МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ
Кафедра комп'ютерних наук та кібербезпеки

На правах рукопису

АРЖАНОВА АЛЬОНА СЕРГІЇВНА

РОЗРОБКА ВЕБПЛАТФОРМИ ДЛЯ ПІДТРИМКИ УКРАЇНСЬКИХ МИТЦІВ ТА ХУДОЖНИКІВ ЗАСОБАМИ ВЕБТЕХНОЛОГІЙ

Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки та інформаційні технології
Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр»

Науковий керівник:
ГЛИНЧУК ЛЮДМИЛА ЯРОСЛАВІВНА,
кандидат фізико-математичних наук, доцент
кафедри комп'ютерних наук та кібербезпеки

РЕКОМЕНДОВАНО ДО ЗАХИСТУ
Протокол №_____
засідання кафедри комп'ютерних наук
та кібербезпеки
від _____ 2025 р.
Завідувач кафедри

(_____) Гришанович Т. О.

ЛУЦЬК – 2025

ЗМІСТ

ВСТУП

Актуальність теми. Проблема доступності цифрових інструментів для українських митців залишається відкритою, попри активний розвиток переходу на цифрові технології. Художники часто не мають ефективних засобів для представлення своїх робіт онлайн, налагодження зв'язків з аудиторією та взаємодії з культурними інституціями. У відповідь на ці виклики розробка вебплатформи набуває особливого значення як спосіб підтримки творчої спільноти та популяризації українського мистецтва. У контексті оцифрування та культурного спротиву платформа може виконувати не лише функцію представлення, а й сприяти збереженню національної ідентичності. Це зумовлює актуальність дослідження, що поєднує інструменти веброзробки з потребами креативного сектору.

Мета дослідження: створення зручної та технологічно актуальної вебплатформи, що дозволяє українським митцям ефективно презентувати свої роботи онлайн, взаємодіяти з аудиторією та культурними організаціями. Акцент зроблено на використанні засобів веброзробки: фреймворків, систем автентифікації та оптимізації роботи з даними для вирішення прикладної задачі у сфері цифрової культури. До основних **завдань** варто віднести наступні:

- удосконалити технічну архітектуру вебплатформи на основі результатів курсової роботи з урахуванням принципів масштабованості, модульності та безпеки інформації;
- доопрацювати і реалізувати функціональні компоненти програмної частини, зокрема: інтерактивний чат у реальному часі на базі WebSocket (SignalR), систему сповіщень, механізми автентифікації та авторизації з використанням JWT;
- розширити логіку взаємодії з базами даних з урахуванням вимог до швидкодії, узгодженості даних та підтримки запитів до фільтрованого контенту;
- оптимізувати REST API, забезпечивши ефективну взаємодію між

клієнтською та серверною частинами, а також спростивши масштабування та супровід системи;

- реалізувати динамічне управління контентом, що дозволить митцям додавати, редагувати та видаляти власні матеріали, а відвідувачам – здійснювати пошук, фільтрацію та перегляд згідно з категоріями;
- удосконалити інтерфейс користувача, за потреби адаптувавши його до сучасних стандартів доступності (accessibility), ґрунтуючись на результатах попереднього аналізу цільової аудиторії, проведеного в курсовій роботі.
- виконати комплексне тестування системи та оцінити її придатність до подальшого розгортання.

Об’єкт дослідження: процес розробки та вдосконалення вебплатформи для підтримки українських митців у цифровому середовищі, зосереджений на поглибленій інтеграції вебтехнологій задля підвищення функціональності, масштабованості, безпеки та якості користувацького досвіду. Увага приділяється створенню умов для ефективної презентації творчості, зручної взаємодії між митцями та аудиторією, а також розширенню можливостей для монетизації та збереження культурного контенту.

Розробка такої платформи розглядається як технологічно обґрунтований крок до сталого розвитку цифрової інфраструктури у сфері культури, що дозволяє не лише демонструвати мистецтво, а й формувати відкриту спільноту навколо української творчості в глобальному інформаційному просторі..

Предмет дослідження: створення та вдосконалення технологічної підтримки вебплатформи “ArtUA” полягає у використанні вебтехнологій, фреймворків та інструментів для забезпечення повного функціоналу публікації, взаємодії та поширення креативного контенту. Предмет дослідження включає розробку механізмів управління контентом, реалізацію механізму авторизації та автентифікації, інтеграцію баз даних та даних і безпеки користувачів, а також використання хмарних сервісів та API для оптимізації та масштабування

платформи.

У фокусі реалізація динамічної функціональності платформи, інтеграція мультимедійних елементів, впровадження систем пошуку та фільтрації контенту, а також аналіз фреймворків і інструментів для забезпечення стабільної роботи та високої продуктивності на всіх етапах життєвого циклу системи.

Апробація результату роботи. Результати дослідження опубліковані у вигляді тез на тему “Використання ASP.NET та React.js для створення платформи підтримки українських митців”, які увійшли до збірника матеріалів II Міжнародної науково-практичної конференції «Проблеми комп’ютерних наук, програмного моделювання та безпеки цифрових систем». Конференція проходила 9-11 червня 2025 року на базі практик табору “Гарт” Волинського національного університету імені Лесі Українки (Шацький район, с. Світязь).

РОЗДІЛ 1
ПІДХОДИ ТА ТЕХНОЛОГІЇ ДЛЯ ПІДТРИМКИ МИТЦІВ В ЦИФРОВОМУ СЕРЕДОВИЩІ

1.1. Вплив цифрових платформ на розвиток культури та мистецтва

1.1.1. Роль онлайн-ресурсів у популяризації мистецьких проектів

Цифрові технології фундаментально змінили процеси виробництва, поширення та сприйняття мистецтва. Цифрові архіви, віртуальні музеї, вебсайти та соціальні мережі зараз є незамінними інструментами для митців та їхньої аудиторії. На відміну від традиційних медіа, цифрові медіа забезпечують доступ до контенту в режимі реального часу, ширше охоплення аудиторії та інтерактивну взаємодію.

Цифрова культура включає твори, створені в цифровому контексті, та оцифровані форми традиційних артефактів. Технологія сканування дозволяє точно зберігати об'єкти та забезпечує доступ незалежно від місцезнаходження. Чудовим прикладом є портал Europeana, який об'єднує мільйони цифрових реплік предметів мистецтва, архітектурних планів, документів та аудіовізуальних матеріалів для досліджень та цілей публічного доступу.

Платформи соціальних мереж, такі як Instagram, TikTok, YouTube та Facebook, пропонують художникам місця для демонстрації своїх робіт, отримання конструктивної критики та створення спільнот. Цифрові платформи, такі як Behance або ArtStation, пропонують можливості для демонстрації портфоліо та сприяння професійній взаємодії. Kickstarter та Patreon також пропонують фінансову підтримку мистецьким підприємствам через краудсорсингові кампанії [1].

Вебсайти, такі як Google Arts & Culture, надають можливість переглядати твори мистецтва у високій роздільній здатності, здійснювати віртуальні відвідування та взаємодіяти з культурною спадщиною через онлайн-медіа.

З іншого боку, інтелектуальна власність є ще більш проблематичною з цифровізацією. Технологія блокчейн та NFT забезпечують автентичність та

прозорість цифрового мистецтва.

Віртуальні ресурси є надзвичайно важливими для розширення доступу до мистецтва, сприяння новим способам взаємодії та збереження культурної спадщини для майбутніх поколінь [2].

1.1.2. Створення цифрових спільнот для підтримки митців

Цифрові платформи революціонізували спосіб взаємодії митців зі своєю аудиторією, пропонуючи нові механізми для демонстрації їхніх робіт, створення співпраці, отримання доходів та захисту прав інтелектуальної власності. Інтернет дозволяє митцям брати участь у культурних дискусіях, створювати спільноти та отримувати підтримку незалежно від географічного розташування.

Онлайн-спільноти дозволяють здійснювати глобальну співпрацю та створювати спільні проекти, і все це за допомогою інструментів Web 2.0 та хмарних сервісів. Такі вебсайти, як DeviantArt, Behance та ArtStation, дозволяють обмінюватися інформацією та просувати креативні продукти [3].

Соціальні мережі дозволяють створити особистий бренд митця, зображуючи процес створення творів мистецтва. Відеоплатформа ідеально підходить для просування та взаємодії з аудиторією. Таким чином, віртуальні спільноти стали засобом для просування професійного зростання та сприяння глобальній інтеграції митців у сучасний світ мистецтва.

1.2. Аналіз потреб митців у цифровому середовищі та їх взаємодія з онлайн-ресурсами

Серед цифрових інновацій митці стикаються з новими вимогами, які вимагають від них адаптації своїх робіт до цифрового світу. Однією з найбільших вимог є створення та управління особистим цифровим “слідом”, де вони можуть просувати свої роботи та забезпечувати свою видимість. Дослідження показали, що успіх митця значною мірою залежить від того, наскільки вміло він використовує онлайн-простір для просування своїх творінь та взаємодії зі своєю аудиторією. Отже, інструменти для побудови особистого

бренду та активна присутність в Інтернеті набувають великого значення.

Крім того, митцям потрібні професійні мережі, а також засоби обміну досвідом. Онлайн-мережі пропонують засоби, що дозволяють співпрацювати між професіоналами в культурному та креативному секторах, а отже, легше обмінюватися продуктивними ініціативами та підтримкою.

Не менш важливою є необхідність набуття навичок роботи з новими цифровими інструментами та технологіями. Онлайн-освіта дозволяє митцям розвивати можливості самоврядування, вивчати юридичну та фінансову сторону своєї професії та ефективно просувати себе. [4]

Крім того, митці намагаються включати нові технології, такі як доповнена реальність (AR), у свої твори. Завдяки цьому включенню уможливлюється розширення традиційних мистецьких меж та створення інтерактивних презентацій, які можна представити широкій аудиторії.

Також актуальною є необхідність захисту та поширення культурної спадщини через цифрові медіа. Використання міждисциплінарного підходу в поєднанні з новими методами підвищує ефективність вивчення, збереження та передачі матеріальних та нематеріальних компонентів культурної спадщини. Таким чином, для того, щоб митці досягли успіху у своїй професійній практиці в онлайн-середовищі, важливо, щоб вони були активними в Інтернеті, стали компетентними у використанні технологій та стали частиною професійних мереж. Така участь значно пришвидшує їхній мистецький розвиток та підвищує їх загальний авторитет. [5]

1.3. Архітектурні принципи побудови вебплатформ для підтримки користувацької взаємодії

Розробка вебплатформ з акцентом на взаємодії з користувачем вимагає врахування різних архітектурних принципів, що забезпечують надійність, продуктивність та безпеку. Зі швидким розвитком цифрових послуг та зростанням очікувань користувачів, архітектура системи повинна бути гнучкою та здатною враховувати майбутні вдосконалення.

В основі більшості таких платформ лежить клієнт-серверна архітектура, яка базується на поділі системи на два компоненти – frontend та backend. Клієнтська сторона (frontend) обробляє інтерфейс користувача та загальну зручність використання, тоді як обробка запитів, взаємодія з базою даних та реалізація бізнес-логіки виконуються на стороні сервера (backend). Ця модель дозволяє відокремити логіку представлення даних від логіки обробки даних, тим самим значно покращуючи розробку, масштабованість та підтримку системи, що детально розглядалось в попередньому дослідженні (курсовій роботі) [6].

Одним із найпоширеніших архітектурних стилів для взаємодії клієнт-сервер є архітектура REST, або передача репрезентативного стану. Ця структура забезпечує ефективний обмін даними на основі стандартних HTTP-запитів та дотримується принципу бездержавності, за яким кожен запит до сервера містить всю інформацію, необхідну для його обробки сервером. Використання REST забезпечує ефективну масштабованість системи, просту інтеграцію із зовнішніми сервісами та збереження архітектурної модульності [7].

Серед ключових особливостей розробки архітектури вебсистем можна виокремити використання REST API, інтерфейсу прикладного програмування, який дозволяє стандартний доступ до системних ресурсів за допомогою звичайних методів HTTP (GET, POST, PUT, DELETE). REST API заохочує чітко визначений формат для зв'язку клієнт-сервер, що спрощує процеси розробки, тестування та подальшого обслуговування програмних продуктів [8].

У сьогоднішніх програмних рішеннях для створення REST API часто застосовується фреймворк ASP.NET Core – багатоплатформна, високопродуктивна платформа, що підтримує модульний підхід до проєктування. Контролери в ASP.NET Core використовуються для обробки HTTP-запитів, вилучення параметрів з URL-адреси або тіла запиту та форматування відповідей у форматі JSON, який широко прийнятий як стандарт для обміну даними у вебдодатках [9].

Однією з ключових переваг ASP.NET Core є те, що він може забезпечити

відмінну продуктивність та масштабованість, а також хорошу інтеграцію з механізмами автентифікації та авторизації, зокрема завдяки використанню JSON Web Tokens (JWT). Це дозволяє безпечно передавати дані без необхідності підтримки стану на сервері, як це передбачено REST. Крім того, ASP.NET Core пропонує підтримку Entity Framework Core, нового фреймворку об'єктно-реляційного відображення (ORM), який спрощує роботу з реляційними базами даних, дозволяє керувати схемами даних, підтримує виконання складних запитів та оптимізує дані [10].

Використання ASP.NET Core в архітектурі REST дозволяє створювати гнучкі, масштабовані та керовані вебсервіси, які можуть ефективно виступати посередником між розробкою на стороні клієнта з використанням фреймворків JavaScript (наприклад, React) та логікою сервера. Це мінімізує проблеми заплутаного архітектурного шаблону, масштабованості системи та інтеграції зі сторонніми сервісами, що є ключовими міркуваннями при розробці складних програмних продуктів [11].

Виходячи з вимоги до високої швидкості відгуку платформ та скорочення часу очікування відповідей, посилена увага приділяється сприянню взаємодії в реальному часі. WebSocket є одним із найновіших рішень для цього, оскільки він дозволяє відкривати постійний двонаправлений канал зв'язку між клієнтом та сервером. Це дозволяє розробляти динамічну функціональність, включаючи обмін миттєвими повідомленнями та сповіщення про оновлення системи [12].

Загалом, архітектурні принципи розробки вебплатформ повинні відповідати динамічній природі онлайн-середовища, сприяючи гнучкості, адаптивності та безперебійній взаємодії між усіма зацікавленими сторонами на платформі. Застосування цих принципів дозволяє створити технологічну основу, яка не лише вирішує поточні проблеми, але й готова до майбутнього розвитку системи.

