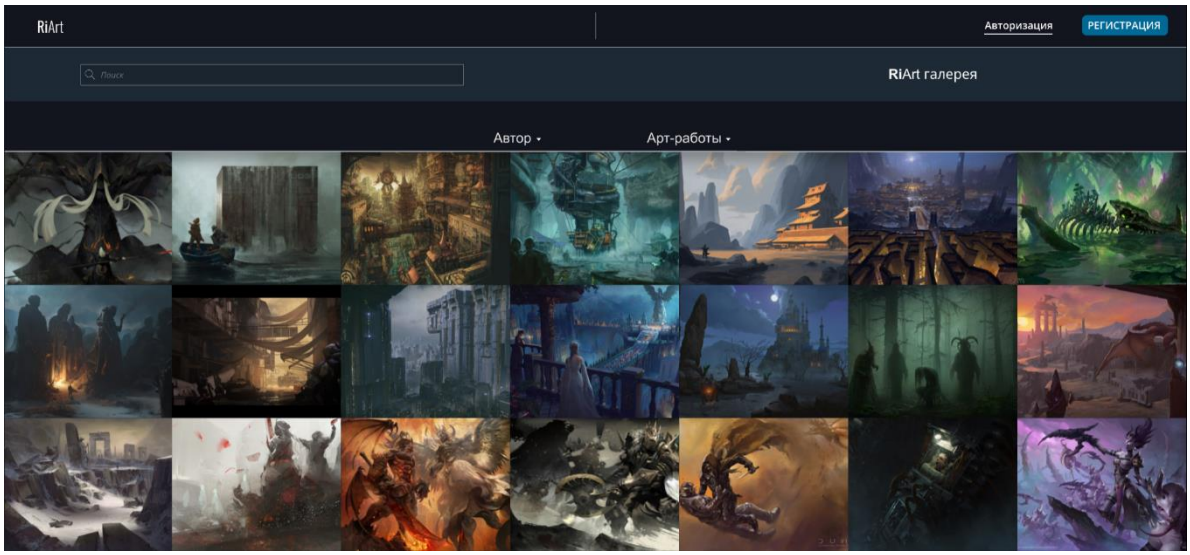


Рисунок 2.31 - Перша сторінка вебсайту яку побачить користувач при пересиланні на вебсайт

Ніяких проблем при завантаженні сторінки не відбулося.

Користувач може авторизуватися чи одразу почати пошук робіт. Пошук робіт представлено на рисунках 2.32 та 2.33. Аналогічно процес реєстрації та авторизації зображений на рисунках 2.34 та 2.35. Перенаправлення до сторінки користувача після реєстрації відбувається автоматично (рисунок 2.36).

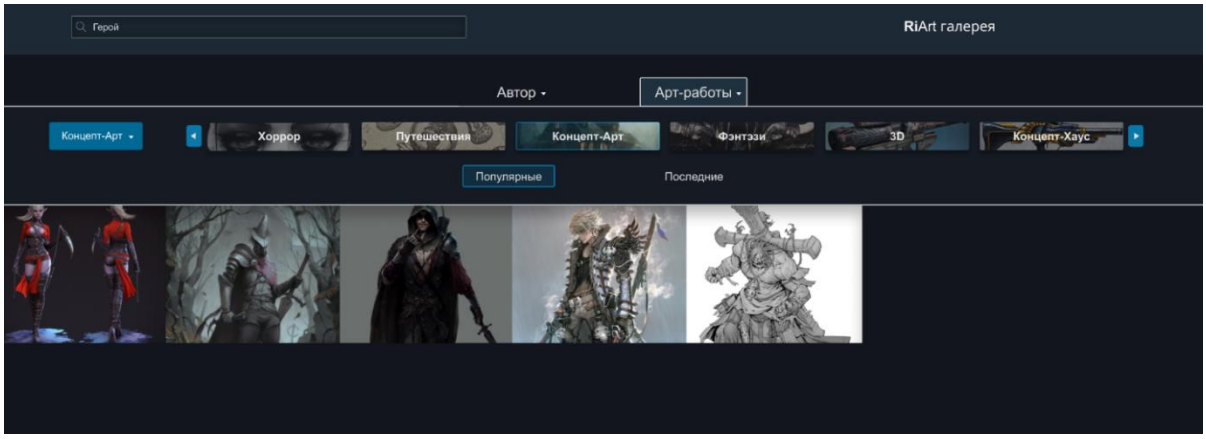


Рисунок 2.32 - Пошук робіт за фільтрами на вебсайті

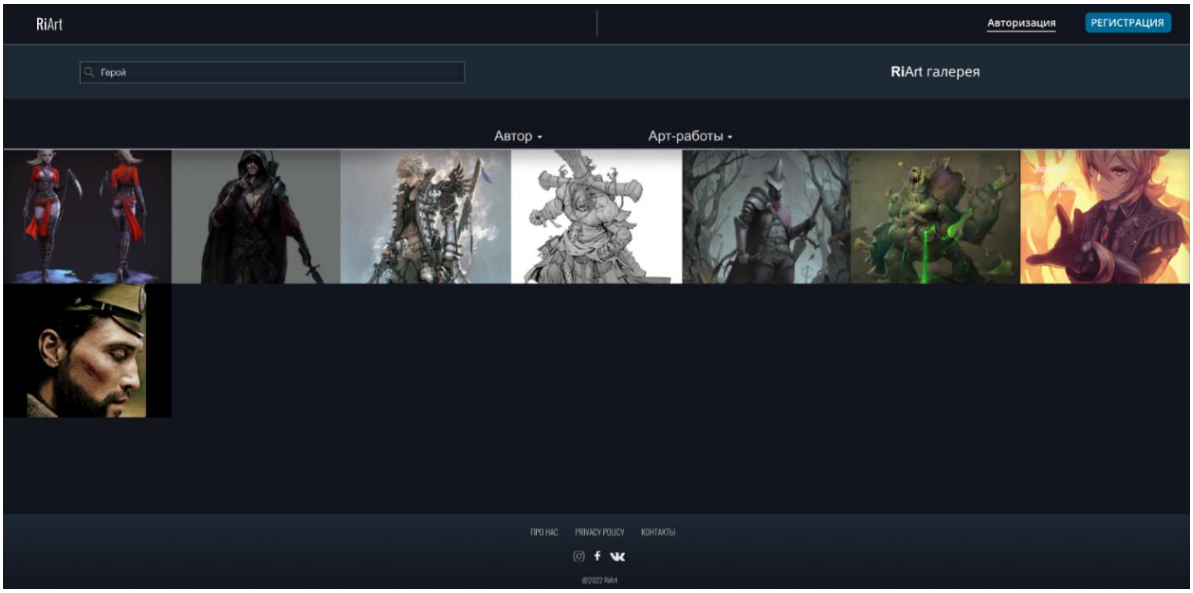


Рисунок 2.33 - Пошук робіт без використання фільтрів на вебсайті

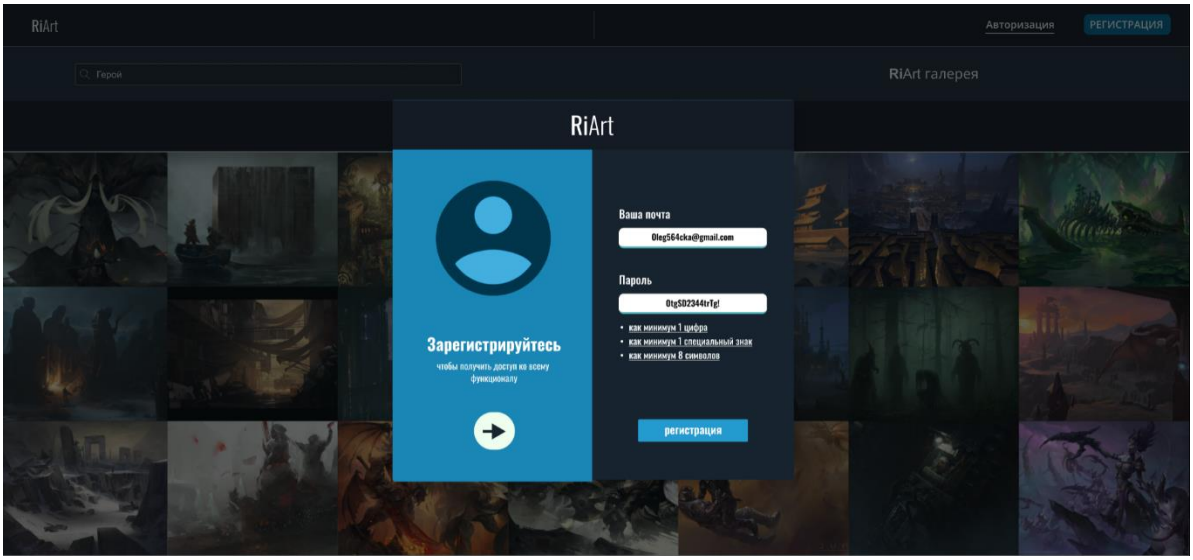


Рисунок 2.34 - Процес реєстрації на вебсайті

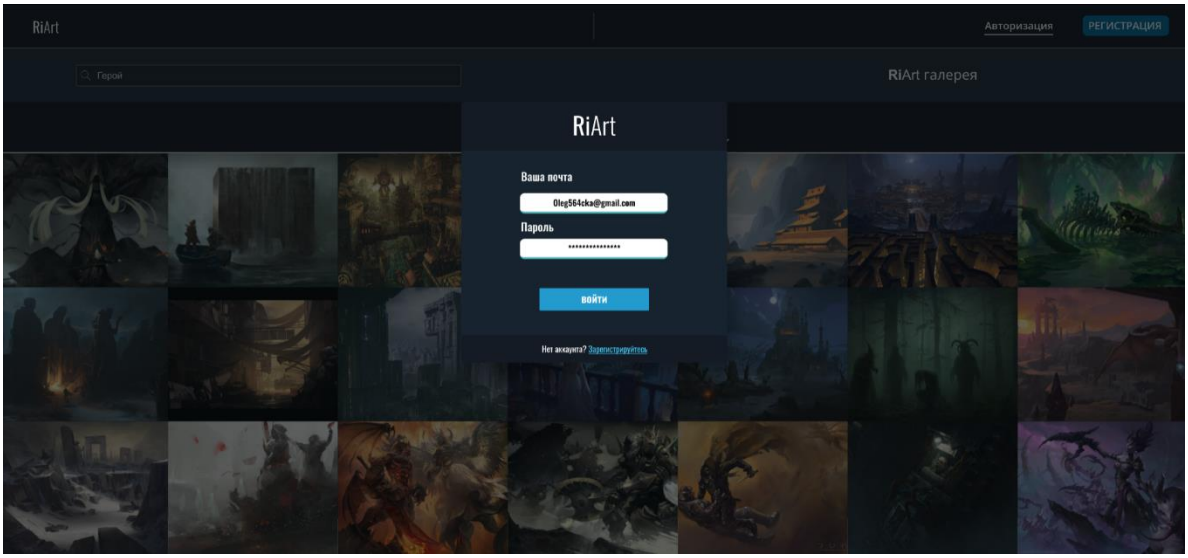


Рисунок 2.35 - Процес авторизації на вебсайті

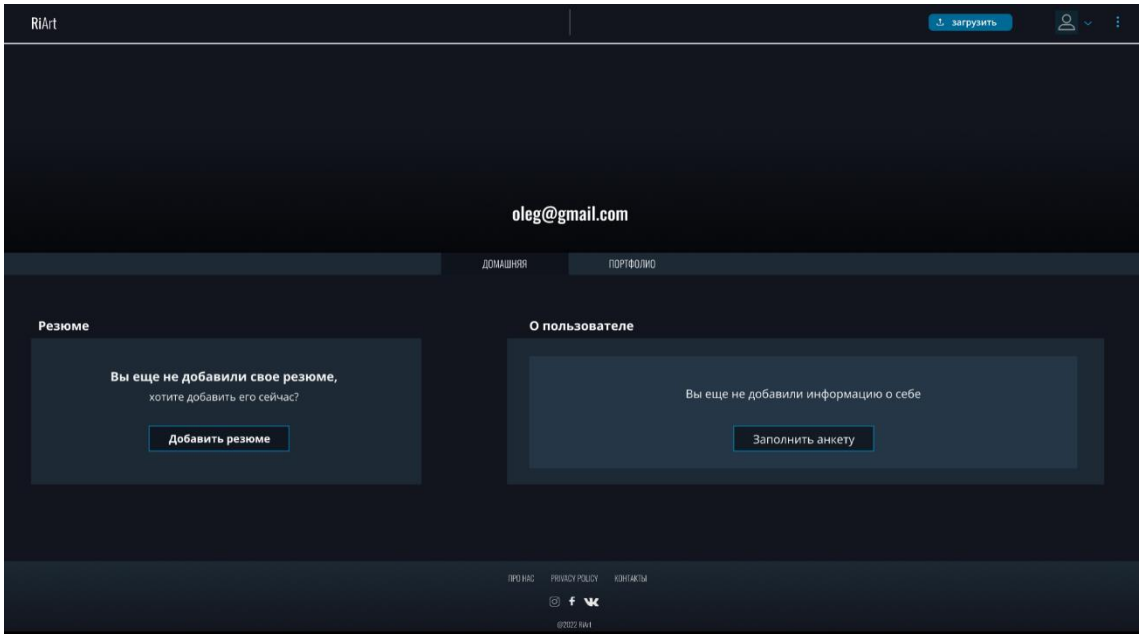


Рисунок 2.36 - Перенаправлення до сторінки профілю користувача на вебсайті

Після авторизації/реєстрації перенаправлення до сторінки з профілем користувача відбувається без помилок.

Для виходу з профілю користувача потрібно на будь-якій сторінці авторизованому користувачу натиснути на портрет користувача, та вибрати зі списку пункт вийти з облікового запису (рисунок 2.37).

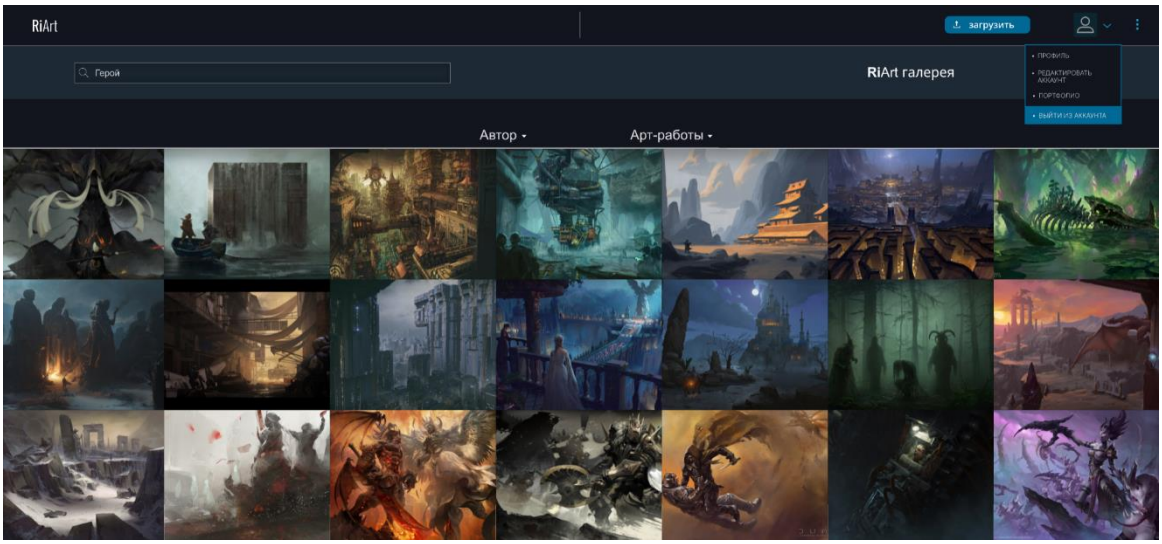


Рисунок 2.37 - Процес виходу із облікового запису на вебсайті

Відкриття списку пройшло успішно.

При натисканні кнопки вийти з облікового запису, користувача поверне до головної сторінки та користувач вийде з профілю (рисунок 2.38).

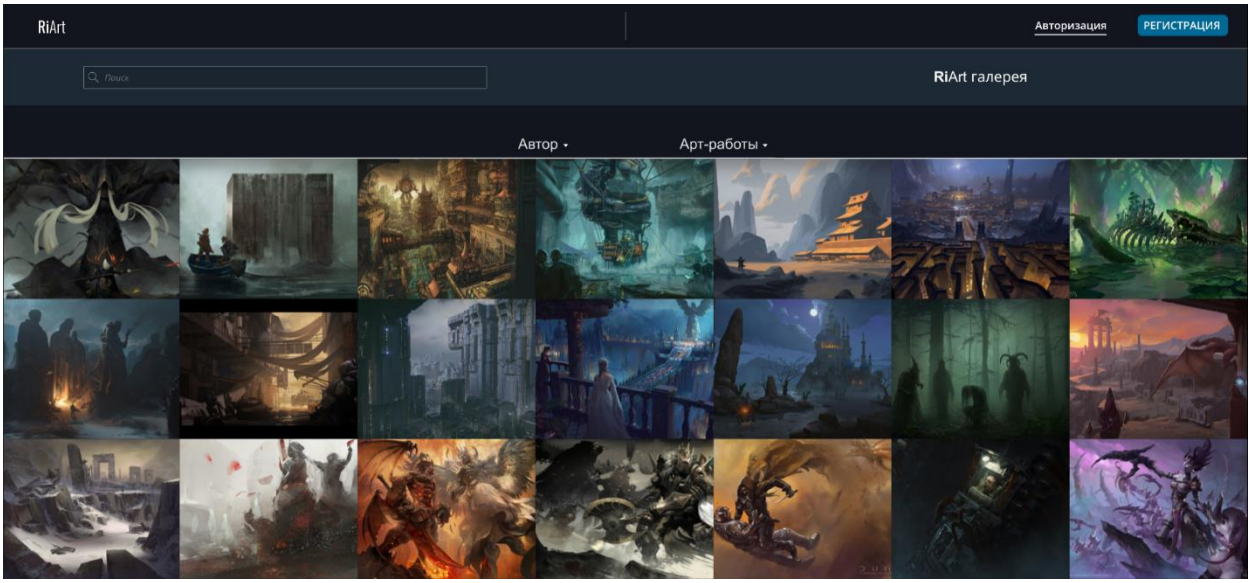


Рисунок 2.38 - переадресація до головної сторінки вебсайту

Переадресація та вихід з профілю користувача пройшло успішно. Ніяких помилок у процесі не було виявлено.

При проведенні усіх тестів помилок виявлено не було, тому можна зробити висновок що програмне забезпечення вебсайту онлайн-галереї “RiArt” працює без помилок, а тому випробування було пройдено успішно.

# 3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБСАЙТУ ДЛЯ ОНЛАЙН-ГАЛЕРЕЇ “RIART”

Було розроблено програмне забезпечення вебсайту для онлайн-галереї “RiArt”.

Розроблене програмне забезпечення було реалізовано на об’єктноорієнтованій мові програмування C# з використанням фреймворку ASP.Net core. Частина тексту реалізованої програми представлено у Додатку А. Програмне забезпечення складається із 1359 рядків.

Якість розробки була протестована за допомогою методики тестування вказаної у Додатку В. Був зроблений висновок що програмний продукт відповідає усім поставленим вимогам, та є повністю працездатним.

У Додатку Б представлена інструкція для користувача, для роботи з програмним забезпеченням вебсайту.

# 4 ОХОРОНА ПРАЦІ

## 4.1 Основні положення

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності [6].

За допомогою охорони праці вирішується наступні завдання:

- інженерно-технічне;
- соціальне.

Інженерно-технічне - направлено на зменшення ризиків при роботі, за допомогою впровадження нових технологій та заміни матеріалів на менш небезпечні.

Соціальне - направлено на відшкодування шкоди, яка була завдана під час трудового процесу, внаслідок нещасних випадків, та захисту прав працівника.

Усі права працівника на безпечні умови праці захищаються законом України “Про охорону праці”.

## 4.2 Аналіз шкідливих та небезпечних факторів під час роботи з екранними пристроями

Існують небезпечні та шкідливі фактори роботи за комп’ютером які потрібно знати. До головних можна віднести роботу з екранними пристроями та робоче місце працівника.

1. Вимоги безпеки до робочих місць працівників з екранними пристроями [7]:

1.1 Під час облаштування робочого місця працівника з екранними пристроями необхідно обирати таке устаткування, яке не створює зайвого шуму та не виділяє надлишкового тепла;

1.2 Робочі місця працівників з екранними пристроями мають бути спроєктовані так і мати такі розміри, щоб працівники мали простір для зміни робочого положення та рухів. (Через тривале перебування у сидячому положенні без рухів виникає напруга, яка може викликати проблеми зі здоров'ям та працездатністю працівника);

1.3 Для забезпечення безпеки та захисту здоров'я працівників усе випромінювання від екранних пристроїв має бути зведене до гранично допустимого рівня з погляду безпеки та охорони здоров'я працівників. (Часте перебування перед екраном комп'ютера може викликати значне навантаження на органи зору. Результатом є погіршення зору, втома й погіршення працездатності працівника);

1.4 Організація робочого місця працівника з екранними пристроями має забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним, антропологічним, психофізіологічним вимогам, а також характеру виконуваних робіт;

1.5 Робочий стіл або робоча поверхня повинні бути достатнього розміру та мати поверхню з низькою відбивною здатністю, допускати гнучкість під час розміщення екрана, клавіатури, документів і відповідного устаткування;

1.6 Робоче крісло має бути стійким і дозволяти працівнику з екранними пристроями легко рухатися та займати зручне положення;

Норми мікроклімату зазначені у санітарних нормах мікроклімату виробничих приміщень ДСН 3.3.6.042-99.

Температура повітря може відрізнятися від норм залежно від пори року та інших умов. Щоб уникнути небажаних відхилень потрібно встановлювати допоміжні засоби регулювання мікроклімату: кондиціонери, обігрівачі тощо.

Найбільш прийнятними методами захисту від шуму є використання акустичних екранів і звукопоглинальних облицювань.

Акустичний екран є перешкодою для звукових хвиль, що знижує рівень звуку шляхом утворення акустичної тіні за екраном в зоні розташування робочого місця.