1.4. Технології забезпечення інтерактивності у вебплатформах

Забезпечення інтерактивної взаємодії в режимі реального часу є ключовою

вимогою вебплатформ. Традиційна модель “HTTP-запит – відповідь” не дозволяє ефективно реалізувати двосторонню комунікацію між клієнтом і сервером, оскільки клієнт змушений постійно ініціювати запити для отримання нових даних. Для подолання цього обмеження було розроблено протокол WebSocket, що забезпечує постійне двостороннє (full-duplex) з’єднання поверх TCP. Це дозволяє обом сторонам обмінюватися даними в режимі реального часу без необхідності повторного встановлення з’єднання, значно знижуючи затримки і підвищуючи продуктивність у порівнянні з технологіями на кшталт Long Polling, де клієнт постійно робить нові HTTP-запити.

Протокол WebSocket підтримується більшістю браузерів і дозволяє передавати як текстові, так і бінарні дані. Метод send() API WebSocket приймає формати ArrayBuffer, Blob, TypedArray або DataView, що дає змогу передавати медіа, файли та інші складні об’єкти. Завдяки цьому WebSocket підходить для створення високонавантажених систем, де важлива не лише швидкість, а й різноманітність типів даних, що передаються [13].

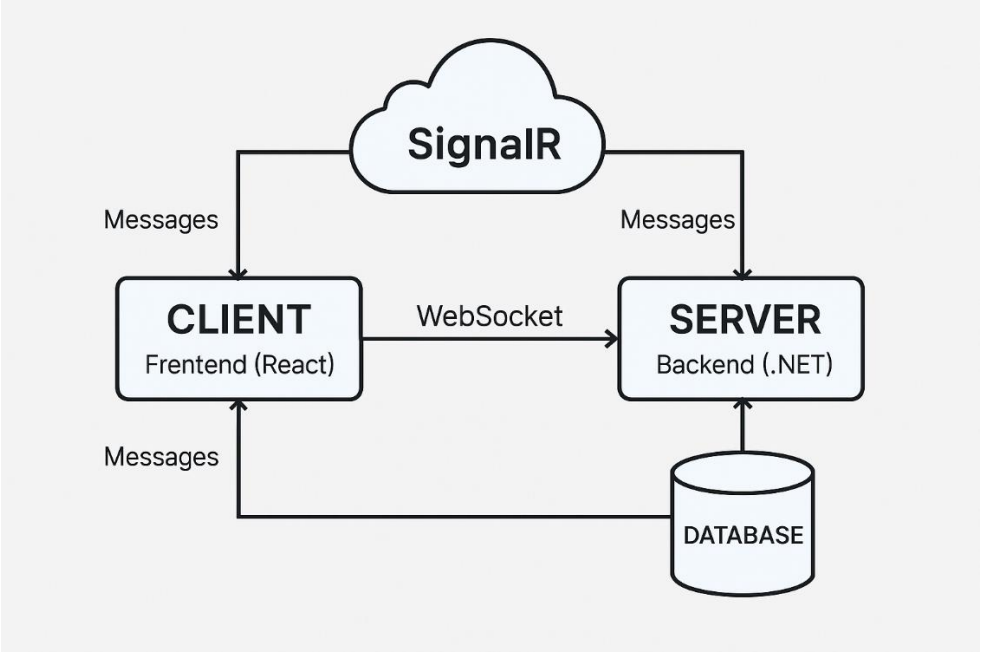

Рисунок 1.1 – Архітектура інтерактивної взаємодії з використанням WebSocket і SignalR

Для спрощення розробки двосторонньої комунікації в екосистемі .NET

застосовується бібліотека SignalR. SignalR є високорівневою абстракцією над WebSocket, яка автоматично переключається на альтернативні транспортні механізми (Server-Sent Events, Long Polling) у випадку, якщо WebSocket недоступний. SignalR реалізує концепцію хабів – спеціальних серверних компонентів, через які клієнти можуть підписуватись на події, надсилати і отримувати повідомлення в режимі реального часу. Це значно спрощує розробку інтерактивних вебдодатків із масштабованою архітектурою [14].

Використання WebSocket разом із бібліотеками, що його інкапсулюють, дозволяє розробляти адаптивні та розширювані рішення для обробки подій у реальному часі. Обидві технології знаходять широке застосування на вебсайтах, орієнтованих на високу залученість користувачів, таких як сайти соціальних мереж, додатки фондового ринку, вебсайти онлайн-консалтингу, багатокористувацькі ігри та численні інші додатки, де своєчасність обміну інформацією є важливою.

1.5. Роль соціальних мереж у просуванні та формуванні мистецьких спільнот

Соціальні мережі є важливою платформою для продажу цифрових творів мистецтва та для побудови професійних зв'язків. Instagram, Facebook, TikTok та Twitter надають митцям канали комунікації, через які вони можуть демонструвати свої роботи, взаємодіяти зі своєю аудиторією, професійно спілкуватися та розвивати свою творчість.

Instagram, будучи візуальною платформою соціальних мереж, тим не менш, є життєво важливою платформою для митців. Понад 80% користувачів мають бізнес-акаунти, включаючи облікові записи, що належать митцям та галереям [15]. Facebook активно використовується в групах, наприклад, «Art Marketing Mastery» має понад 50 000 учасників, які обговорюють, як просувати та продавати мистецтво [Facebook Groups, 2023].

Режими коротких відео в TikTok та Instagram Reels дозволяють поширювати вірусний контент. Митці можуть залучити велику аудиторію, що

впливає на продажі. Наприклад, цифрова художниця Аманда Олеандер отримала понад мільйон підписників та стабільні продажі після своєї популяризації в TikTok [16].

Згідно зі звітом Art Basel та UBS Global Art Market Report 2022, близько 73% майбутніх митців використовують Instagram як робочий інструмент для зв'язку з галереями та колекціонерами [17]. Коментування з повідомленнями, прямі трансляції та спільна робота поглиблюють занурення у творчу спільноту.

Економічний аспект також є важливим, оскільки соціальні мережі дозволяють продавати твори мистецтва через такі опції, як Instagram Shopping та Facebook Marketplace. Онлайн-продажі творів мистецтва та антикваріату зросли з 3,75 мільярда доларів у 2013 році до 11,8 мільярда доларів у 2023 році [18].

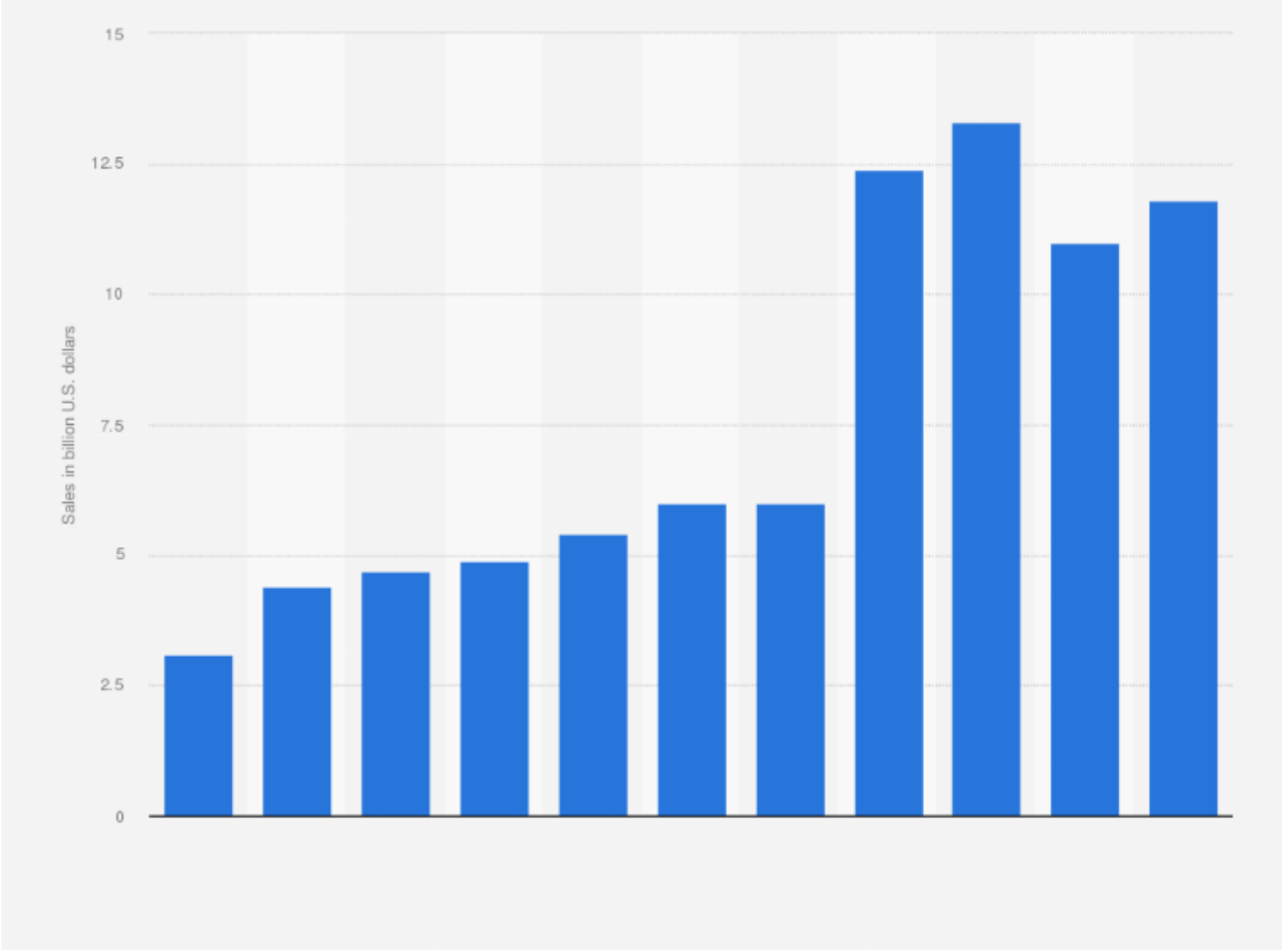

Рисунок 1.2 – Онлайн-продажі мистецтва та антикваріату у світі (2013–2023)

Соціальні мережі суттєво трансформували взаємодію митців з аудиторією, сприяючи популяризації, формуванню спільнот і монетизації творчості.

Платформи як Instagram, TikTok, Twitter і Pinterest забезпечують незалежним художникам прямий доступ до глядачів, минаючи галереї чи агентів. Згідно зі звітом Artsy (2023), понад 60% молодих митців отримують перші замовлення саме через соціальні мережі [19].

Соціальні платформи також сприяють утворенню професійних спільнот. Наприклад, Facebook-групи та Reddit-спільноти, як “Art Business” і “Independent Artists”, об’єднують тисячі учасників для обміну досвідом, обговорення технік і пошуку однодумців [20].

Таким чином, цифрові платформи виступають ключовим інструментом у професійному розвитку художників, забезпечуючи видимість, зв’язки та комерційні можливості на глобальному рівні.

1.6. Оптимізація продуктивності вебплатформ

1.6.1. Швидкість завантаження

Час завантаження сторінки вебсайту є визначальним фактором якості взаємодії з користувачем та ефективності використання ресурсів. 53% мобільних користувачів залишать сайт, якщо він завантажується більше ніж за 3 секунди, стверджує Google [1]. Дослідження Akamai показують, що одна додаткова секунда завантаження знижує конверсію на 7%, а затримка в 1 секунду знижує задоволеність користувачів на 16% [21].

Для підвищення швидкості використовуються оптимізація медіа, зменшення кількості HTTP-запитів, кешування, використання CDN та оптимізація коду. Формати WebP та AVIF дозволяють оптимізувати зображення, щоб значно зменшити розмір файлу без втрати якості. Технологія лінивого завантаження допомагає забезпечити завантаження зображень лише тоді, коли вони знаходяться в полі зору, зменшуючи початкове завантаження.

Мінімізація HTTP-запитів здійснюється шляхом об'єднання CSS та JavaScript, групування зображень та вбудовування критичних ресурсів. Кешування на стороні клієнта (браузера) та кешування на стороні сервера (Redis,

Memcached) економить завантаження та прискорює перезавантаження.

CDN доставляє копії контенту на географічно розподілені сервери, що призводить до менших затримок та меншого часу завантаження [21]. Мінімізація коду за допомогою скриптів, таких як UglifyJS та CSSNano, а також асинхронне та ліниве завантаження скриптів покращують рендеринг сторінок [22].

Для оцінки продуктивності вебплатформ використовуються різноманітні інструменти:

- Google PageSpeed Insights – аналізує швидкість завантаження та надає рекомендації з оптимізації.
- Lighthouse – комплексний інструмент аудиту продуктивності, доступності та SEO, інтегрований у Chrome DevTools.
- GTmetrix – дозволяє оцінити ефективність сторінки, враховуючи показники Web Vitals.

Оптимізація швидкості завантаження є важливим етапом у розробці вебплатформ, оскільки вона напряму впливає на користувацький досвід та бізнес-результати. Використання методів, таких як впровадження CDN, кешування ресурсів та мінімізація HTTP-запитів, дозволяє значно покращити продуктивність сайту.

1.6.2. Зручність використання

Зручність використання (usability) є найважливішою сферою розробки програмного забезпечення та вебплатформ, що впливає на якість користувацького досвіду. Основними вимогами є швидкість та здатність діяти без помилок і без складнощів у роботі з інтерфейсом. Інтерфейс має бути зрозумілим для користувачів різного рівня кваліфікації.

Навігація має бути простою, з швидким доступом до інформації за допомогою ретельно розроблених меню, елементів керування та фільтрів. Зменшення кількості помилок та представлення чітких повідомлень про них, як зазначає Nielsen, знижує стрес користувача та максимізує впевненість [23].

Інтерфейс має бути адаптивним до різних пристроїв (ПК, смартфонів, планшетів) та доступним за стандартами (наприклад, підтримка програм зчитування з екрана, налаштування шрифту).

Час відгуку є ключовим, як показують дослідження Google, 53% користувачів залишають сторінку, якщо вона завантажується довше 3 секунд, що робить оптимізацію продуктивності надзвичайно важливою.

Простота використання також включає інтеграцію з іншими сервісами, тобто сприяння соціальним мережам для спрощення автентифікації та взаємодії [24]. Навігація, доступність та продуктивність, відповідно, формують основу зручності використання, яка визначає успіх платформи.

1.6.3. Інформаційна безпека

Інформаційна безпека є критичним компонентом функціонування онлайн-платформ, особливо при роботі з персональними даними. Відсутність належного захисту підвищує ризик несанкціонованого доступу, витоку інформації та фішингових атак.

Захист конфіденційності користувачів забезпечується через шифрування даних під час передачі (SSL/TLS) і зберігання (наприклад, AES-256). Це дозволяє запобігти перехопленню або втраті інформації.

Аутентифікація є основою контролю доступу. Найбільш ефективною є двофакторна автентифікація (2FA), яка додає додатковий рівень перевірки особи (через SMS, email або біометрію), знижуючи ризик компрометації навіть за наявності вкраденого пароля.