Ергономічна безпека праці досягається за допомогою нескладних правил. Спина людини повинна бути нахилена назад під кутом в декілька градусів для збільшення кута між тулубом і стегнами, посилення кровообігу і зменшення тиску на хребет. Руки розслаблені й вільно опущені уздовж боків, передпліччя і кисті розташовані паралельно підлозі. Стегна знаходяться під прямим кутом до тулуба. Коліна - під прямим кутом до стегон.

Спинка стільця повторює лінію вигину нижньої частини спини. Сидіння злегка нахилене вперед для перенесення тиску з хребта на стегна і ноги. Край подушки сидіння заломлений вниз для ослаблення тиску на стегна.

Максимальний час безперервної роботи за комп'ютером не повинен перевищувати 4-х годин в день. Через кожні 7 хвилин потрібно робити коротку перерву на 10-15 секунд. Можна подивитися у вікно (у далечінь) або на картину з приємним для очей пейзажем, яку необхідно повісити за комп'ютером. У картині повинні переважати неяскраві, заспокійливі зір кольори, такі, наприклад, як зелені, блакитні. Також потрібно робити декілька простих вправ для очей.

2. Вимоги безпеки після закінчення роботи:

2.1 Після закінчення роботи фахівець зобов’язаний: відключити від електромережі екранний пристрій, засоби оргтехніки та інше устаткування, крім обладнання, яке використовується цілодобово (апарати факсимільного зв’язку, мережеві сервери тощо); упорядкувати робоче місце, провітрити

приміщення; привести себе у порядок, вимити руки й обличчя та переодягнутися;

2.2 Зберегти інформацію на комп’ютері;

2.3 Прибрати робоче місце.

# ВИСНОВКИ

В кваліфікаційній роботі вирішено практичну задачу інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розроблено програмне забезпечення вебсайту для онлайн-галереї "RiArt".

У процесі виконання кваліфікаційної роботи було проведено аналіз онлайн-галерей та існуючих вебсайтів онлайн-галерей.

Після аналізу існуючих аналогів вебсайтів онлайн-галерей таких як “ArtStation” та “Pixiv”, виявленню їх основного функціоналу, переваг та недоліків, було зроблено висновок про те що вони не надають можливості безоплатного створення та розміщення резюме та портфоліо.

Завдяки проведеному аналізу було зроблено висновок про те, що власна розробка є доцільною. Було розроблено ескізний, технічний та робочий проєкти програмного забезпечення вебсайту для онлайн-галереї “RiArt”.

Після виконання випробувань за Додатком В, було визначено, що ніяких помилок, які свідчать про неправильну роботу програмного продукту, виявлено не було.

У розділі з охорони праці було розглянуто завдання які вирішує охорона праці та проаналізовані небезпечні фактори роботи працівників з комп’ютером.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Місце ArtStation у світі за популярністю та дата графіками. URL: https://www.similarweb.com/website/artstation.com (дата звернення: 28.04.2022);

2. Інформація стосовно функціоналу платної підписки вебсайту Pixiv. URL: https://www.pixiv.net/premium/ (дата звернення: 29.04.2022);

3. Pixiv - особливості та аналоги. URL: https://ruprogi.ru/software/pixiv (дата звернення: 29.04.2022);

4. SaaS платформи - URL: https://hostiq.ua/wiki/about-sitebuilder/ (дата звернення: 27.05.2022);

5. Microsoft документація - компіляція C# та .NET коду. URL: https://docs.microsoft.com/ru-ru/dotnet/standard/managed-execution-process (дата звернення: 27.05.2022);

6. Про охорону праці. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 30.04.2022);

7. Про затвердження Вимог щодо безпеки та захисту здоров’я працівників. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#n14 (дата звернення: 30.04.2022).

# ДОДАТОК А КОД ПРОГРАМИ

Startup.cs

```
public class Startup
    {
        public Startup(IConfiguration configuration)
        {
            Configuration = configuration;
        }

        public IConfiguration Configuration { get; }

        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllersWithViews(mvcOtions=>
            {
                mvcOtions.EnableEndpointRouting = false;
            });
     services.AddAuthentication(CookieAuthenticationDefaults.Authentic
ationScheme)
                .AddCookie(options =>
                {
                    options.LoginPath = new
Microsoft.AspNetCore.Http.PathString("/Account/Login");
                    options.AccessDeniedPath = new
Microsoft.AspNetCore.Http.PathString("/Account/Login");
                });

        }

        public void Configure(IApplicationBuilder app,
IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }

app.UseHttpsRedirection();
          app.UseStaticFiles();
```

```
            app.UseStaticFiles();

            app.UseAuthentication();
             app.UseAuthorization();


             // добавление компонентов mvc и определение маршрута
             app.UseMvc(routes =>
             {

           routes.MapRoute(
                     name: "default",
                     template:
"{controller=Home}/{action=Index}/{id?}");

           routes.MapRoute(
                     name: "admin",
                     template:
"{controller=Home}/{action=Index}/{id?}");

                   routes.MapRoute("profile", "profile/user{id?}", new
{ controller = "User", action = "ShowUserAccountInfo" });

                   routes.MapRoute("art-post", "art-post/user{id?}",
new { controller = "ArtWorkPost", action = "ShowUserAccountInfo" });

             });
         }
    }
```

Надалі наведенні класи моделей:

```
public class User
{

    public int Id { get; set; }

    public string Email { get; set; }

    public string Password { get; set; }

    public string RoleId{ get; set; }

    public UserProfile userProfile {get; set; }
```

```
    public int? RoleId { get; set; }

    public DateCreated DateTime { get; set; }

    public Role Role { get; set; }

    pubic User(){
     Role  = getRole(RoleId);
}
}

public class Role
{
    public int Id { get; set; }
    public string Name { get; set; }
    public List<User> Users { get; set; }
    public Role()
    {
        Users = new List<User>();
    }
}

public class UserProfile
    {
        [Key]
        public int ID { get; set; }
        [MaxLength(50)]
        public string FullName { get; set; }
        [MaxLength(20)]
        public string City { get; set; }
        public string Country { get; set; }
        [Range(10,15)]
        public string PhoneNumber { get; set; }
        public string Background { get; set; }
        public string Resume { get; set; }
        public string ContactInformation { get; set; }
        public string Skills { get; set; }
        public string[] Languages { get; set; }
        public string WorkExperience { get; set; }
        public string EducationalBacground { get; set; }
        public string AboutImg { get; set; }
        public List<ArtWorkPost> ArtWorkPosts { get; set; }

        public UserProfile()
        {

        }
```

```
    }
public class Image
    {
        [Key]
        public int Id { get; set; }
        [Required]
        public string Name { get; set; }
        [Required]
        public string Location { get; set; }
        public ArtWorkPost Post { get; set; }

        public Image()
        {

        }
    }
public class ArtWorkPost
    {
        [Key]
        public int Id { get; set; }
        [MaxLength(20), Required]
        public string Name { get; set; }
        [MaxLength(250)]
        public string Description { get; set; }
        [Required]
        public List<Image> Images { get; set; }
        [Required]
        public List<Filter> Filters { get; set; }
        [Required]
        public User User { get; set; }
         public DateCreated DateTime { get; set; }
        public bool State { get; set; } = true;
        public int ComplaintsCounter { get; set; } = default

        public ArtWorkPost(List<Image> images, List<Filter> filters)
        {
            Images = images;
            Filters = filters;
        }
    }
public class Filter
    {
        [Key]
        public int Id { get; set; }
       [Required, MaxLength(15)]
        public string Name { get; set; }

        public Filter()
        {
```

```
        }
    }
```

Надалі наведено клас для роботи з БД та міграцією:

```
  public class ApplicationContext : DbContext
    {
        public DbSet<User> Users { get; set; }
        public DbSet<UserRole> UserRoles { get; set; }
        public DbSet<Image> Images { get; set; }
        public DbSet<ArtWorkPost> ArtWorkPost  { get; set; }
        public DbSet<Filter> Filter { get; set; }
        public DbSet<Role> Roles { get; set; }
        public DbSet<UserProfile> UserProfile { get; set; }

        public ApplicationContext(DbContextOptions<ApplicationContext>
options)
            : base(options)
        {

        }
        protected override void OnModelCreating(ModelBuilder
modelBuilder)
        {
            string adminRoleName = "admin";
            string userRoleName = "user";

            string adminEmail = "admin@mail.ru";
            string adminPassword = "admin";

            Role adminRole = new Role { Id = 1, Name = adminRoleName
};
            Role userRole = new Role { Id = 2, Name = userRoleName };
            User adminUser = new User { Id = 1, Email = adminEmail,
Password = adminPassword, RoleId = adminRole.Id };

            modelBuilder.Entity<Role>().HasData(new Role[] {
adminRole, userRole });
            modelBuilder.Entity<User>().HasData( new User[] {
adminUser });
            base.OnModelCreating(modelBuilder);
        }
    }
}
```

Надалі будуть представленні контролери:

```
public class AccountController : Controller
    {
        private ApplicationContext _context;
        public AccountController(ApplicationContext context)
        {
            _context = context;
        }
        [HttpGet]
        [Route("home/Register")]
        public IActionResult Register()
        {
            return ViewComponent();
        }
        [HttpGet]
        [Route("home/Login")]
        public IActionResult Login()
        {
            return ViewComponent();
        }
         [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Register(User model)
        {
            if (ModelState.IsValid)
            {
                User user = await _context.Users.FirstOrDefaultAsync(u
=> u.Email == model.Email);
                if (user == null)
                {
                    user = new User { Email = model.Email, Password =
model.Password };
                    Role userRole = await
_context.Roles.FirstOrDefaultAsync(r => r.Name == "user");
                    if (userRole != null)
                        user.Role = userRole;

                    _context.Users.Add(user);
                    await _context.SaveChangesAsync();

                    await Authenticate(user);

                    return RedirectToAction("Index", "Home");
                }
                else
                    ModelState.AddModelError("", "Некорректные логин
и(или) пароль");
            }
```

```
            return View(model);
        }
        [HttpGet]
        public IActionResult Login()
        {
            return View();
        }
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Login(User model)
        {
            if (ModelState.IsValid)
            {
                User user = await _context.Users
                    .Include(u => u.Role)
                    .FirstOrDefaultAsync(u => u.Email == model.Email
&& u.Password == model.Password);
                if (user != null)
                {
                    await Authenticate(user);

                    return RedirectToAction("Index", "Home");
                }
                ModelState.AddModelError("", "Некорректные логин
и(или) пароль");
            }
            return View(model);
        }
        private async Task Authenticate(User user)
        {

            var claims = new List<Claim>
            {
                new Claim(ClaimsIdentity.DefaultNameClaimType,
user.Email),
                new Claim(ClaimsIdentity.DefaultRoleClaimType,
user.Role?.Name)
            };

            ClaimsIdentity id = new ClaimsIdentity(claims,
"ApplicationCookie", ClaimsIdentity.DefaultNameClaimType,
                ClaimsIdentity.DefaultRoleClaimType);

            await
HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationSch
eme, new ClaimsPrincipal(id));
        }
    }
```

```
    public class ArtWorkPostController : Controller
    {
        private ApplicationContext _context;
        public ArtWorkPostController(ApplicationContext context)
        {
            _context = context;
        }
        [Route("home/post{?id}")]
        public IActionResult Post()
        {
            return View();
        }

        [HttpGet]
        public async Task<IActionResult> GetListOfWorks(ArtWorkPost
model)
        {
            var stroke = from m in _context.ArtWorkPost
                         select m;

            return View(await stroke.ToListAsync());
        }

        [HttpGet]
        public async Task<IActionResult> GetListOfFilters(Filter
model)
        {
            var stroke = from m in _context.Filter
                         select m;

            return View(await stroke.ToListAsync());
        }


        [HttpGet]
        public async Task<IActionResult> ShowPost(ArtWorkPost model)
        {
            var artWorkPost = from m in _context.ArtWorkPost
                         select m;

            if (model.State)
            {
                artWorkPost = artWorkPost.Where(s =>
s.Id!.Contains(model.Id));
                return View(await artWorkPost.ToListAsync());
            }
            else
            {
                return View(await Exception(StringBuilderContext));
            }
        }
```

```
        [HttpGet]
        public async Task<IActionResult> Search(string searchString)
        {
            var artWorksPosts = from m in _context.Filter
                        select m;

            if (!String.IsNullOrEmpty(searchString))
            {
                artWorksPosts = artWorksPosts.Where(s =>
s.Title!.Contains(searchString));
            }

            return View(await artWorksPosts.ToListAsync());
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> BlockPost(ArtWorkPost model)
        {
            if (model.Id == null)
            {
                return NotFound();
            }

            var artWorkPostToUpdate = await _context.Courses
                .FirstOrDefaultAsync(c => c.CourseID == id);

            if (await
TryUpdateModelAsync<ArtWorkPost>(artWorkPostToUpdate,
                "",
                c => c.State))
            {
                try
                {
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateException)
                {

                    ModelState.AddModelError("", "Unable to save
changes. " +
                        "Try again, and if the problem persists, " +
                        "see your system administrator.");
                }
                return RedirectToAction(nameof(home));
            }

            return View(artWorkPostToUpdate);
```

```
        }


          [HttpPost]
           [ValidateAntiForgeryToken]
           public async Task<IActionResult> Ulpoad(FileContentResult
file)
           {
               long size = file.FileDownloadName(f => f.Length);

               private string[] permittedExtensions = {
Formats.Aviliable.GetList() };

               var ext =
Path.GetExtension(uploadedFileName).ToLowerInvariant();

               if (string.IsNullOrEmpty(ext) ||
!permittedExtensions.Contains(ext))
               {
                   await formFile.CopyToAsync(stream);
               }else{
                        foreach (var formFile in file)
                        {
                            if (formFile.Length > 0)
                            {
                                var filePath = Path.GetTempFileName();

                                using (var stream =
System.IO.File.Create(filePath))
                                {
                                    await formFile.CopyToAsync(stream);
                                }
                            }
                        }
                       await return Ok(new { count = file.Count, size
});
                   }
             }


        [HttpGet]
        public async Task<IActionResult> DownloadImage(FileContent
file)
        {
            long size = file.FileDownloadName(f => f.Length);

            private string[] permittedExtensionsStroke = {
Formats.Aviliable.GetList() };
```

```
            var ext =
Path.GetExtension(uploadedFileName).ToLowerInvariant();
            File img = file.get(artWorkPost.Where(s =>
s.Image!.Contains(ext)));
            return View(await img);
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> CreatePost(ArtWorkPost model)
        {
            long size = file.FileDownloadName(f => f.Length);

                        private string[] permittedExtensions = {
Formats.Aviliable.GetList() };

        var ext =
Path.GetExtension(uploadedFileName).ToLowerInvariant();

                if (string.IsNullOrEmpty(ext) ||
!permittedExtensions.Contains(ext))
                {
                    await formFile.CopyToAsync(stream);
                }
                else
                {
                    foreach (var formFile in file)
                    {
                        if (formFile.Length > 0)
                        {
                            var filePath = Path.GetTempFileName();

                            using (var stream =
System.IO.File.Create(filePath))
                            {
                                await formFile.CopyToAsync(stream);
                            }
                        }
    } catch (DbUpdateException)
    {

        ModelState.AddModelError("", "Unable to create. " +
            "Try again, and if the problem persists, " +
            "see your system administrator.");
    }
    var ext = Path.GetExtension(uploadedFileName).ToLowerInvariant();
    File img = file.get(artWorkPost.Where(s =>
s.Image!.Contains(ext)));
    await return Ok(new ArtWorkPost);
                }
```

```
            }

            }
}
  public class UserController : Controller
    {
        private ApplicationContext _context;
        public UserController(ApplicationContext context)
        {
            _context = context;
        }

        [Route("home/profile/edit")]
        public IActionResult Edit()
        {
            return View();
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> ChangeAccountInfo(User model)
        {

            var userToUpdate = await _context.User
                .FirstOrDefaultAsync(c => c.model.Id == id);

            if (await TryUpdateModelAsync<User>(userToUpdate,
                "",
                c => c.Login, c.Email, c.Password))
            {
                if (Created() == CValidtorSystem{
                    try
                    {
                        await _context.SaveChangesAsync();
                    }
                    catch (DbUpdateException)
                    {


ModelState.AddModelError(DbUpdateException.Message);
                    }
                    return RedirectToAction(nameof(edit));
                }
                else { return View(User); }


            }

            [HttpPost]
```

```
            [ValidateAntiForgeryToken]
            public async Task<IActionResult>
ChangeProfileInfo(UserProfile model)
            {

                var userProfileToUpdate = await _context.User
                    .FirstOrDefaultAsync(c => c.model.Id == id);

                if (await
TryUpdateModelAsync<UserProfile>(userProfileToUpdate,
                    "",
                    c => c.AboutImg, c => c.Background, c => c.City, c
=> c.Country))
                {
                    try
                    {
                        if (ModelState.IsValid)
                        {
                            _context.Add(userProfileToUpdate);
                            await _context.SaveChangesAsync();
                            return RedirectToAction(nameof(profile));
                        }
                    }
                        try
                    {
                        await _context.SaveChangesAsync();
                    }
                    catch (DbUpdateException)
                    {


ModelState.AddModelError(DbUpdateException.Message);
                    }
                    return RedirectToAction(nameof(edit));
                }
                await
TryUpdateModelAsync<UserProfile>(userProfileToUpdate,
                   "",
                   c => c.AboutImg, c => c., c => c.City, c =>
c.Country)
                else { return View(User); }
            }
        }


        [HttpGet]
        public async Task<IActionResult>
ShowUserAccountInfo(UserProfile model)
        {
            var User = from m in _context.User.Where(s =>
s.Id!.Contains(id))
```

```
                                select m;


            var info = from User in _context.UserProfile select m;

            Constraint.All(User, info).ToList;

            return await View(info);
        }


        [HttpGet]
        public async Task<IActionResult> ShowUserPortfolio(int? id)
        {
            var portfolio = from m in _context.ArtWorkPost.Where(s =>
s.Id!.Contains(id))
                            select m;

            return View(await portfolio.ToListAsync());
        }


        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult>
UlpoadResume(FileContentResult file)
        {
            long size = file.FileDownloadName(f => f.Length);

               private string[] permittedExtensions = {
Formats.Aviliable.GetList() };

                var ext = Path.GetExtension(file.Name);

               if (string.IsNullOrEmpty(ext))
               {
                   await formFile.CopyToAsync(stream);
                    }else{
                       foreach (var formFile in file)
                       {
                           if (formFile.Length > 0)
                           {
                               var filePath = Path.GetTempFileName();

                               using (var stream =
System.IO.File.Create    (filePath))
                               {
                                   await formFile.CopyToAsync(stream);
                                }
                           }
```

```
                    await return Ok(new { FileResult });

                }
            }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult>
UlpoadImg(FileContentResult file)
        {
            string uniqueFileName = null;

            if (model != null)
            {
                string uploadsFolder =
Path.Combine(webHostEnvironment.WebRootPath, "images");
                uniqueFileName = Guid.NewGuid().ToString() + "_" +
model.ProfileImage.FileName;
                string filePath = Path.Combine(uploadsFolder,
uniqueFileName);
                using (var fileStream = new FileStream(filePath,
FileMode.Create))
                {
                    model.ProfileImage.CopyTo(fileStream);
                }
            }
            return uniqueFileName;
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> CreateResume(UserProfile
model)
        {

            _context.Entry(model).State = EntityState.Modified;
            _context.SaveChanges();
            return RedirectToAction("Edit");
        }

    }
}
```

# ДОДАТОК Б ІНСТРУКЦІЯ КОРИСТУВАЧА

Для роботи з програмним забезпеченням потрібно перейти до вебсайту за допомогою одного із визначеного у програмних вимогах браузера.

При переході на вебсайт першим що побачить користувач буде головною сторінкою незареєстрованого користувача (рисунок Б.1).

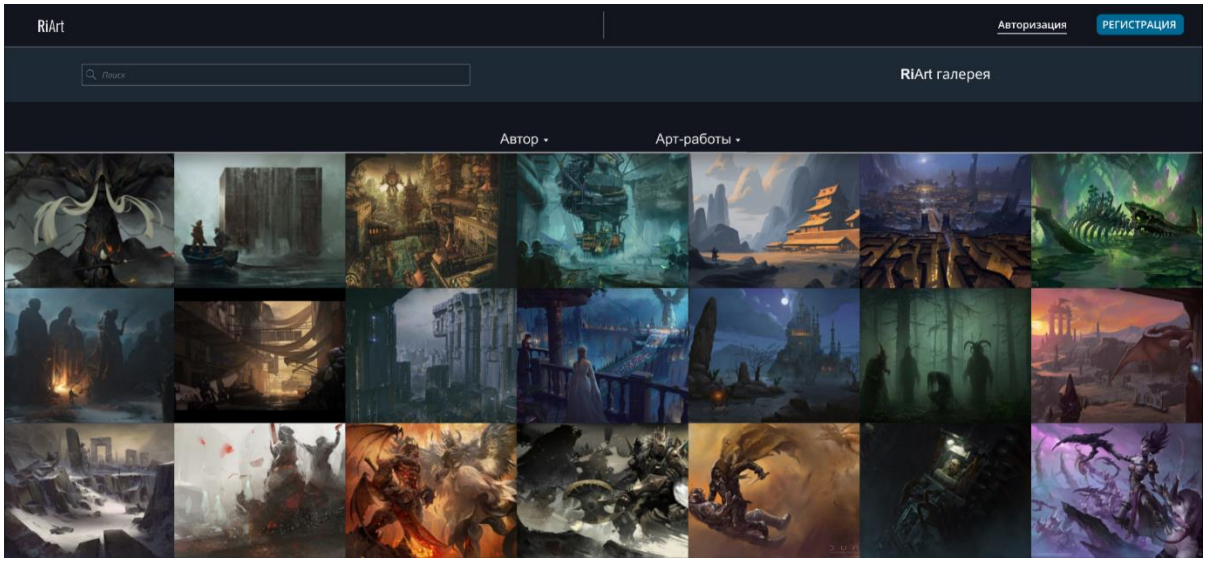


Рисунок Б.1 - Головна сторінка вебсайту

На цій сторінці розміщена коротка інформація про сайт, яка потрібна для розуміння куди потрапив користувач, а також розташовані одні з основних функціональних можливостей, а саме Авторизація/Реєстрація користувача, пошук робіт за фільтром та без, та перегляд робіт.

Для авторизації/реєстрації потрібно натиснути на відповідні кнопки Авторизація та Реєстрація.

Для ініціювання пошуку ввести запит до строки пошуку, та натиснути Enter на клавіатурі.