Контроль доступу повинен реалізовуватись згідно з ролевою моделлю. Адміністратори отримують розширені повноваження, тоді як звичайні користувачі – лише доступ до власних даних. Такий підхід мінімізує внутрішні загрози та забезпечує цілісність інформації [25].

Не менш важливим є захист від кібератак, які можуть призвести до витоку інформації або відмови системи. Найважливіші загрози:

– фішинг – вид атаки, коли зловмисники маскуються під довірених осіб

або організації, щоб змусити користувачів розкрити свої паролі чи іншу конфіденційну інформацію;

- DDoS-атаки – метод злому, при якому хакери надсилають велику кількість запитів на сервер, перевантажуючи його та спричиняючи відмову в обслуговуванні. Для захисту від таких атак використовуються фільтри трафіку та спеціальні інструменти, що автоматично блокують підозрілу активність;
- шкідливе програмне забезпечення – може бути використане для крадіжки даних або віддаленого контролю над пристроєм користувача.

Для хмарних технологій захист даних має бути пріоритетом, оскільки багато платформ зберігають дані в хмарі. Безпека забезпечується за допомогою багаторівневого шифрування, регулярного резервного копіювання та дотримання моделі Zero Trust, яка передбачає постійну перевірку прав доступу навіть для законних користувачів [26].

Дотримання нормативних вимог, таких як GDPR (ЄС) та CCPA (США), також є обов'язковим для платформ, що обробляють персональні дані. GDPR (ЄС) та CCPA (США) керують процесом збору, зберігання та обробки даних, а також підтримують контроль над даними користувачів. Відповідність також підвищує рівень довіри та правову безпеку [27].

Ефективна платформа інформаційної безпеки повинна включати регулярне сканування систем на наявність вразливостей, зокрема за допомогою етичного хакінгу (penetration testing). Такий підхід дозволяє виявляти потенційні загрози ще до їх експлуатації зловмисниками. Регулярні аудити та оновлення політик безпеки забезпечують підтримку стабільного рівня захисту. Розглянемо кілька кейсів, які ілюструють наслідки недотримання заходів безпеки:

- витік даних у Facebook (2019) через недоліки в захисті API та відсутність належного контролю доступу до баз даних було розкрито дані понад 533 млн користувачів;
- кібератака на Colonial Pipeline (2021) зловмисники отримали доступ до

системи через скомпрометований VPN-акаунт без багатофакторної аутентифікації, що призвело до зупинки роботи трубопроводу та значних фінансових збитків;

– витік даних у Marriott International (2020) через компрометацію облікових записів співробітників та відсутність багаторівневого контролю доступу було розкрито персональні дані 5,2 млн гостей [28].

Ці інциденти підтверджують необхідність цілісного підходу, який включатиме технічні рішення в поєднанні з політикою щодо поведінки персоналу.

Дослідження показує, що автоматизовані системи виявлення загроз на основі штучного інтелекту здатні знизити вартість інцидентів в середньому на 2,2 мільйона доларів. Компанії, які не використовують автоматизовані засоби кіберзахисту, несуть вищі витрати на відновлення функцій систем.

1.7. Огляд існуючих платформ для підтримки художників та митців

Назва: Spilne Art.

Розробник: Наталія Ткаченко.

Архітектура: вебплатформа.

Мова реалізації: WordPress.

Перелік функцій:

– каталогізація та презентація новітнього українського мистецтва;
– продаж мистецьких робіт онлайн;
– створення культурного діалогу між художниками та поціновувачами мистецтва;
– проведення аналітичних досліджень артринку та тенденцій розвитку мистецтва.

Переваги:

– сприяння популяризації українського мистецтва на міжнародному рівні;

– забезпечення прямого зв'язку між художниками та покупцями;

– наявність аналітичних матеріалів щодо стану та перспектив арт ринку.

Недоліки платформи включають обмежену інформацію про технічні аспекти її розробки та команду, а також можливу залежність від інтернет-з'єднання для доступу до контенту.

Джерело інформації: https://spilne.art

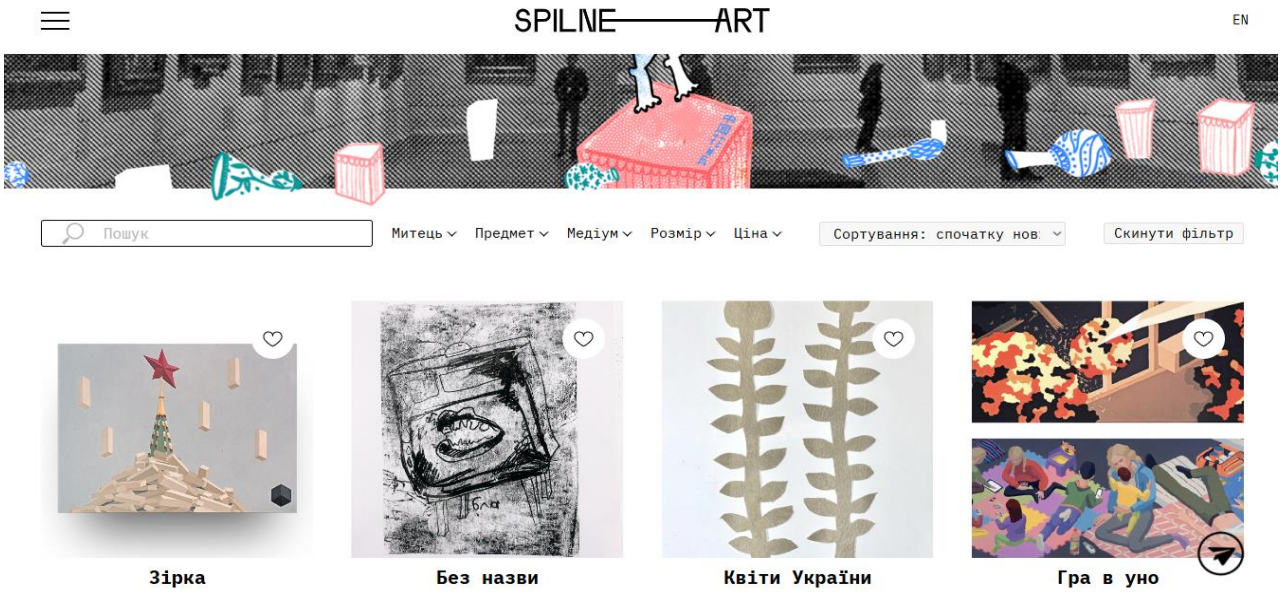

Рисунок 1.3 – Скріншот програмного продукту Spilne Art

Назва: IMPULSE TRANSFORMATION PLATFORM

Розробник: Міжнародний благодійний фонд “Імпульс Трансформація Платформа”.

Архітектура: вебплатформа.

Мова реалізації: WordPress, PHP, MySQL.

Перелік функцій:

– підтримка створення та презентації нових творчих проєктів у сфері перформативного мистецтва (танець, театр, новий цирк, перформанс та музика);

– надання екстреної фінансової допомоги митцям у скрутних життєвих обставинах;

– організація та участь у міжнародних культурних заходах для

популяризації українського мистецтва за кордоном;

– сприяння відновленню та розвитку культурних ініціатив в Україні через мистецтво та культурні практики.

Переваги:

– комплексна підтримка митців та діячів культури, особливо в умовах кризових ситуацій;
– активна участь у міжнародних культурних заходах, що сприяє популяризації українського мистецтва на світовій арені;
– гнучкість у реагуванні на актуальні потреби мистецької спільноти, зокрема через створення спеціальних фондів підтримки.

Недоліки включають обмежену інформацію про технічні аспекти платформи та її розробників, а також залежність від зовнішнього фінансування та пожертвувань, що може впливати на стабільність реалізації проєктів.

Джерело інформації: https://icfitp.org.ua

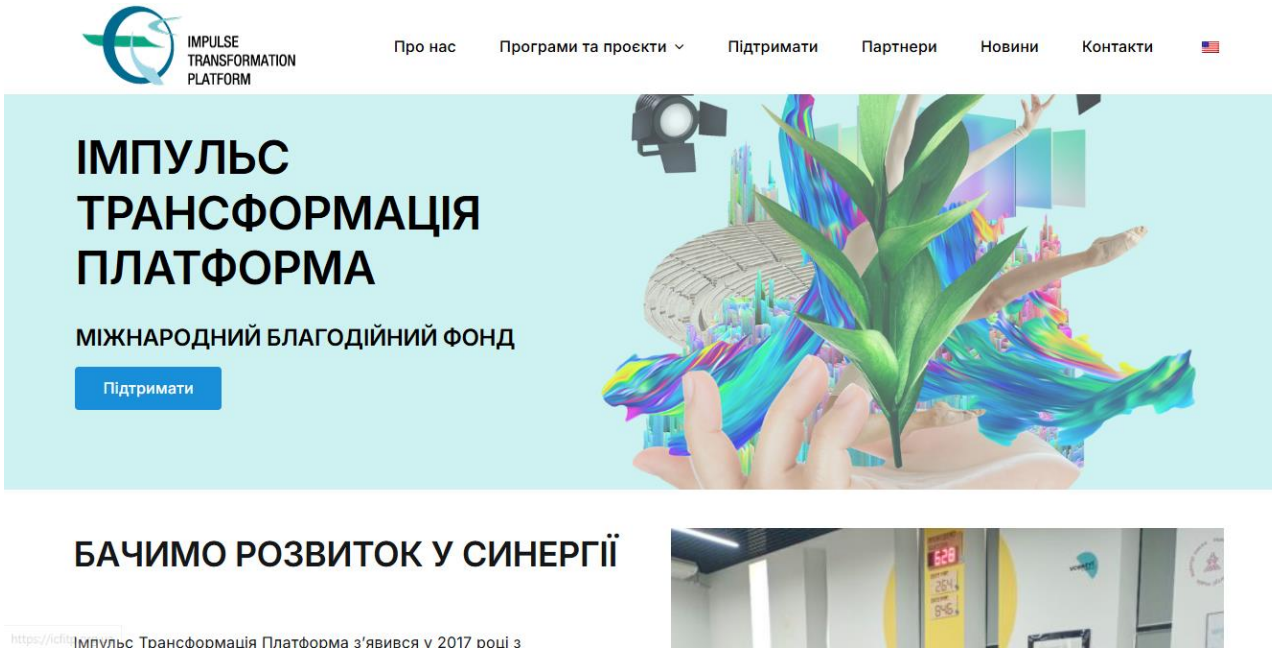

Рисунок 1.4 – Скріншот програмного продукту IMPULSE TRANSFORMATION PLATFORM

Назва: Jose Art Gallery.

Розробник: Jose Art Gallery Team.

Архітектура: онлайн-галерея.

Мова реалізації: Shopify, Magento.

Перелік функцій:

- продаж та купівля творів мистецтва онлайн через інтуїтивно зрозумілий інтерфейс;
- можливість створення персонального профілю для митців із портфоліо та біографією;
- взаємодія між художниками, колекціонерами та поціновувачами мистецтва через коментарі, рейтинги та обговорення робіт;
- проведення виставок та аукціонів у цифровому форматі, що дозволяє купувати твори мистецтва з будь-якої точки світу;
- просування художників через спеціальні маркетингові програми, що включають рекламу на платформі, email-розсилки та участь у тематичних добірках.

Переваги:

- поєднує функції онлайн-галереї та соціальної мережі та взаємодіяти з іншими митцями та поціновувачами мистецтва;
- широкий спектр категорій, включаючи живопис, графіку, фотографію, цифрове мистецтво та скульптуру;
- організовує конкурси для художників, надаючи можливість отримати визнання та розширити свою аудиторію.

Недоліки:

- певні обмеження у безкоштовному використанні платформи (можливо, платний доступ до розширених функцій);
- конкуренція серед великої кількості художників, що може ускладнювати просування.

Джерело інформації: https://joseartgallery.com/uk

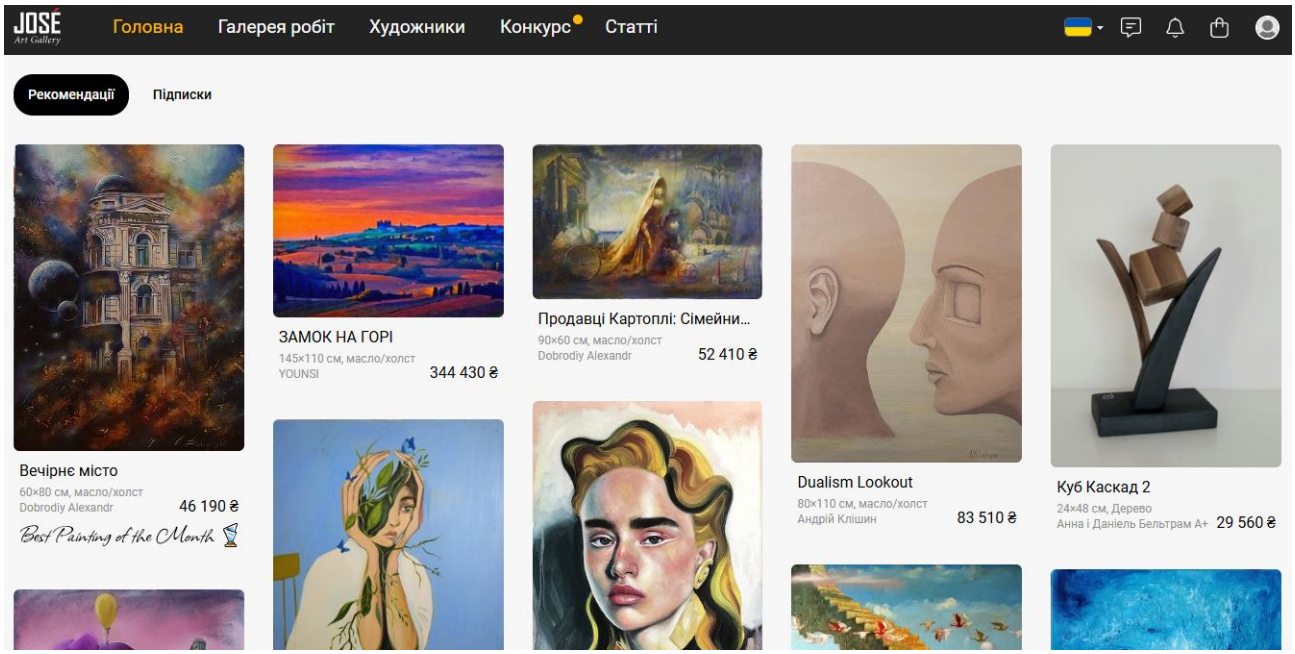

Рисунок 1.5 – Скріншот програмного продукту Jose Art Gallery

Порівняльний аналіз аналогічних платформ дозволив сформувати чітке уявлення про функціональні можливості, архітектурні підходи та технічні обмеження, характерні для вебрішень у сфері представлення мистецького контенту. Досвід вивчених систем вказав на необхідність забезпечення простоти у користуванні, підтримки персоналізованих профілів користувачів, можливості інтерактивної взаємодії між митцями та аудиторією, а також важливість включення інструментів для аналітики. Окрему увагу було виділено на недоліки, зокрема відсутність прозорості технічної реалізації та залежність від зовнішніх ресурсів, що було враховано при проєктуванні власного рішення. Завдяки цьому аналізу сформовано базові вимоги до функціоналу та технічної реалізації платформи ArtUA, які відповідають як стандартам веброзробки, так і специфіці цільової аудиторії.