Для вибору фільтрів потрібно відкрити список допоміжних фільтрів натиснувши на список кнопки Автор чи Арт Роботи.

Для відкриття роботи із представлених в галереї, потрібно натиснути на роботу яка зацікавила користувача.

Уся інтерактивна частина сайту тут і надалі буде вказана та виділена червоним кольором на рисунках.

На рисунку Б.2. представлена інтерактивна частина головної сторінки.

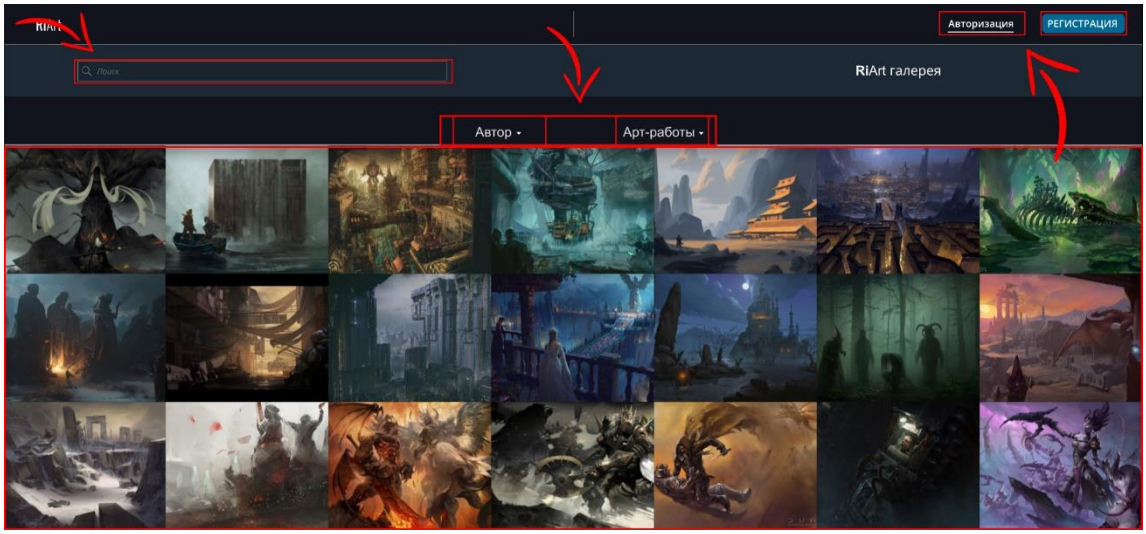


Рисунок Б.2 – Інтерактивна частина головної сторінки вебсайту

Для реєстрації користувача на сайті та створенню профілю користувача як клієнта, потрібно заповнити форму, яка з’явиться після натискання вище вказаних кнопок. Після заповнення форми натиснути кнопку Реєстрація.

Інтерактивна частина форми реєстрації представлена на рисунку Б.3.

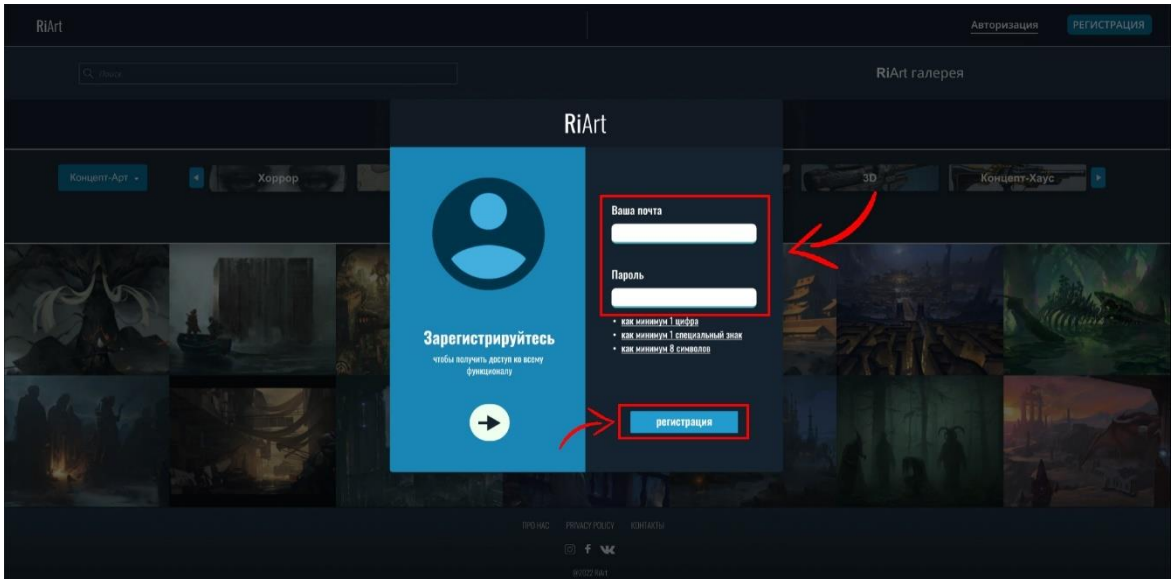


Рисунок Б.3 – Інтерактивна частина форми реєстрації

Зауваження щодо використання форми реєстрації:

- на одну електронну пошту можливо зареєструвати лише одного користувача;
- пароль повинен відповідати вказаним у формі правилам заповнення.

Для авторизації зареєстрованого користувача потрібно заповнити форму, даними вказаними при реєстрації, яка з’явиться після натискання кнопки авторизація, та натиснути кнопку Увійти. Якщо користувач ще не має зареєстрованого облікового запису, форма представляє можливість натиснувши на посилання “зареєструватись”, відкрити форму для реєстрації.

Інтерактивна частина форми авторизації представлена на рисунку Б.4.

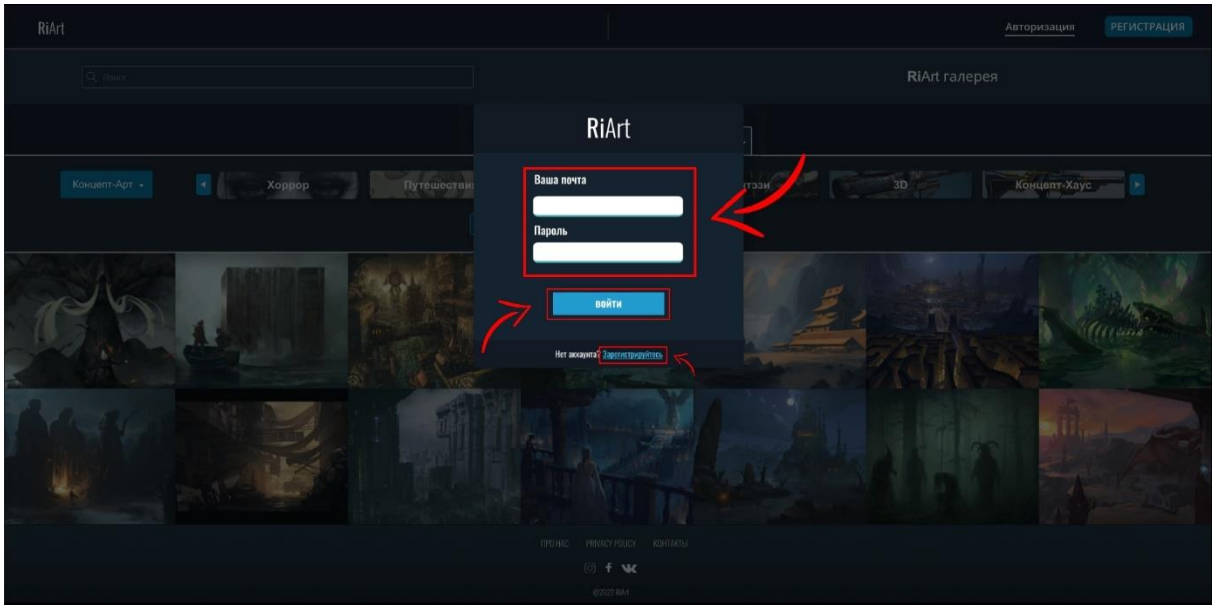


Рисунок Б.3 – Інтерактивна частина форми авторизації

Якщо користувач захоче закрити форму авторизації/реєстрації, потрібно натиснути у будь-якому місці окрім форми та вона закриється.

Інтерактивна частина яку потрібно натиснути, щоб закрити форму зображена на рисунках Б.4 та Б.5.

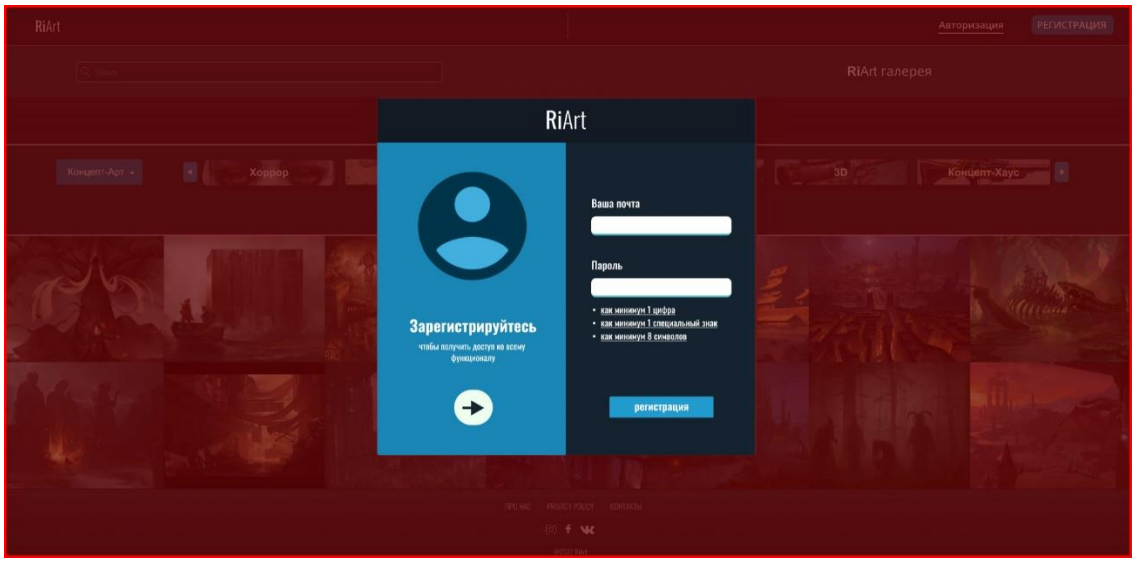


Рисунок Б.4 – Інтерактивна частина для закриття форми реєстрації

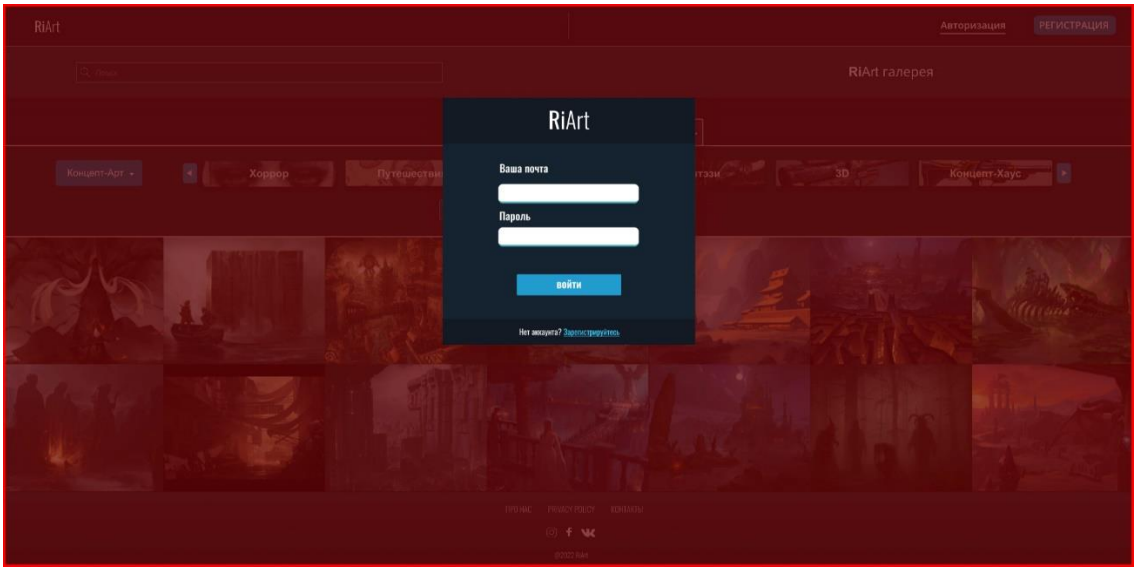


Рисунок Б.5 – Інтерактивна частина для закриття форми авторизації

Після авторизації користувачу надається додатковий функціонал за ролями. Через навігаційне меню користувача можна переходити до потрібних сторінок на вебсайті (рисунок Б.6).