РОЗДІЛ 2
РОЗРОБКА ТА ПРОЄКТУВАННЯ ВЕБПЛАТФОРМИ “ArtUA”

2.1. Постановка задачі, призначення та вимоги до програмного засобу “ArtUA”

Програмний засіб “ArtUA” розробляється з метою створення вебплатформи, яка слугуватиме цифровим простором для українських митців, де вони зможуть представляти свої роботи, розвивати власний бренд, взаємодіяти з аудиторією та зберігати цифрову колекцію творчості. Основним призначенням системи є надання користувачам – як авторам, так і глядачам – можливості брати участь у культурному процесі.

Мета програмного забезпечення – забезпечити українським митцям зручний цифровий простір для демонстрації творчих робіт, формування особистого бренду, комунікації з аудиторією та участі в культурних ініціативах. Програмне забезпечення має сприяти збереженню, поширенню та популяризації українського мистецтва через функціональний вебінтерфейс, адаптований до потреб творчої спільноти.

Основна задача – завершення повного циклу розробки вебплатформи “ArtUA” шляхом удосконалення технічної архітектури, реалізації та інтеграції ключових функціональних компонентів, а саме: модуля взаємодії користувачів у реальному часі (чат), системи сповіщень, механізмів автентифікації та авторизації з використанням JWT, динамічного управління контентом, оптимізованого REST API та вдосконаленої логіки роботи з базами даних. Завершальним етапом є проведення комплексного тестування з метою перевірки стійкості, узгодженості та готовності системи до розгортання.

Функціональні вимоги включають автентифікацію та авторизацію користувачів, створення та редагування портфоліо, публікацію робіт з тегами та описами, виконання функцій фільтрації та контекстного пошуку, а також систему коментарів та сповіщень. Фреймворк також дозволяє масштабованість у збільшенні кількості користувачів та впровадженні нових модулів.

Інтерфейс не повинен відволікати від головної події – мистецтва, а навпаки, покращувати її, забезпечуючи середовище, яке сприяє концентрації на контенті.

Отже, “ArtUA” – це гнучка вебплатформа, яка вирішує актуальні проблеми митців та адаптується до еволюції культурного середовища, технологічного прогресу та соціальних запитів.

2.2. Вибір моделі розробки програмного засобу “ArtUA”

Розробка програмного забезпечення для платформи ArtUA проводилася з урахуванням структурної моделі життєвого циклу міжнародного стандарту ISO/IEC 12207:1995. Згідно зі стандартом, програмне забезпечення – це інтегрована система, яка включає не лише комп'ютерні програми, але й пов'язані з ними процедури, документацію та дані, що формують цілісний процес від розробки до виведення з експлуатації. Основою цього методу є ідея процесу як сукупності взаємопов'язаних дій, які перетворюють вхідну інформацію на вихідні результати [29].

У контексті розгортання платформи ArtUA розробка розпочалася з визнання існуючої вимоги – наявності цифрової платформи для продажу та підтримки українських художників. Це послужило основою для встановлення вимог до майбутнього програмного інструменту, як функціонального, так і нефункціонального характеру. Після цього було проведено ідентифікацію та опис вимог користувача, що дозволило створити основу для архітектурного та інтерфейсного проектування.

Архітектурний стиль включав реалізацію клієнт-серверної моделі, де серверний компонент був розроблений на C# та ASP.NET Core, а frontend – на React [30]. Для обробки та зберігання даних використовувався PostgreSQL у поєднанні з технологією ORM Entity Framework. Безпека забезпечувалася за допомогою механізму JWT-автентифікації, а зв'язок у режимі реального часу досягався за допомогою технології SignalR. REST API документувався за допомогою Swagger.

Протягом процесу впровадження платформа пройшла кілька циклів тестування та ітерацій. Спочатку було встановлено базову функціональність, а потім вона добудовувалася та переглядалася відповідно до результатів, отриманих в результаті зворотного зв'язку та тестування. Це забезпечувало простір для адаптивного реагування на зміну вимог та оптимізації інтерфейсу, який був адаптований до різних пристроїв та рівнів підготовки користувачів. Через поетапне впровадження, можливість покрокового впровадження функцій, періодичного тестування та зворотного зв'язку, було бажано обрати ітеративну модель розробки. Модель дозволила збалансувати гнучкість та контроль якості, а також динамічне вдосконалення програмного продукту. Такий підхід повністю відповідає філософії життєвого циклу програмного забезпечення згідно зі стандартом ISO/IEC 12207:1995 [29] та був ефективним у проекті ArtUA.

2.3. Загальний опис проєкту “ArtUA”

Платформа “ArtUA” розробляється як веборієнтований програмний засіб для підтримки діяльності українських митців шляхом створення інтегрованого цифрового середовища, що забезпечує можливості для презентації авторських робіт, взаємодії між користувачами та сприяння розвитку національного мистецтва в онлайн-просторі. У межах попереднього етапу дослідження, що був реалізований у курсовій роботі, проведено аналіз потреб цільової аудиторії, яка охоплює митців, колекціонерів і глядачів. На підставі результатів цього аналізу було сформульовано технічне завдання до програмного забезпечення, визначено ключові функціональні вимоги та розроблено базову структуру інтерфейсу користувача. Також було обґрунтовано вибір архітектурних рішень і реалізовано прототип основного функціоналу системи. Отримані результати підтвердили актуальність розробки та створили підґрунтя для подальшої реалізації повнофункціональної платформи в межах дипломної роботи.

Архітектура платформи побудована за клієнт-серверною моделлю. Фронтенд-частина реалізована з використанням бібліотеки React.js, що забезпечує динамічність, зручність у створенні компонентів та високу

продуктивність на стороні клієнта. Серверна частина написана з використанням середовища ASP.NET Core на мові програмування C# [31]. Такий вибір зумовлений потребою в масштабованості, безпеці та інтеграції з іншими сервісами. Система побудована на принципах багаторівневої архітектури, яка передбачає розділення на презентаційний рівень, логічний рівень і рівень доступу до даних. Комунікація між клієнтом і сервером відбувається через REST API []. Всі дані зберігаються в реляційній базі даних, що розміщена у хмарній інфраструктурі Realhost. Це забезпечує масштабовану та безперебійну роботу платформи.

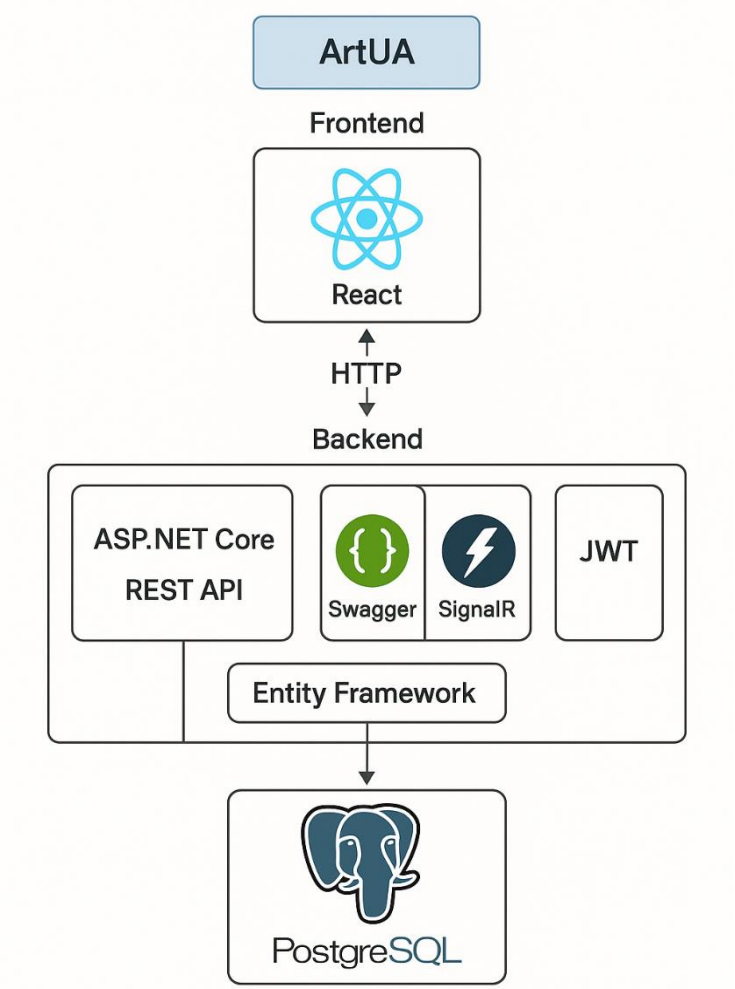

Рисунок 2.1 – Схема архітектури вебплатформи “ArtUA”

У реалізації захисту персональних даних у системі використано механізми хешування паролів та автентифікацію з використанням JWT (JSON Web Token). Зокрема, для хешування паролів застосовується UserManager з ASP.NET Core

Identity, який використовує алгоритм PBKDF2 із солями за замовчуванням. Під час оновлення пароля викликається метод:

```
272             var result = await _userManager.ChangePasswordAsync(
273                 user,
274                 request.CurrentPassword,
275                 request.NewPassword
276             );
```

Рисунок 2.2 – Оновлення пароля через ChangePasswordAsync у ASP.NET Core

Маємо реалізований функціонал – новий пароль автоматично хешується перед збереженням у базі даних, що відповідає принципам безпечного зберігання облікових даних [31].

Для автентифікації реалізовано обробку JWT-токена, який генерується під час реєстрації або входу користувача та передається клієнту у відповідь на запит. У системі передбачено два основні публічні методи контролера AuthController – Register та Login, які викликають відповідні методи сервісу автентифікації AuthService. У результаті клієнт отримує об’єкт з полем token, що містить сформований JWT [32].

```
[HttpPost("register")]
0 references
public async Task<IActionResult> Register([FromForm] UserRegisterModel model)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(new { message = "Invalid registration data" });
    }

    AuthResult result = await _authService.RegisterAsync(model, model.Avatar);

    if (!result.Success)
    {
        return BadRequest(new { message = result.ErrorMessage });
    }

    return Ok(new { token = result.Token });
}
```

Рисунок 2.3 – Формування JWT-токена після успішної реєстрації користувача

У подальшому токен додається до заголовків запитів у вигляді Authorization зі значенням Bearer <token>, що дозволяє контролерам системи

здійснювати авторизацію на основі вбудованого механізму ASP.NET Core.

Таким чином, використання JWT забезпечує безпечну та масштабовану автентифікацію користувачів відповідно до вимог безпеки [33, 34].

Також було реалізовано механізм автентифікації та авторизації використовуючи ASP.NET Identity. Клас користувача UserEntity розширює базовий клас IdentityUser і включає додаткові поля, що відповідають функціональним потребам проєкту. Нижче представлено приклад частини коду цієї моделі:

```
public class UserEntity : IdentityUser
{
    [Required]
    [StringLength(50)]
    9 references
    public string FirstName { get; set; } = string.Empty;
    [Required]
    [StringLength(50)]
    9 references
    public string LastName { get; set; } = string.Empty;
    [Url]
    7 references
    public string? Avatar { get; set; }
    [StringLength(100)]
    2 references
    public string? MyCountry { get; set; }
    // Відносини для месенджера
    0 references
    public List<ConversationParticipant> Conversations { get; set; } = new();
    0 references
    public List<Message> Messages { get; set; } = new();
    // Для продуктів (Products)
    1 reference
    public List<UserProductLike> LikedProducts { get; set; } = new();
    1 reference
    public List<UserProductView> ViewedProducts { get; set; } = new();
    1 reference
    public List<UserSubscription> Subscriptions { get; set; } = new();
    1 reference
    public List<UserSubscription> Followers { get; set; } = new();
}
```

Рисунок 2.4 – Фрагмент моделі користувача UserEntity у системі “ArtUA”

Ця модель є центральною у побудові взаємодії користувача з іншими елементами платформи. Вона пов’язана з сутностями повідомлень, діалогів, переглянутих та вподобаних робіт тощо.

Структура бази даних сформована у вигляді логічно взаємопов’язаних сутностей: користувачів, продуктів, зображень, повідомлень, діалогів, підписок,

вподобань і переглядів. Всі таблиці пов’язані зовнішніми ключами, що забезпечує референційну цілісність даних. Проєктування структури бази даних здійснювалося відповідно до принципів нормалізації, що дозволило уникнути надлишковості інформації та забезпечити ефективне виконання SQL-запитів. Усі сутності системи візуалізовані у вигляді ERD-діаграми, яка слугує основою для подальшого розширення функціоналу [35].