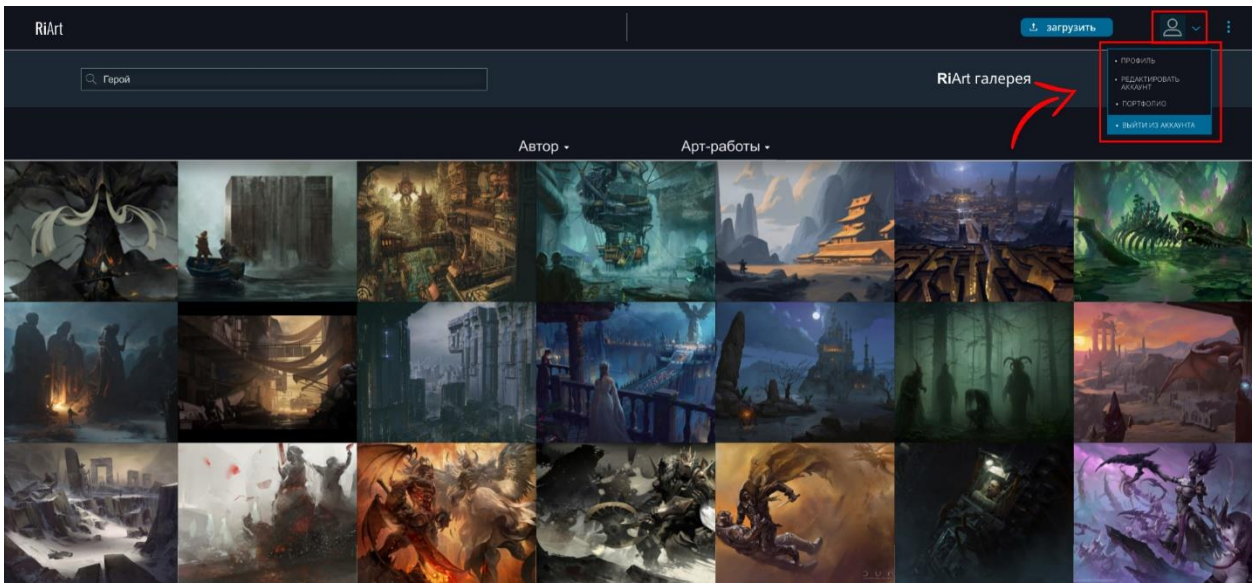


Рисунок Б.6 – Навігаційне меню користувача

Наступні операції виконуються за допомогою навігаційного меню користувача:

- Для редагування облікового запису та заповнення профілю інформацією потрібно обрати пункт редагувати обліковий запис;

- Для заповнення та редагування портфоліо, потрібно опубліковувати роботи на сайті. Для цього потрібно обрати пункт портфоліо або натиснути кнопку загрузити;
- Для переходу до перегляду профілю користувача, потрібно обрати пункт профіль;
- Для виходу з облікового запису користувача потрібно обрати пункт вийти з облікового запису.

До сторінки персонального профілю користувача також автоматично перенаправить після реєстрації. Ця вебсторінка потрібна для відображення інформації про користувача, тобто його резюме, короткої біографії, та портфоліо.

Профіль користувача зображено на рисунку Б.7.

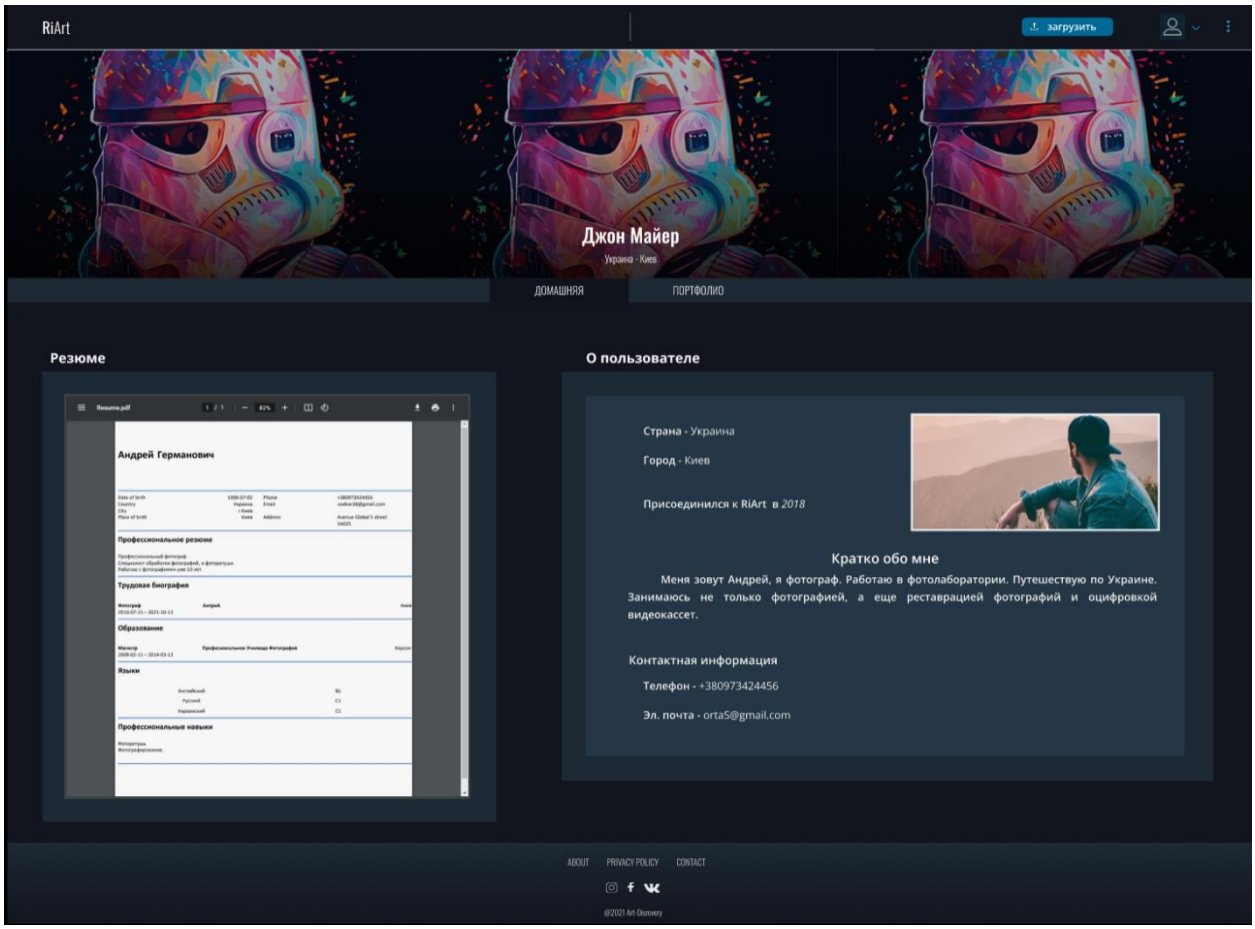


Рисунок Б.7 – Профіль користувача

Вебсторінка редагування облікового запису користувача містить різноманітні форми для заповнення профілю даними. Заповнивши, чи змінивши данні в елементах форми потрібно натиснути кнопку Зберегти для того, щоб зміни набули чинності.

На рисунку Б.8 та Б.9 зображена деяка частина доступних користувачу форм.