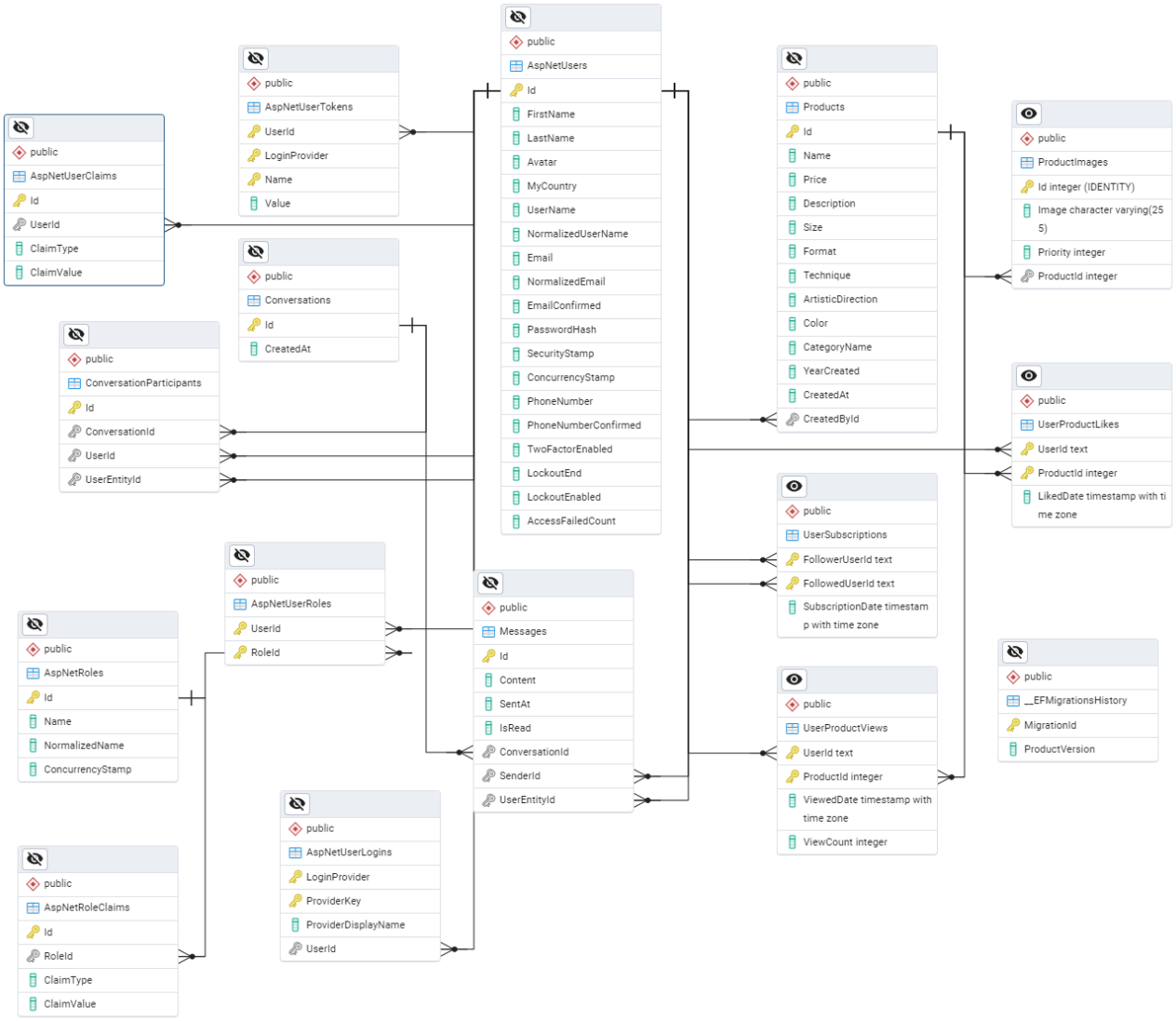

Рисунок 2.5 – ERD-діаграма бази даних вебплатформи “ArtUA”

Інтерфейс платформи розроблений із дотриманням принципів UX/UI-дизайну. При розробці враховувалась логічна послідовність дій користувача, інтуїтивна зрозумілість елементів, візуальна привабливість та доступність [36]. Навігація між сторінками відбувається за допомогою верхнього меню, що

включає основні розділи сайту. Користувач має змогу створити профіль, переглядати головну сторінку з добірками, шукати роботи в каталозі, взаємодіяти з іншими митцями, додавати нові твори, спілкуватися через повідомлення та брати участь у виставках. Навігаційна структура спроєктована таким чином, щоб користувачі легко орієнтувались у середовищі платформи.

Архітектура проєкту додатково описується за допомогою UML-діаграми у попередньому дослідженні (курсовій роботі) . Діаграма варіантів використання ілюструє типову взаємодію користувача з системою, зокрема перегляд профілю, додавання роботи, участь у виставках та виконання інших дій.

Отже, платформа “ArtUA” являє собою вебсервіс, побудований на основі гнучкої архітектури, оптимізованої структури бази даних, продуманого інтерфейсу та адаптивної логіки рекомендацій. Усі компоненти системи взаємодіють між собою в межах єдиного цифрового середовища, що орієнтоване на розвиток українського мистецтва в умовах цифрової трансформації.

2.4. Обґрунтування вибору інструментальних засобів розробки “ArtUA”

Розробка вебзастосунку “ArtUA” вимагала визначення нового, масштабованого та ефективного технологічного стеку, який би забезпечував стабільну взаємодію клієнт-сервер у системі, дозволяв доступ для багатьох користувачів у режимі реального часу та пропонував високий рівень безпеки. З огляду на це, було обрано набір інструментів та технологій, які допомогли в реалізації складних вебзастосунків.

Під час створення клієнтської частини платформи було використано React – бібліотеку, яка особливо підходить для створення інтерфейсу на основі компонентної моделі та дозволяє створювати адаптивні та високопродуктивні користувацькі інтерфейси. Використання React певною мірою було зумовлене тим, що він ідеально інтегрується з SignalR – бібліотекою, яка спрощує використання WebSocket-з'єднань та покращує зв'язок між сервером та клієнтом у режимі реального часу. Такий підхід гарантує динамічність контенту, підвищує

актуальність змін та робить систему більш інтерактивною загалом.

Окрім технічної цінності, вибір React також підтверджується його постійною популярністю серед розробників. Як показано на рисунку нижче, дані Stack Overflow (Tag Trends) показують, що React.js став найчастіше згадуваним JavaScript-фреймворком останнім часом серед Angular та Vue.js. Це свідчить про широку підтримку, активний розвиток екосистеми та доступ до величезної кількості ресурсів для розробників.

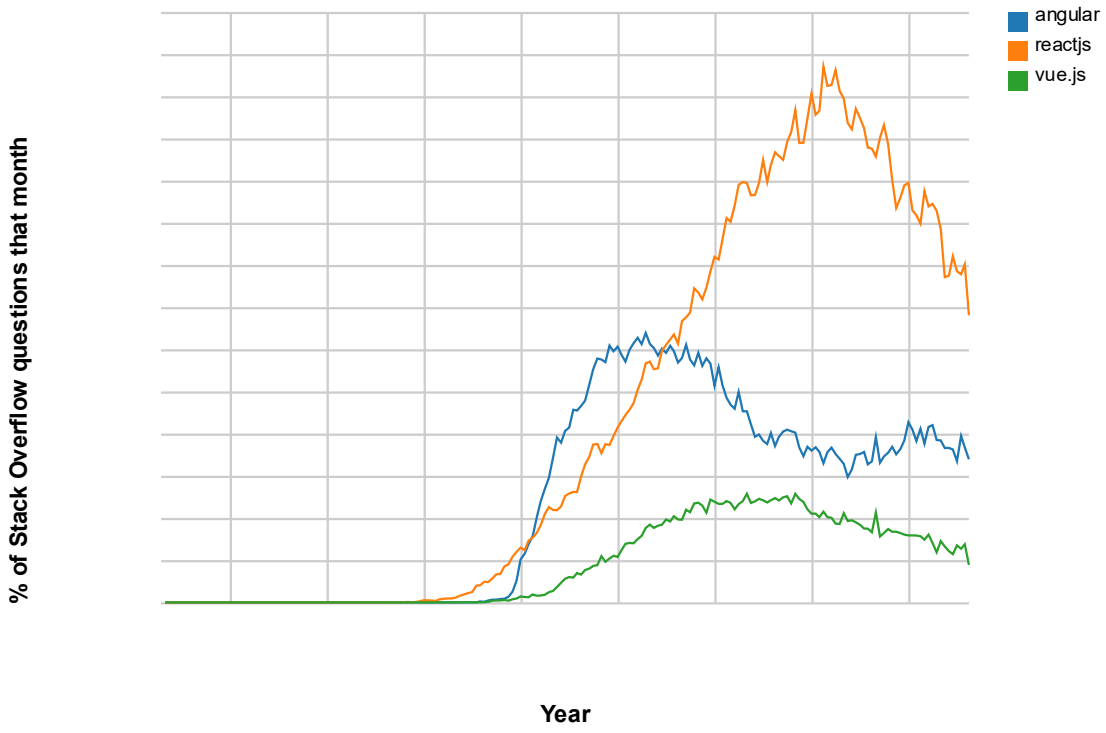

Рисунок 2.6 – Динаміка популярності React, Angular і Vue (Stack Overflow, 2008–2024)

Серверний компонент реалізовано за допомогою вебфреймворку ASP.NET Core, який є високопродуктивним та масштабованим. Обробка HTTP-запитів та розробка REST API була досягнута завдяки застосуванню архітектурного стилю REST у поєднанні з інструментом Swagger, що дозволяє документувати API-інтерфейс, тестувати та інтегрувати серверні та клієнтські компоненти програми.

Мову програмування C# було обрано завдяки її повній сумісності з .NET, строгій типізації, простому асинхронному програмуванню та складному

середовищу Visual Studio. Зв'язок з реляційною системою баз даних PostgreSQL забезпечувався за допомогою Entity Framework Core – ORM-системи, яка підтримувала просту реалізацію міграції.

PostgreSQL було обрано для зберігання даних, як надійну базу даних з відкритим кодом, яка вміло обробляє складні запити, зберігаючи при цьому масштабованість та цілісність даних. Порівняно з SQL Server, PostgreSQL продемонстрував більшу кросплатформну гнучкість для використання у виробничому середовищі.

Протокол JSON Web Token було використано як метод автентифікації та авторизації, що забезпечувало ефективний та безпечний обмін обліковими даними між клієнтом і сервером, і запобігало зберіганню стану на сервері. Ця система забезпечувала кращу масштабованість системи та ефективне управління сеансами користувачів у розподіленій системі.

Використання середовища розробки пояснюється перевагами, які пропонує Visual Studio для розробки серверної частини на ASP.NET Core, включаючи підтримку розширень, ефективні інструменти налагодження логіки та інтеграцію з базами даних. У той час як розробка клієнтської частини на React здійснювалася у Visual Studio Code, що є легким, зручним та швидким редактором із багатою екосистемою розширень для JavaScript і React. Контроль версій забезпечується використанням Git з GitHub, що сприяє ефективній співпраці та відстеженню змін.

2.5. Етапи реалізації програмного засобу “ArtUA”

У процесі розробки вебплатформи “ArtUA” було реалізовано низку функціональних модулів, які забезпечують повноцінну взаємодію користувачів з системою. Програмний продукт реалізований на основі клієнт-серверної архітектури, де фронтенд-частина створена за допомогою бібліотеки React з TypeScript, а серверна частина – на платформі ASP.NET Core з мовою програмування C#. Такий підхід дозволив досягти високої гнучкості, масштабованості та розділення відповідальностей між частинами застосунку.

Кожен модуль платформи має чітко визначене призначення і взаємодіє з іншими через стандартизовані API-запити, що сприяє зручному супроводу проєкту та можливості його подальшого розвитку.

До основних функціональних підсистем платформи належать:

- система аутентифікації та авторизації користувачів;
- модуль обробки зображень;
- управління мистецькими роботами;
- обробка підписок;
- система чатів та повідомлень у реальному часі;
- інтеграція користувацьких профілів і взаємодія між учасниками.

Одним із ключових етапів реалізації програмного засобу “ArtUA” стала розробка механізму аутентифікації користувачів. За її реалізацію відповідає сервіс AuthService, який взаємодіє з UserManager, SignInManager, ImageService та системою конфігурації для створення нового облікового запису, входу в систему та формування JWT-токенів.

У методі RegisterAsync відбувається створення нового користувача, зчитування даних із форми реєстрації та, за потреби, збереження аватару через ImageService. Якщо реєстрація успішна, користувачу присвоюється роль та генерується JWT-токен.

RegisterAsync отримує модель реєстрації користувача, яка містить деяку просту особисту інформацію (електронна пошта, ім'я, прізвище, пароль) та файл аватара типу IFormFile. Процес починається зі створення нового об'єкта користувача, який зберігається в базі даних за допомогою UserManager.CreateAsync. Після успішного створення облікового запису виконується додаткова перевірка наявності аватара. Таким чином, метод забезпечує повний цикл реєстрації, що включає генерацію даних, перевірку та зберігання, обробку медіафайлів, призначення ролей і генерацію токенів доступу.

```
// Метод для реєстрації нового користувача з можливістю завантажити аватар
1 reference
public async Task<AuthResult> RegisterAsync(UserRegisterModel model, IFormFile avatarFile)
{
    // Створення нового об'єкта користувача на основі отриманої моделі
    var user = new UserEntity...;
    // Створення користувача в базі даних із валідацією пароля
    var result = await _userManager.CreateAsync(user, model.Password);
    // Якщо створення не вдалося — повертаємо помилку з описом
    if (!result.Succeeded)
    {
        return new AuthResult...;
    }
    // Якщо передано аватар, зберігаємо його через відповідний сервіс
    if (avatarFile != null && avatarFile.Length > 0)
    {
        try
        {
            // Збереження файлу з аватаром та оновлення даних користувача
            string savedFileName = await _imageService.SaveUserAvatarAsync(avatarFile, user.Id);
            user.Avatar = savedFileName;
            await _userManager.UpdateAsync(user);
        }
        catch (Exception ex)
        {
            // У разі помилки з аватаром — повідомлення про помилку, без відміни реєстрації
            return new AuthResult...;
        }
    }
    // Призначення ролі новому користувачу
    await _userManager.AddToRoleAsync(user, model.Role);
    // Генерація JWT-токена та повернення результату
    return new AuthResult
    {
        Success = true,
        Token = await GenerateJwtToken(user)
```

Рисунок 2.7 – Фрагмент коду методу RegisterAsync сервісу AuthService, що реалізує реєстрацію користувача та генерацію JWT-токена

Контролер AuthController відповідає за обробку HTTP-запитів, що стосуються аутентифікації та авторизації користувачів на платформі “ArtUA”.

```
[AllowAnonymous]
[HttpPost("register")]
0 references
public async Task<IActionResult> Register([FromForm] UserRegisterModel model)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(new { message = "Invalid registration data" });
    }

    AuthResult result = await _authService.RegisterAsync(model, model.Avatar);

    if (!result.Success)
    {
        return BadRequest(new { message = result.ErrorMessage });
    }

    return Ok(new { token = result.Token });
}
```

Рисунок 2.8 – Фрагмент коду контролера AuthController для реєстрації користувача

Register – метод для реєстрації нового користувача. Він обробляє запит на створення облікового запису з можливістю завантаження аватару користувача. Цей метод перевіряє правильність введених даних за допомогою моделі UserRegisterModel. Якщо дані валідні, викликається метод RegisterAsync з сервісу AuthService для створення користувача та збереження аватару, якщо він був наданий. У разі успішної реєстрації генерується JWT-токен для нового користувача.

```
[AllowAnonymous]
[HttpPost("login")]
0 references
public async Task<IActionResult> Login([FromBody] LoginModel model)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(new { message = "Invalid login data" });
    }

    AuthResult result = await _authService.LoginAsync(model);

    if (!result.Success)
    {
        return Unauthorized(new { message = result.ErrorMessage });
    }

    return Ok(new { token = result.Token });
}
```

Рисунок 2.9 – Фрагмент коду контролера AuthController для логіна користувача.

Login – метод для аутентифікації користувача. Цей метод перевіряє правильність даних, що надходять від користувача (модель LoginModel). Якщо введені дані правильні, сервіс AuthService викликає метод LoginAsync, який здійснює перевірку користувача та генерує JWT-токен для подальшого доступу до захищених ресурсів.