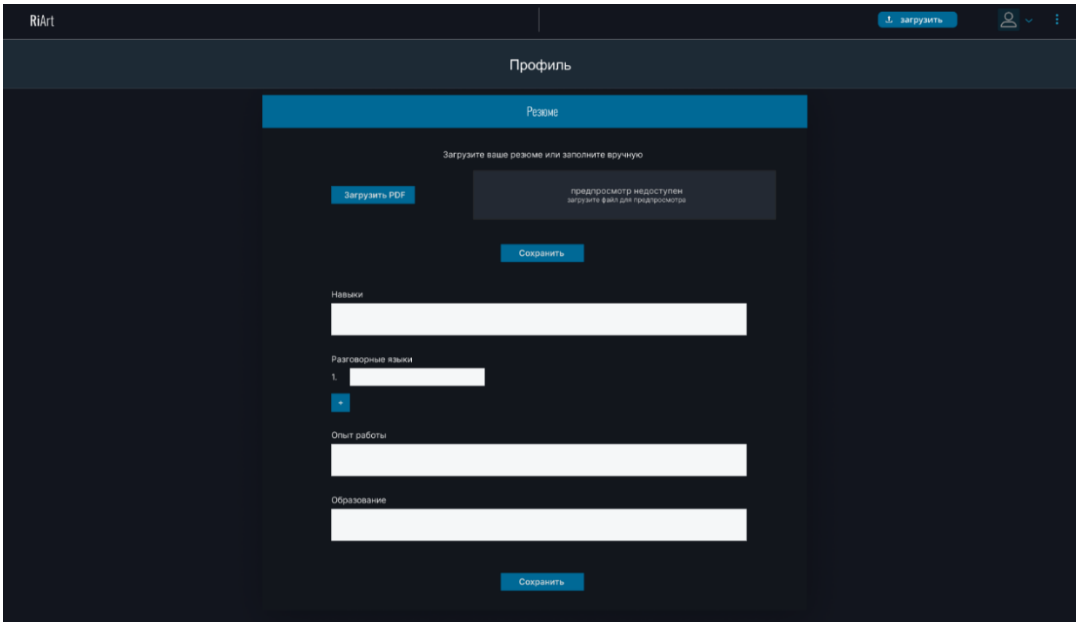


Рисунок Б.8 – Форма для заповнення/загрузки резюме

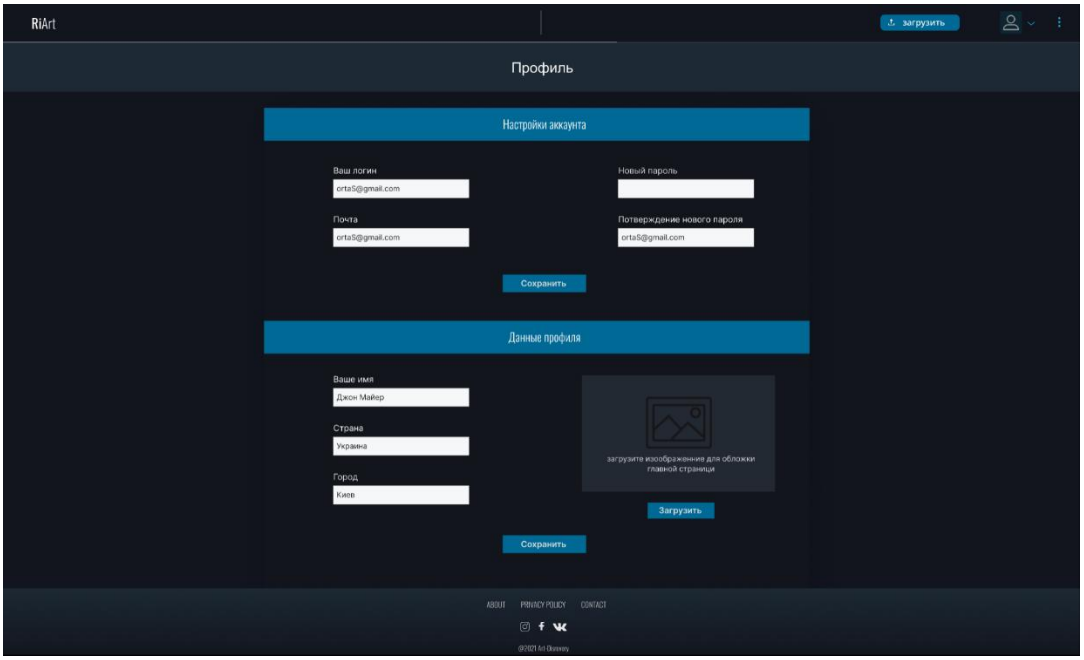


Рисунок Б.9 – Форма для заповнення профілю

Для загрузки матеріалів на вебсайт використовуються форми. У будь-якій із доступних форм для загрузки матеріалу, потрібно перетягнути потрібний матеріал чи натиснути кнопку загрузити, та вибрати потрібний матеріал на комп’ютері.

Вебсторінку для редагування портфоліо та публікації робіт зображено на рисунку Б.10.

Для публікації роботи потрібно заповнити усі елементи виділеної на рисунку Б.10 форми під номером 1.

Для видалення опублікованих робіт потрібно у формі виділеної на рисунку Б.10 під номером 2, з назвою “Видалення опублікованих робіт”, натиснути кнопку видали під потрібною опублікованою роботою.

Зауваження:

- якщо при публікації було обрано декілька робіт, то найперша робота у списку, вказаному на рисунку Б.10 під номером 1.1, буде виступати у вигляді попереднього перегляду для відображення публікації у галереї;
- до однієї публікації можна додати максимум 3 роботи;
- у формі для публікації робіт потрібно заповнити усі елементи.

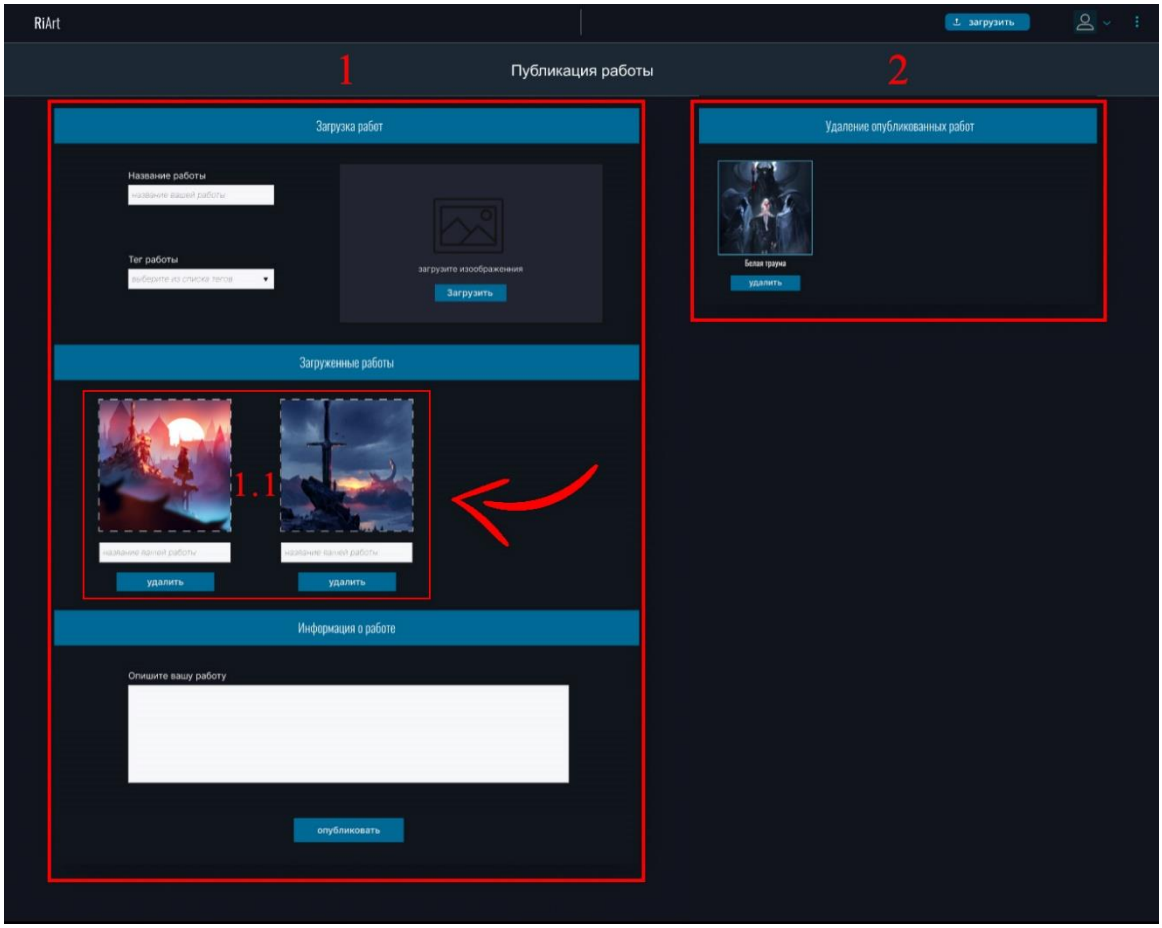


Рисунок Б.10 – Вебсторінка для редагування портфоліо та публікації робіт

Для повного перегляду публікації з роботами потрібно натиснути на роботу на головній вебсторінці галереї чи у портфоліо користувача. Відкриється вебсторінка з публікацією (рисунок Б.11).

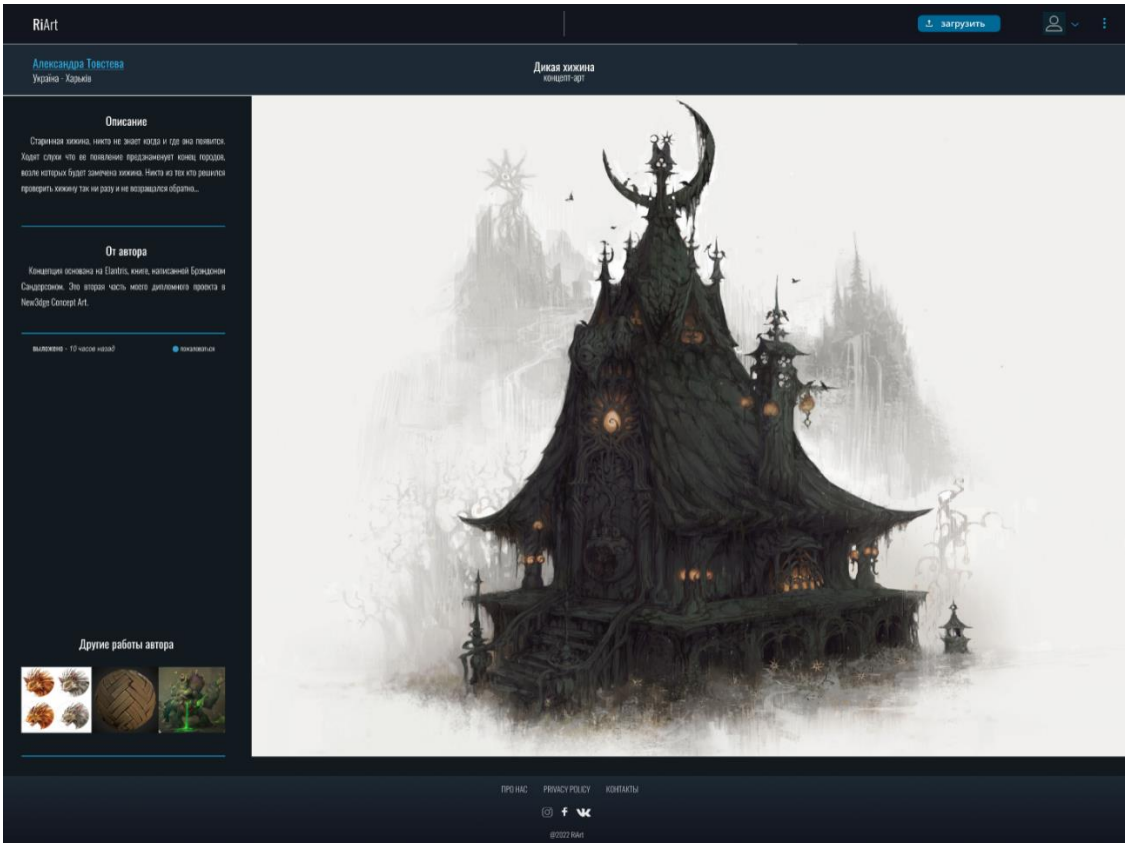


Рисунок Б.11 – Вебсторінка для перегляду публікації з роботами

Для авторизації користувача як адміністратора, йому видають спеціальні данні для входу до облікового запису адміністратора. Адміністратору надається спеціальна сторінка, зі списком публікацій та користувачів що їх опублікували у вигляді таблиці. До таблиці автоматично потрапляють публікації на які було подано скарги. Адміністратор має можливість заблокувати публікацію якщо вона не відповідає правилам вебсайту, натиснувши на червоний текст - заблокувати.

Сторінка адміністратору зображена на рисунку Б.12.

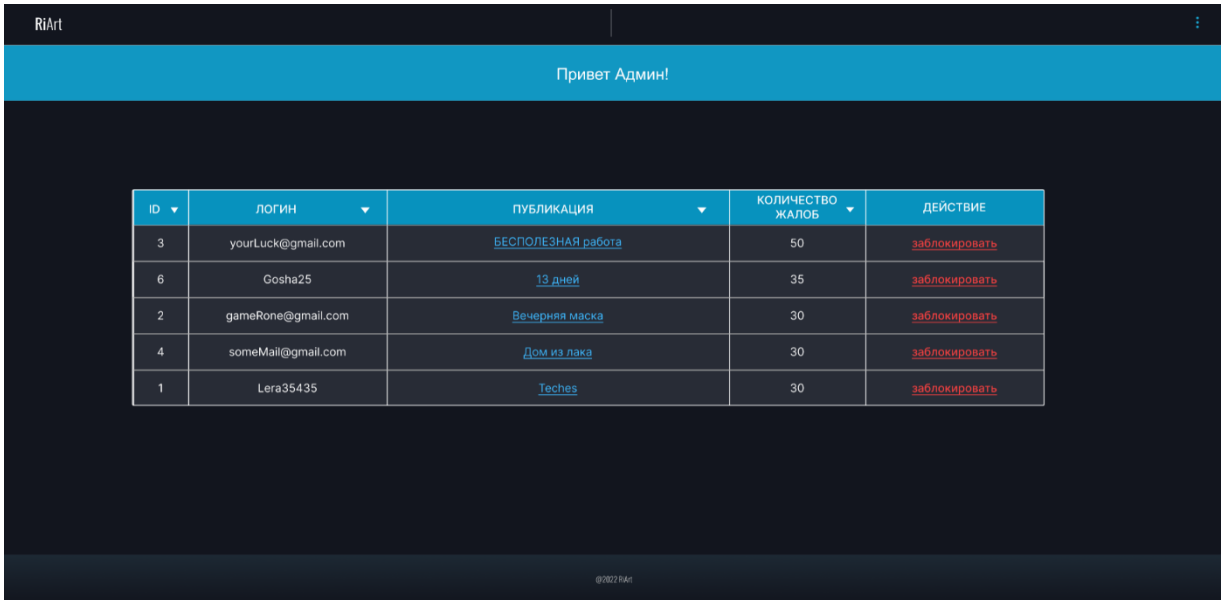


Рисунок Б.12 – Вебсторінка адміністратору

Зауваження щодо роботи адміністратору:

- адміністратор при переході на перегляд публікації буде вважатися гостем;
- декілька адміністраторів може бути авторизовано до облікового запису адміністратору;
- роботи які мають понад 30 скарг будуть автоматично поміщатися до таблиці адміністратору.

# ДОДАТОК В ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

## 1. Об’єкт випробувань

Об’єктом випробувань є програмне забезпечення вебсайту онлайн-галереї “RiArt”.

Областю застосування є дизайнерська сфера.

## 2. Мета випробувань

Метою проведення випробувань є перевірка роботи програмного забезпечення. Перевірка передбачає перевірку правильності виконання функцій ПЗ за встановленими вимогами в документації.

## 3. Вимоги до програми

При проведенні випробувань програмного забезпечення, функціональні характеристики повинні відповідати вимогам, які були викладенні у пункті “Постановка задачі”.

4. Вимоги до документації програми

До складу програмної документації входить:

- текст програми;
- інструкція користувача;
- програма та методика випробувань.

5. Склад та порядок випробувань

5.1 Склад технічних засобів

До складу технічних засобів, які мають використовуватися при випробуваннях належать:

- процесор: 12 поточний Amd Ryzen 5 2600;
- оперативна пам’ять: 16гб;
- відеоадаптер: Amd RX570 4гб;
- роздільна здатність екрану: 2080 x 1119 пікселей.
- інтернет зі швидкістю 10мбіт/с;

5.2 Склад програмних засобів

До складу програмних засобів, які мають використовуватися при випробуваннях належать:

- операційна система Windows 10;

- браузер Chrome версії 102;
- браузер Firefox версії 60;
- браузер Opera версії 61.

## 5.3 Порядок проведення випробувань

Порядок проведення випробувань:

- перевірка вимог до функціональних характеристик;
- виконання тестів в довільному порядку;
- аналіз отриманих результатів тестування;
- відповідність програмного продукту вимогам.

# 6. Методи випробувань

Методом випробувань для тестування розробленого програмного забезпечення було обрано метод чорної скриньки. Цей метод дозволяє перевіряти програму без прямого доступу до коду.

Перелік випробувань основних функціональних можливостей:

- авторизація гостя як клієнта;
- авторизація гостя як адміністратора;
- пошук робіт за фільтром;
- пошук робіт без використання фільтра;
- вихід з профілю.

В процесі випробувань необхідно перевірити правильність виконання визначених функціональних можливостей.

# ДОДАТОК Г ОПИС ПРОГРАМИ

## 1 Загальні відомості

### 1.1 Позначення і найменування програми

Найменування: Програмне забезпечення вебсайту для онлайн-галереї “RiArt”

Позначення: онлайн-галерея для дизайнерів

### 1.2 Програмне забезпечення, необхідне для функціонування програми

Програмне забезпечення розділено на серверну частину та клієнтську. Для повноцінної роботи програмного продукту нижче вказано технічні характеристики як для серверу так і клієнтської частини.

#### 1.2.1 Серверна частина

Сервер повинен мати наступні мінімальні технічні параметри:

- жорсткий диск – 100Мб вільного простору для можливості хостингу програмного забезпечення, та 250гб для подальшого розміщення закачаних клієнтами ресурсів;
- операційна система: Windows 8 та вище;

- доступ до виходу в мережу Internet;
- оперативна пам’ять: 2gb;
- процесор: 4 ядра.

### 1.2.2 Клієнтська частина

Клієнтська частина повинна мати будь який із браузерів наведених у програмних вимогах та доступ до виходу у мережу Internet.

## 1.3 Мови програмування

Для розробки програмного забезпечення вебсайту було використано фреймворк Asp.Net, реалізований на мові програмування C#, та СУБД Microsoft SQL Server.

Інтегроване середовище розробки що було використано для розробки – MS Visual Studio 2019.

# 2 Функціональне призначення

## 2.1 Класи розв’язуваних завдань та призначення програми

Призначенням розроблюваного програмного забезпечення є поліпшення процесу зберігання та розповсюдження інформації про дизайнерів.

Воно повинно реалізовувати наступні функції:

- авторизація;
- реєстрація;
- завантаження роботи на сайту;
- скачування роботи з сайту;
- публікація роботи на сайті;
- перегляд публікацій на головній сторінці сайту;
- пошук робіт;
- видалення публікації;
- блокування публікації клієнта;
- перегляд опублікованої роботи;
- перегляд профілю клієнта;
- редагування профілю клієнта;
- вихід з облікового запису.

## 2.2 Функціональні обмеження на застосування

Одним з найголовніших функціональних обмежень розробленого програмного забезпечення вебсайту для онлайн-галереї “RiArt” є необхідність підключення до мережі інтернету.

# 3 Опис логічної структури

## 3.1 Використовувані методи

При розробці програмного забезпечення вебсайту для онлайн-галереї “RiArt” та проведення його аналізу було використано об’єктноорієнтований метод та об’єктноорієнтовану мову програмування C#.

## 3.2 Алгоритм програми

Програмне забезпечення вебсайту виконано з використанням технології MVC. Вебсайт розділено на клієнтську частину та серверну. Клієнтом виступає браузер. Сервером виступає комп’ютер чи комп’ютери на яких знаходиться код програми та база даних.

Алгоритм взаємодії клієнту з сервером, та серверу з базою даних представлено надалі:

1) Користувач заходить на вебсайт у браузері.

2) Браузер відсилає запит на сервер.

3) Модуль IIS обробляє запит та передає його далі до вебсерверу Kestrel де запит оброблюється програмним забезпеченням.

4) Якщо потрібно, код на веб-сервер Kestrel відсилає запит до бази даних та отримує результат операції взаємодії с СУБД.

5) Програмне забезпечення формує відповідь для запиту, та відправляє від веб-серверу Kestrel до IIS.

6) Модуль IIS в свою чергу відправляє результат клієнту, тобто браузеру користувача.

## 3.3 Структура програми

Структуру програми представлено діаграмою розгортання на рисунку Г.1.

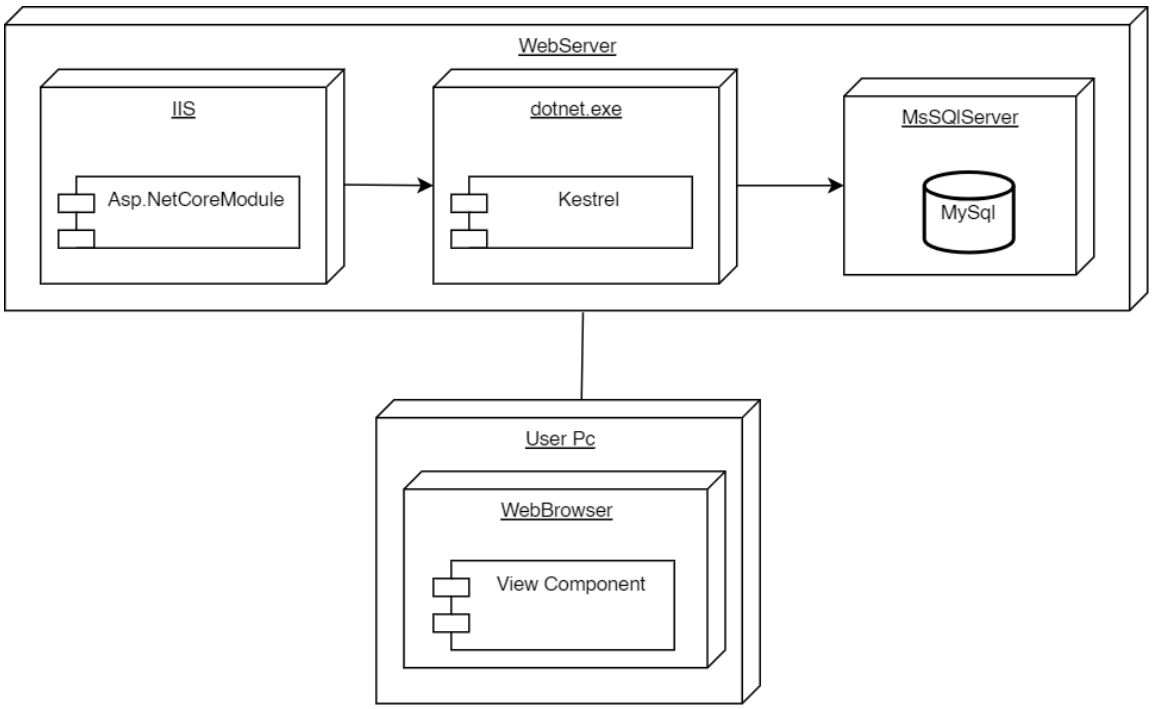


Рисунок Г.1 – Діаграма розгортання програмного забезпечення

### 3.4 Зв’язок програми з іншими програмами

Розроблене програмне забезпечення взаємодіє з MS SQL Server та модулем IIS у операційній системі Windows.

## 4 Виклик та загрузка програмного забезпечення

### 4.1 Спосіб виклику програми

Виклик програмного забезпечення відбувається після запуску програмного забезпечення на вебсервері хостингу. А активізація роботи відбувається після переходу на посилання вебсайту.

## 4.2 Вхідні точки в програму

Вхідною точкою в програмне забезпечення вебсайту для онлайн галереї “RiArt” є функція Main(). Через неї запускається програмне забезпечення через консоль або через модуль IIS.

Ця функція створює та налаштовує хоста для запуску серверу, та ініціює клас для подальшого налаштування маршрутизації на вебсайті.