Контролер AuthController працює як посередник між клієнтськими запитами і логікою, що виконується на сервері через сервіс AuthService. Кожен з методів контролера – Register і Login, викликає відповідні асинхронні методи сервісу AuthService, щоб виконати бізнес-логіку, пов'язану з аутентифікацією користувачів.

У процесі розробки було запроваджено функціональність приватного чату для спілкування користувачів. Основною метою цієї функції є сприяння миттєвому обміну повідомленнями в режимі реального часу, тим самим підвищуючи ефективність комунікації на платформі. Для реалізації цієї функції було використано технологію SignalR, яка спирається на протокол WebSocket, а також центральний концентратор обробки з'єднань (ChatHub) та REST API-контролер для обробки повідомлень (ChatController).

Комунікація в реальному часі реалізується за допомогою класу ChatHub, який є похідним від Hub (простір імен Microsoft.AspNetCore.SignalR). Саме в цьому класі обробляються події підключення, надсилання та читання повідомлень. З іншого боку, ChatController відповідає за взаємодію з базою даних, а також надає API-інтерфейс для створення чатів, отримання історії листування та списку бесід. Файл ChatHub.cs відповідає за двосторонню комунікацію клієнта і сервера.

```
0 references
public override async Task OnConnectedAsync()
{
    try
    {
        var httpContext = Context.GetHttpContext();
        var token = httpContext?.Request.Query["access_token"].ToString();

        if (string.IsNullOrEmpty(token))
        {
            Context.Abort();
            return;
        }

        var principal = ValidateJwtToken(token);
        Context.Items["User"] = principal;

        await base.OnConnectedAsync();
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Connection error: {ex}");
        Context.Abort();
    }
}
```

Рисунок 2.10 – Фрагмент коду реалізації функції з'єднання

Після з'єднання з клієнтом сервер витягує access_token із запиту, перевіряє автентичність користувача та додає його до відповідних SignalR-груп на основі ідентифікаторів бесід. Це дозволяє доставляти повідомлення лише тим

користувачам, які беруть участь у конкретній бесіді.

Загальна логіка передбачає:

– аутентифікацію користувачів через JWT;
– зберігання повідомлень у базі даних через Entity Framework;
– відправку повідомлень через WebSocket у групу SignalR;
– підрахунок непрочитаних повідомлень.

```
0 references
public async Task SendMessage(int conversationId, string content)
{
    try
    {
        var user = GetAuthenticatedUser();
        var userId = user.FindFirstValue(ClaimTypes.NameIdentifier);

        var message = new Message
        {
            Content = content,
            SenderId = userId,
            ConversationId = conversationId,
            SentAt = DateTime.UtcNow,
            IsRead = false
        };
        await _context.Messages.AddAsync(message);
        await _context.SaveChangesAsync();
        // Відправка повідомлення усім учасникам бесіди
        await Clients.Group(conversationId.ToString())
            .SendAsync("ReceiveMessage", new
            {
                message.Id,
                message.Content,
                message.SentAt,
                SenderId = userId,
                ConversationId = conversationId,
                message.IsRead
            });
        // Знаходимо всіх учасників бесіди, крім відправника
        var participants = await _context.ConversationParticipants
            .Where(cp => cp.ConversationId == conversationId && cp.UserId != userId)
            .Select(cp => cp.UserId)
            .ToListAsync();
```

Рисунок 2.11 – Фрагмент коду реалізації функції відправки повідомлень

На рисунку 2.11 продемонстровано як здійснюється збереження повідомлення у базі даних та передача об'єкта в SignalR-групу. Метод Clients.Group(...) гарантує, що повідомлення побачать лише учасники відповідної розмови.

Контролер ChatController надає REST API, який працює з базою даних,

дозволяючи створювати бесіди, отримувати історію повідомлень, та забезпечує перевірку токена. Метод перевіряє, чи існує вже чат між двома користувачами. Якщо ні – створює новий запис у таблиці Conversations.

Таким чином, кожна дія користувача у чаті проходить автентифікацію, що унеможливлює несанкціонований доступ до сторонніх діалогів. Хоч ці два компоненти не викликають один одного, вони взаємодіють через спільний рівень даних (базу). ChatController – для HTTP-запитів, ChatHub – для миттєвого обміну.

```
public async Task<IActionResult> CreateConversation([FromBody] CreateConversationRequest request)
{
    try
    {
        var userId = GetUserIdFromToken(request.Token);
        // Перевірка існування учасника
        var participantExists = await _context.Users.AnyAsync(u => u.Id == request.ParticipantId);
        if (!participantExists)
            return BadRequest(new { Message = "Користувача не знайдено" });
        // Пошук існуючого чату
        var existingConversation = await _context.Conversations
            .Where(c => c.Participants.Any(p => p.UserId == userId))
            .Where(c => c.Participants.Any(p => p.UserId == request.ParticipantId))
            .Select(c => new { c.Id, ParticipantsCount = c.Participants.Count })
            .FirstOrDefaultAsync(c => c.ParticipantsCount == 2);

        if (existingConversation != null)
            return Ok(new { conversationId = existingConversation.Id, Message = "Чат вже існує" });
        // Створення нового чату
        var conversation = new Conversation
        {
            Participants = new List<ConversationParticipant>
    {
        new() { UserId = userId },
        new() { UserId = request.ParticipantId }
    },
            CreatedAt = DateTime.UtcNow
        };
        _context.Conversations.Add(conversation);
        await _context.SaveChangesAsync();

        return Ok(new { conversation.Id });
    }
```

Рисунок 2.12 – Фрагмент коду з реалізованою функцією створення чату

Реалізація методу дає змогу централізовано керувати логікою створення діалогів, що спрощує підтримку та масштабування. У разі розширення функціоналу, наприклад додавання групових чатів, цю структуру можна легко модифікувати. Це демонструє важливість чітко визначених API-інтерфейсів та їхньої відповідності загальній архітектурі проєкту.

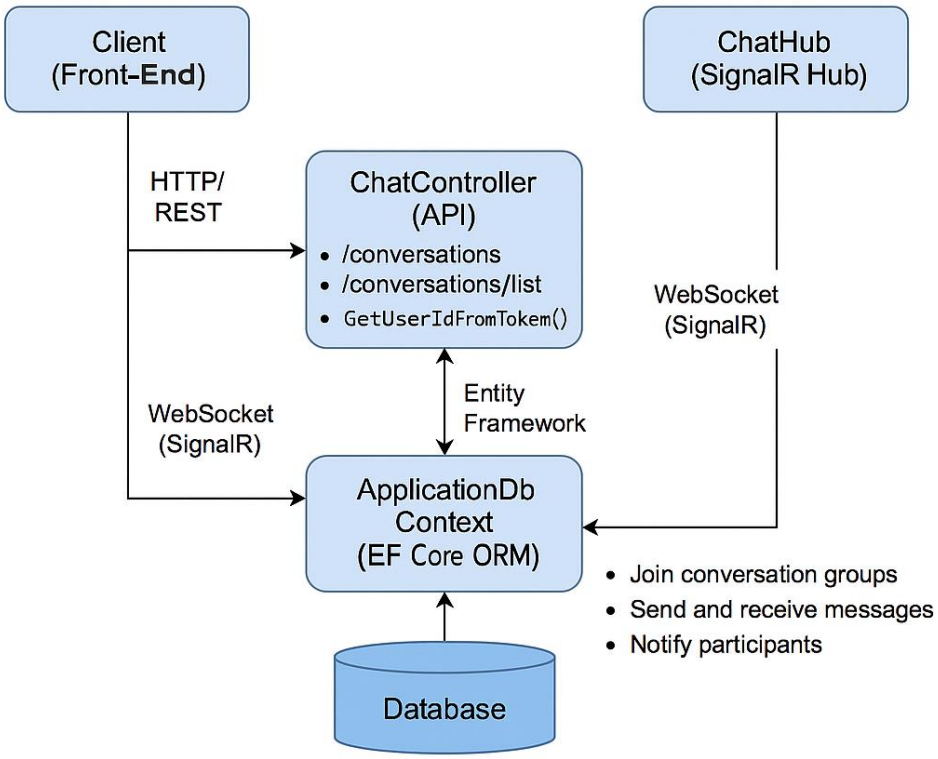

Рисунок 2.13 – Взаємодія компонентів системи чату

Схема представляє логіку взаємодії основних компонентів системи обміну повідомленнями у вебдодатку “ArtUA”. Обмін даними між клієнтською стороною (frontend) та серверною (backend) здійснюється за допомогою двох механізмів – REST API-запитів і WebSocket-з'єднання, реалізованого за допомогою бібліотеки SignalR.

На першому етапі автентифікований користувач, передаючи JWT у заголовку запиту, ініціює створення нового чату або отримання списку діалогів через ChatController. Контролер перевіряє токен, звертається до ApplicationDbContext, формує або витягує відповідні записи з бази даних та повертає результат у форматі JSON.

Після надсилання повідомлення клас ChatHub встановлює WebSocket-з'єднання через SignalR. Повідомлення миттєво транслюється всім учасникам відповідного чату (групи), а також зберігається у базі даних методами, реалізованими у ChatController.

Отже, REST API використовується для операцій зі структурованими

даними (створення чату, отримання історії повідомлень), а SignalR – для комунікації в режимі реального часу (отримання та надсилання повідомлень, сповіщення про нові події в чаті).

2.6. Організація тестування та налагодження програмного засобу “ArtUA”

Відповідно до бізнес-вимог, первинне ручне тестування кожного функціонального компонента вебсистеми було проведено після завершення етапів розробки API. У процесі розробки перевірено коректну роботу API, створених на ASP.NET Core, з акцентом на бізнес-логіку, валідацію вхідних даних, обробку помилок та взаємодію між клієнтом і сервером.

Для тестування REST API було використано вбудований інструмент Swagger UI, який автоматично генерується з опису контролерів та моделей у проєкті. Це дозволило інтерактивно взаємодіяти з ендпоінтами безпосередньо через браузер, здійснюючи запити із зазначенням параметрів, заголовків, токенів авторизації тощо.

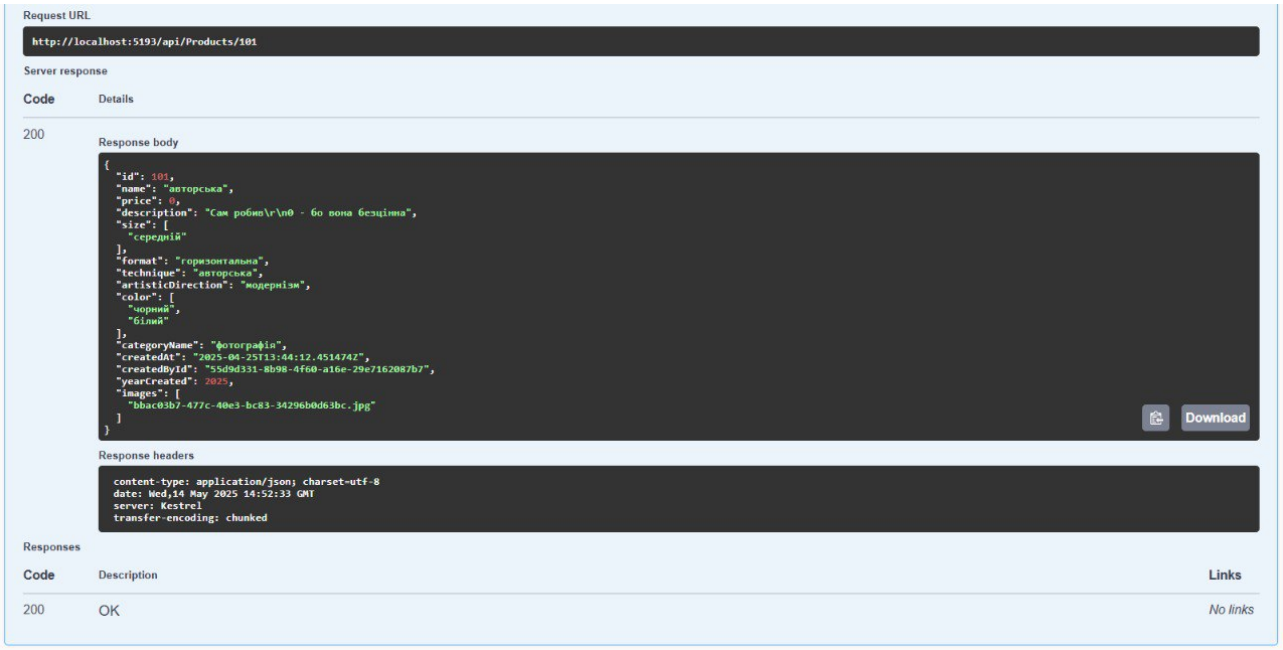

Рисунок 2.14 – Тестування ендпоінту отримання товару в Swagger UI

На рисунку 2.14 представлено результат тестування ендпоінту GET

/api/Products/{id}, який відповідає за отримання детальної інформації про конкретний товар. У Swagger UI вручну введено значення ідентифікатора, після чого було надіслано запит із валідним токеном у заголовку Authorization: Bearer. У відповідь сервер повернув JSON-об'єкт із полями id, name, description, price, imageUrl, що підтверджує правильну інтеграцію з базою даних та відповідність логіки контролера специфікації.

Окремо було перевірено поведінку сервера у випадках, коли товар із зазначеним id не існує. У таких ситуаціях API повертає статус 404 Not Found із поясненням у тілі відповіді. Також перевірялися випадки відсутності токена, що призводить до статусу 401 Unauthorized, що демонструє коректну роботу механізму контролю доступу.

Під час ручного тестування інтерфейсу користувача було перевірено функціональність основних елементів взаємодії на платформі ArtUA. Зокрема, увагу зосереджено на роботі кнопок, форм редагування, повідомлень про помилки, поведінці фільтрів та оновленні вмісту сторінок без перезавантаження. Виявлено кілька недоліків, зокрема некоректне оновлення даних після редагування, відсутність повідомлень про успішні дії та помилки у фільтрації товарів. Після доопрацювання було реалізовано автоматичне оновлення вмісту, відображення повідомлень про статус операцій та забезпечено відповідність між станом клієнтської частини і серверними змінами. Це дозволило покращити динамічність роботи інтерфейсу та його зручність для користувача.

2.7. Рекомендації по використанню та впровадженню програмного засобу “ArtUA”

Програмне забезпечення “ArtUA” може перетворитися на комплексну платформу, яка дозволяє художникам спілкуватися зі своєю цільовою аудиторією, презентувати їй свої роботи та сприяти можливостям для обговорення між художниками. Архітектура інструменту базується на вебтехнологіях, що забезпечує масштабованість, швидкість реагування інтерфейсу та можливості інтеграції з іншими сервісами в майбутньому.

Для оптимального функціонування системи рекомендується мати серверне середовище, яке має такі якості:

- операційна система Windows Server 2019 або Linux (наприклад, Ubuntu 22.04 LTS);
- процесор щонайменше 4 ядра (Intel i5 і вище або AMD Ryzen 5 і вище);
- ОЗП від 8 ГБ;
- накопичувач SSD-диск на 100 ГБ (з можливістю розширення ємності);
- серверна платформа підтримує ASP.NET Core (версія 7.0 або пізніша), наявність встановленого середовища .NET Runtime;
- СУБД PostgreSQL 14 або пізніша.

На стороні клієнта програма не потребує встановлення жодного програмного забезпечення, оскільки є вебзастосунком. Доступ до неї можна отримати за допомогою будь-якого браузера з підтримкою JavaScript, такого як Google Chrome, Firefox або Microsoft Edge. В умовах експлуатації важливо відзначити стабільне мережеве з'єднання, наявність сертифікатів безпеки для зашифрованої передачі даних та налаштування систем резервного копіювання баз даних. Рекомендується запускати програму в закладах культури, художніх музеях, молодіжних освітніх програмах, державних установах або інкубаторах творчих проектів. Подальше зростання, залежно від попиту та потужностей, може бути досягнуто шляхом впровадження на хмарних платформах (наприклад, Azure або AWS), що забезпечить системі додаткову гнучкість та відмовостійкість.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було завершено повний цикл розробки вебплатформи “ArtUA”, яка є результатом подальшого вдосконалення і реалізації ідей, закладених у курсовому проєкті. Основний акцент було зроблено на програмну частину, динаміку взаємодії, зручність використання, захищеність даних, а також ефективну обробку контенту та масштабованість рішення. Відповідно до поставлених завдань, вдалося суттєво удосконалити технічну архітектуру системи з урахуванням принципів модульності, безпеки, продуктивності та розширюваності, що створює передумови для подальшого розвитку функціоналу.

Було реалізовано і впроваджено низку ключових функціональних компонентів, зокрема модуль інтерактивної взаємодії користувачів у реальному часі з використанням SignalR, систему сповіщень, а також механізми автентифікації та авторизації на основі JWT. Це дозволило забезпечити захищену та гнучку модель доступу до функцій платформи з урахуванням ролей користувачів. Значну увагу було приділено оптимізації REST API для покращення продуктивності обміну даними між клієнтською і серверною частинами, що забезпечило стабільну й швидку роботу платформи в умовах навантаження.

Окремо було вдосконалено логіку взаємодії з базами даних – реалізовано динамічну фільтрацію контенту за категоріями, підвищено ефективність виконання запитів та забезпечено узгодженість даних. Також впроваджено механізми динамічного управління контентом, які надають можливість користувачам (митцям) додавати, редагувати та видаляти власні матеріали, а відвідувачам – здійснювати пошук і перегляд відповідно до заданих фільтрів.

Результати ручного та автоматизованого тестування підтвердили працездатність реалізованої системи та відповідність основним технічним вимогам. У процесі перевірки було виявлено та усунуто низку недоліків, пов’язаних із візуалізацією змін у стані інтерфейсу, коректністю збереження даних та обробкою фільтрів. Після усунення вказаних проблем було досягнуто

належного рівня динамічності взаємодії користувачів із системою.

Водночас у процесі реалізації було виявлено певні обмеження, які доцільно врахувати під час подальшого розвитку платформи. Зокрема, відсутність засобів розмітки мультимедійного контенту, обмежені інструменти модерації користувацьких матеріалів та недостатній рівень аналітики поведінки користувачів становлять виклики для ефективного адміністрування та масштабування сервісу. У подальшій розробці доцільним є впровадження засобів автоматичної модерації, розширення адміністративного функціоналу, а також інтеграція аналітичних інструментів для моніторингу активності користувачів.

Також перспективними напрямами удосконалення є впровадження багатомовної підтримки інтерфейсу, інтеграція платіжних систем з метою забезпечення монетизації творчості та подальше покращення користувацького досвіду відповідно до принципів доступності. Таким чином, реалізована вебплатформа “ArtUA” не лише відповідає сучасним вимогам до програмних продуктів у галузі цифрової культури, а й створює підґрунтя для сталого розвитку онлайн-середовища для українських митців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rab Á. Digital culture – Digitalised culture and culture created on a digital platform. Budapest : Leonadro da Vinci, 2007. 24 c. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=f5f866fb13a283617998ccac0b128162568c1a31 (дата звернення: 11.02.2025).

2. Роль медіа в поширенні мистецьких цінностей. URL: : https://lms.e-school.net.ua/meta_user/download/irex/Mystetstvo_9-klass/II_semestr/12/Prezentatsiia_do_vpravy_Rol_media_u_poshyrenni_mystetskykh_tsinnostei.pdf (дата звернення: 11.02.2025).

3. Christodoulou S. P., Styliaras G. D. Digital Art 2.0: Art meets Web 2.0 trend. Greece : DIMEA, 2008. URL: https://doi.org/10.1504/IJWBC.2013.057218 (дата звернення: 12.02.2025).

4. Овчарук О. В. Підходи до розвитку цифрової компетентності людини та цифрового громадянства в європейських країнах. Information technologies and learning tools. 2020. Т. 76, № 2. С. 1–13. URL: https://doi.org/10.33407/itlt.v76i2.3526 (дата звернення: 13.02.2025).

5. Козік В. Микитенко В. Белянська Т. ВЗАЄМОДІЯ МИТЦІВ ІЗ ТЕХНОЛОГІЄЮ ДОПОВНЕНОЇ РЕАЛЬНОСТІ: ПРАКТИЧНИЙ ДОСВІД У СФЕРІ ГРАФІЧНОГО ДИЗАЙНУ. Актуальні питання гуманітарних наук. 2024. Т. 2, № 72. С. 2–3. URL: https://doi.org/10.24919/2308-4863/72-2-12.

6. An introduction to web applications architecture. *Open Learning*. URL: https://www.open.edu/openlearn/science-maths-technology/an-introduction-web-applications-architecture/content-section-1.1 (дата звернення: 12.02.2025).

7. RESTful APIs: Principles and Best Practices - API7.ai. *RESTful APIs: Principles and Best Practices - API7.ai*. URL: https://api7.ai/learning-center/api-101/restful-api-best-practices (дата звернення: 12.02.2025).

8. RESTful Web Services. O'Reilly Media, Incorporated, 2007.

9. ASP.NET documentation. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/aspnet/core/?view=aspnetcore-

9.0 (дата звернення: 12.02.2025).

10. Overview of Entity Framework Core - EF Core. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/ef/core/ (дата звернення: 13.02.2025).

11. Newman S. Building Microservices. O'Reilly Media, Incorporated, 2015.

12. Real-time ASP.NET with SignalR | .NET. *Microsoft*. URL: https://dotnet.microsoft.com/en-us/apps/aspnet/signalr (дата звернення: 13.02.2025).

9. Ethereum Foundation Report 2024 Edition: 2023 Total Expenditure $134.9 Million. URL: https://www.binance.com/en-TR/square/post/16050532969025 (дата звернення: 14.02.2025).

10. Welcome to Decentraland. URL: https://decentraland.org (дата звернення: 14.02.2025).

11. Nevil S. What Is Proof of Work (PoW) in Blockchain?. Investopedia. URL: https://www.investopedia.com/terms/p/proof-work.asp (дата звернення: 15.02.2025).

12. Леонід Тарасенко. NFT – НОВІТНІЙ ЦИФРОВИЙ ОБ'ЄКТ АВТОРСЬКОГО ПРАВА ЧИ ФОРМА ВИРАЖЕННЯ ТВОРУ. Теорія і практика інтелектуальної власності. 2022.

13. WebSocket - Web APIs | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket (дата звернення: 15.02.2025).

14. Overview of ASP.NET Core SignalR. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-9.0 (дата звернення: 15.02.2025).

15. Instagram. Instagram. URL: https://www.instagram.com (дата звернення: 19.02.2025).

16. TikTok – Make Your Day. TikTok – Make Your Day. URL: https://www.tiktok.com/uk-UA/ (дата звернення: 19.02.2025).

17. Art Basel. The Art Basel and UBS Global Art Market Report. Art Basel.

URL: https://www.artbasel.com/stories/the-art-basel-and-ubs-global-art-market-report-2022 (дата звернення: 19.02.2025).

18. Online sales of the global art market 2023 | Statista. Statista. URL: https://www.statista.com/statistics/886776/online-art-and-antiques-market-total-global-sales/ (дата звернення: 19.02.2025).

19. Artsy – Discover and Buy Fine Art. Artsy. URL: https://www.artsy.net (дата звернення: 19.02.2025).

20. Facebook – URL: https://www.facebook.com/groups/discover/ (дата звернення: 19.02.2025).

21. Akamai. URL: https://www.akamai.com (дата звернення: 19.02.2025).

22. Optimize JavaScript execution | Articles | web.dev. web.dev. URL: https://web.dev/articles/optimize-javascript-execution (дата звернення: 19.02.2025).

23. Nielsen J. 10 Usability Heuristics for User Interface Design. Nielsen Norman Group. URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (дата звернення: 20.02.2025).

24. Jha A. K., Verma N. K. Social Media Platforms and User Engagement: A Multi-Platform Study on One-way Firm Sustainability Communication. Information Systems Frontiers. 2023. URL: https://doi.org/10.1007/s10796-023-10376-8 (дата звернення: 20.02.2025).

25. Shostack A. Threat Modeling: Designing for Security. Wiley & Sons, Incorporated, John, 2014. 624 с.

26. What is Zero Trust Data Protection? | Object First. *Best Storage for Veeam | Object First*. URL: https://objectfirst.com/guides/data-security/what-is-zero-trust-data-protection/?utm_source=chatgpt.com (дата звернення: 01.04.2025).

27. ISTR 2019: Internet of Things Cyber Attacks Grow More Diverse. Symantec Enterprise Blogs. URL: https://www.security.com/expert-perspectives/istr-2019-internet-things-cyber-attacks-grow-more-diverse.

28. Contributors to Wikimedia projects. List of data breaches - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/List_of_data_breaches (дата звернення:

16.03.2025).

29. ISO/IEC 12207:1995. Information technology – Software life cycle processes. Geneva: International Organization for Standardization, 1995. 112 p. URL: https://cdn.standards.iteh.ai/samples/21208/b2ffd902bf8446b4b68e19a5c05c6b87/ISO-IEC-12207-1995.pdf

30. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley Professional, 2002. 560 c.

31. Introduction to Identity on ASP.NET Core. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-9.0&tabs=visual-studio (дата звернення: 03.04.2025).

32. Configure JWT bearer authentication in ASP.NET Core. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/configure-jwt-bearer-authentication?view=aspnetcore-9.0 (дата звернення: 03.04.2025).

33. Microsoft Docs. Authentication and Authorization in ASP.NET Core. URL: https://learn.microsoft.com/en-us/aspnet/core/security/authentication (дата звернення: 03.04.2025).

34. Jones M., Bradley J., and Sakimura N. "RFC 7519: JSON Web Token (JWT)". IETF, 2015. URL: https://www.rfc-editor.org/rfc/rfc7519.html (дата звернення: 03.04.2025).

35. Fundamentals of database systems / ред. N. Sham. 5-те вид. Upper Saddle River, NJ : Pearson, 2008. 1143 с.

36. Nielsen, J. (1995). 10 Usability Heuristics for User Interface Design. URL: https://www.nngroup.com/articles/ten-usability-heuristics/

ДОДАТОК А

Технічне завдання

Вступ

Програмний продукт “ArtUA” – це вебпортал для електронної підтримки українських митців. Вебсайт надає корисні інструменти для публікації творчих робіт, комунікації з аудиторією та просування сучасного українського мистецтва.

Сайт застосовується у сфері культури та мистецтва і може використовуватися митцями, галереями, громадськими та культурними установами. “ArtUA” відповідає вимогам до вебсервісів: безпека, гнучкість, масштабованість та підтримка функціонування в режимі реального часу.

Підстави для розробки

Основою для розробки вебплатформи “ArtUA” став зростаючий суспільний запит на підтримку української культури в умовах діджиталізації та військової агресії, які загрожують культурній спадщині, свободі творчості та інституційній стабільності митців. Внаслідок витіснення митців, втрати традиційних засобів комунікації з аудиторією та скорочення фінансування культури виникла потреба у створенні інструменту, який би надавав можливість митцям публікувати свої роботи, спілкуватися один з одним та з аудиторією, а також просувати національне мистецтво на світовій цифровій арені.

Проєкт “ArtUA” також розглядає тенденції цифрової трансформації культурної індустрії України, включаючи створення існуючих онлайн-рішень з метою просування креативної економіки, діджиталізації креативної діяльності та позиціонування митців у новому інформаційному просторі.

Робота виконується в рамках кваліфікаційної бакалаврської роботи за освітньою програмою “Комп’ютерні науки”.

Призначення розробки

Створення програмного продукту “ArtUA” спрямоване на створення зручного інструменту, який буде корисним для українських митців у віртуальному середовищі. Основна мета цього програмного застосунку – допомогти користувачам:

- розробляти та заповнювати свої індивідуальні профілі;
- ділитися своїми власними художніми роботами;
- вивчати та розвивати творчі роботи інших;
- взаємодіяти: коментувати, ставити лайки;
- забезпечувати спілкування в режимі реального часу через функцію чату;
- підтримувати онлайн-портфоліо власних художніх робіт.

Основна функція системи – забезпечити легкий, безпечний та стабільний доступ до платформи через веббраузери на різних пристроях. Платформа повинна стабільно функціонувати в періоди високого попиту, зберігати цілісність даних, захищати особисту інформацію користувачів та пропонувати гнучкі адміністративні функції. Система призначена для окремих митців та організацій, що займаються популяризацією української культури.

Вимоги до програмного продукту

Вимоги до функціональних характеристик

Програмний продукт має забезпечувати виконання таких головних функції: реєстрацію та авторизацію користувачів за допомогою JWT, створення та зміну профілю, публікацію мистецької роботи з метаданими, перегляд та пошук робіт із функцією фільтрації, обмін повідомленнями за реальним часом, відображення, а також взаємодію користувачів між собою за допомогою листувань та позначок “подобається”.

Вхідні дані складаються з текстових полів (ім'я, опис, повідомлення) та

зображень (файли робіт), які передаються користувачем у вигляді форми. Вихідними даними будуть побудовані сторінки з візуалізованим вмістом, повідомленнями, сповіщеннями.

Часові характеристики передбачають миттєве оновлення інформації в чаті та на сторінках (до 1 секунди після дій користувача), а також загальне завантаження вмісту не більше 2–3 секунд.

Платформа є стабільною під час одночасного доступу кількох користувачів із забезпеченням збереження даних і вірної зміни стану інтерфейсу.

Вимоги до надійності

Програмний продукт має гарантувати стабільну та безпечну роботу, навіть за умов часткових збоїв, таких як тимчасова втрата зв’язку з сервером або проблеми з доступом до бази даних. У випадку виникнення збоїв система повинна автоматично або вручну відновити свою роботу протягом не більше ніж 5 хвилин. Передбачено обов’язкову перевірку вхідних та вихідних даних на помилки, спроби несанкціонованого доступу або інші порушення. Усі запити до бази даних повинні проходити авторизацію відповідно до рівня прав користувача. Для запобігання втраті інформації реалізується щоденне резервне копіювання основних даних платформи – профілів користувачів, мистецьких робіт, повідомлень та коментарів.

Умови експлуатації

Вебдодаток “ArtUA” розроблений для роботи в середовищі веббраузера (Google Chrome, Mozilla Firefox, Microsoft Edge тощо) на комп'ютерах або ноутбуках з операційною системою Windows, Linux або macOS. Стабільна робота платформи забезпечується за наявності швидкості інтернет-з'єднання не менше 5 Мбіт/с. Обслуговування вимагає періодичних оновлень серверної частини, бази даних та клієнтської частини. Для цього достатньо одного системного адміністратора або розробника, знайомого з роботою з ASP.NET

Core, React, PostgreSQL та інструментами розгортання вебдодатків. Користувачам не потрібна спеціальна технічна підготовка – достатньо загальних навичок роботи з вебресурсами.

Вимоги до складу і параметрів технічних засобів

Робоча станція розробника повинна мати процесор Intel Core i5 або аналогічний, не менше 8 ГБ оперативної пам'яті, жорсткий диск SSD обсягом не менше 256 ГБ, дисплей з роздільною здатністю не менше 1920×1080 та підключення до стабільного підключення до Інтернету з мінімальною пропускною здатністю 10 Мбіт/с.

Для зберігання даних використовується реляційна база даних PostgreSQL з можливістю створення резервних копій та масштабованістю. Ці технічні можливості забезпечують належне функціонування системи в середовищі розробки, тестування та виробництва.

Вимоги до інформаційної і програмної сумісності

Платформа “ArtUA” обробляє вхідні та вихідні дані у форматі JSON з використанням REST API. Основні інформаційні структури охоплюють дані користувачів, профілі авторів, художні роботи, повідомлення.

Програмна реалізація серверної частини виконана на C# із використанням ASP.NET Core, Entity Framework, SignalR, JWT для авторизації та захисту даних. Клієнтська частина розроблена на JavaScript з використанням React. Взаємодія з базою даних відбувається через PostgreSQL.

Забезпечується сумісність компонентів, підтримка стандартів HTTP, авторизація, захист доступу та можливість масштабування. Для документування API використовується Swagger.

Вимоги до маркування і упаковки

Готовий програмний засіб маркується найменуванням – “ArtUA”, версією

випуску, датою останнього оновлення та ідентифікатором розробника. Упаковка програмного забезпечення передбачає архівування вихідного коду та документації у форматі .zip для зручного зберігання та передачі. Крім того, додається інструкція з розгортання системи, файл README з описом структури проєкту та вимогами до запуску.

Вимоги до транспортування і збереження

Програмний продукт “ArtUA” має бути упакований та доставлений у вигляді збережених файлів (.zip) або через систему контролю версій, а саме Git. Доставка програмного продукту має здійснюватися через безпечні засоби зв'язку, такі як SSH або HTTPS, щоб забезпечити конфіденційність та цілісність даних.

Вихідні коди, документи та файли конфігурації необхідно зберігати в захищеному середовищі з регулярним резервним копіюванням. Для забезпечення узгодженості та доступності даних необхідно використовувати централізовані репозиторії (наприклад, GitHub, GitLab або самостійно керований сервер контролю версій).

Вимоги до програмної документації

До складу програмної документації вебплатформи “ArtUA” входять:

- технічне завдання;
- архітектурна документація, що описує структуру системи та її основні компоненти;
- керівництво користувача, яке містить інструкції з встановлення, налаштування та використання платформи;
- коментарі у вихідному коді, що пояснюють логіку основних модулів і алгоритмів;
- інструкція з розгортання системи.

Спеціальні вимоги до документації передбачають її актуальність, повноту, чіткість викладу та зручність для розуміння як розробниками, так і кінцевими користувачами.

Техніко-економічні показники

Орієнтовна річна потреба платформи “ArtUA” становить близько 10 000 активних користувачів та 500 активних митців, які використовують сервіс для просування творчості. Економічна ефективність полягає у зниженні витрат на традиційні виставки та маркетинг за рахунок онлайн-інструментів, що дозволяє скоротити бюджет підтримки мистецьких проектів приблизно на 30-50%.

Переваги “ArtUA” у порівнянні з кращими вітчизняними та зарубіжними аналогами полягають у більш гнучкій архітектурі, використанні технологій (React, ASP.NET Core, SignalR) для покращення інтерактивності, а також адаптації платформи під потреби локального ринку. Це забезпечує кращу масштабованість, швидкість реагування та зручність використання, що підвищує конкурентоспроможність проекту.

Стадії та етапи розробки застосунку “ArtUA”

Розробка вебплатформи “ArtUA” складається з низки послідовних етапів. На першому етапі проводиться збір та аналіз вимог, досліджується предметна область та деталізується технічне завдання. Паралельно деталізується архітектурна документація, яка визначає основні компоненти системи та їх взаємодію.

Наступний етап присвячений власне розробці програмного забезпечення. На цьому етапі розробляється вихідний код, реалізується серверна частина на базі ASP.NET Core з використанням REST API, фронтенд реалізується на React, реалізується база даних PostgreSQL та реалізуються функції реального часу за допомогою SignalR. Критичним компонентом цього етапу є документування коду та структурування документації. Після завершення розробки проводиться внутрішнє тестування та, за потреби, вносяться зміни.

Заключний етап включає повне тестування системи – функціональне, навантажувальне та безпекове. Паралельно розробляється документація для кінцевого користувача та інструкції з розгортання платформи. Після успішного проходження тестів платформа розгортається на робочому сервері. Терміни розробки визначаються на основі термінів виконання дисертації, враховуючи контрольні точки та проміжні звіти.

Порядок контролю і приймання

Прийняття вебплатформи “ArtUA” здійснюється на основі результатів функціонального, інтеграційного та модульного тестування. Основна вимога – повна відповідність технічному завданню, стабільна робота всіх модулів, правильна обробка запитів API та безпечна автентифікація. Робота вважається прийнятою після успішного тестування і наявності всієї супровідної документації.

ДОДАТОК Б

Керівництво користувачу

1. Загальні відомості

Найменування програми: ArtUA

Позначення: ArtUA_2.0

ArtUA – це вебплатформа, розроблена для популяризації та підтримки українського мистецтва. Сервіс забезпечує можливість обміну творчими роботами, замовлення авторських виробів, комунікацію між користувачами, а також сприяє формуванню активної мистецької спільноти.

2. Функціональне призначення

Програма призначена для реалізації онлайн-простору, у якому митці мають можливість:

- публікувати свої роботи (зображення, описи);
- демонструвати твори;
- переглядати інші роботи;
- вести комунікацію через вбудовані чати;
- отримувати сповіщення про активність на платформі (нові повідомлення, вподобання, коментарі тощо).

Користувачі можуть:

- реєструватися та створювати акаунти;
- переглядати й фільтрувати твори мистецтва;
- взаємодіяти з авторами робіт;
- залишати повідомлення та вподобання;
- спілкуватися у приватних чатах.

3. Умови застосування програми

Тип ЕОМ: персональний комп’ютер або ноутбук із доступом до мережі Інтернет.

Операційна система: Windows 10 або новіша, macOS 10.13 або новіша,

Linux (Ubuntu 20.04+).

Оперативна пам’ять: мінімум 4 ГБ.

Процесор: від 2.0 GHz.

Периферійні пристрої: стандартна клавіатура, миша.

Необхідне програмне забезпечення: остання випущена версія браузера (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari).

4. Повідомлення оператору

У процесі взаємодії користувача з платформою ArtUA система генерує низку інформаційних повідомлень, що відображають стан виконання дій, повідомляють про помилки або інформують про успішне завершення операцій. Після успішного входу в систему користувач бачить повідомлення “Вхід виконано успішно”, що свідчить про коректне введення логіна та пароля. У разі, якщо під час входу були введені некоректні дані (наприклад, помилкова електронна адреса або пароль), система видає повідомлення “Невірна електронна пошта або пароль”. При створенні нового облікового запису важливо дотримуватись вимог до складності пароля. Якщо пароль не відповідає встановленим правилам, наприклад, не містить спеціальних символів, з’являється повідомлення “Пароль має містити хоча б один спеціальний символ (!@#$%^&*0,?":0|<>)”. Після збереження змін у профілі, публікації нової роботи чи оновлення вмісту користувачу демонструється підтвердження у вигляді повідомлення “Успішно!” або “Інформацію збережено”.

5. Опис роботи програми

Платформа ArtUA реалізована з урахуванням принципів зручності, доступності та логічної послідовності дій користувача. Інтерфейс розроблено відповідно до стандартів UX/UI-дизайну, що забезпечує інтуїтивне сприйняття елементів системи. Опишемо роботу основних етапів взаємодії користувача з функціональними модулями вебзастосунку.

Для створення нового облікового запису користувач повинен натиснути кнопку “Реєстрація” на головній сторінці платформи або на кнопку “Приєднатися”. Після чого відкривається форма реєстрації (рис. Б.1), де

необхідно ввести ім’я, адресу електронної пошти та пароль, після чого підтвердити реєстрацію. У разі коректного заповнення форми та відсутності помилок акаунт буде створено.

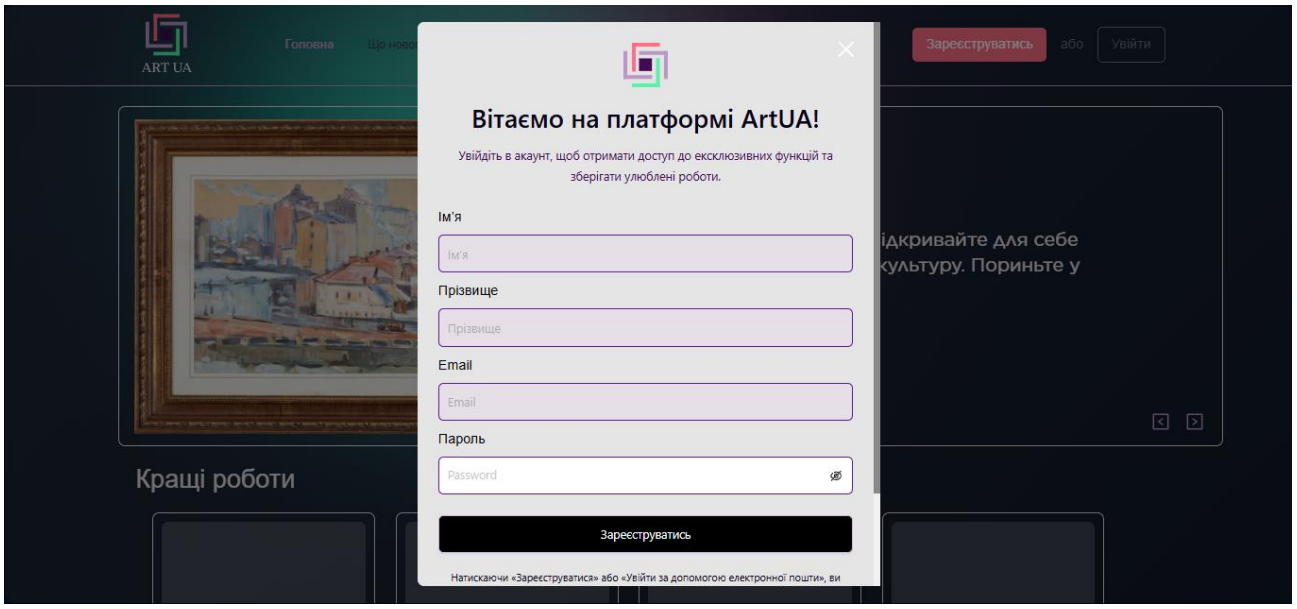

Рисунок Б.1 – Форма реєстрації

Вхід до системи здійснюється через форму авторизації шляхом введення email-адреси та пароля. У разі невірних даних виводиться повідомлення “Невірний логін або пароль”.

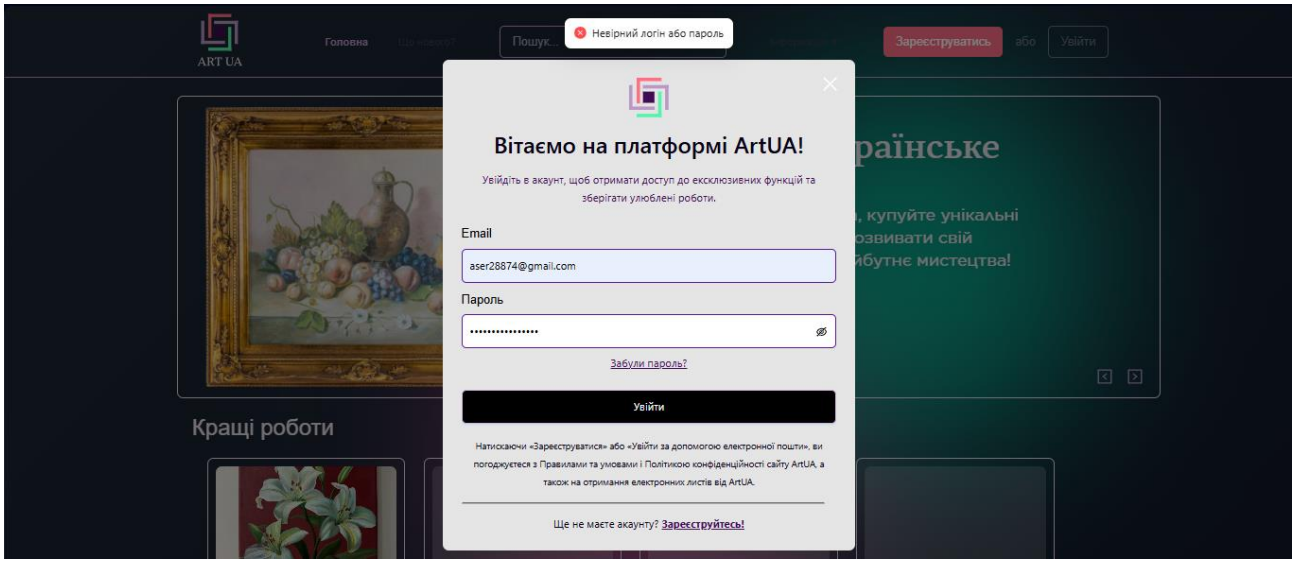

Рисунок Б.2 – Форма авторизації з помилкою

Після входу користувач потрапляє на сторінку особистого профілю (рис. Б.3), яка містить основну інформацію про користувача: ім’я, опис, контактні дані,

кількість робіт, підписників, вподобань тощо.

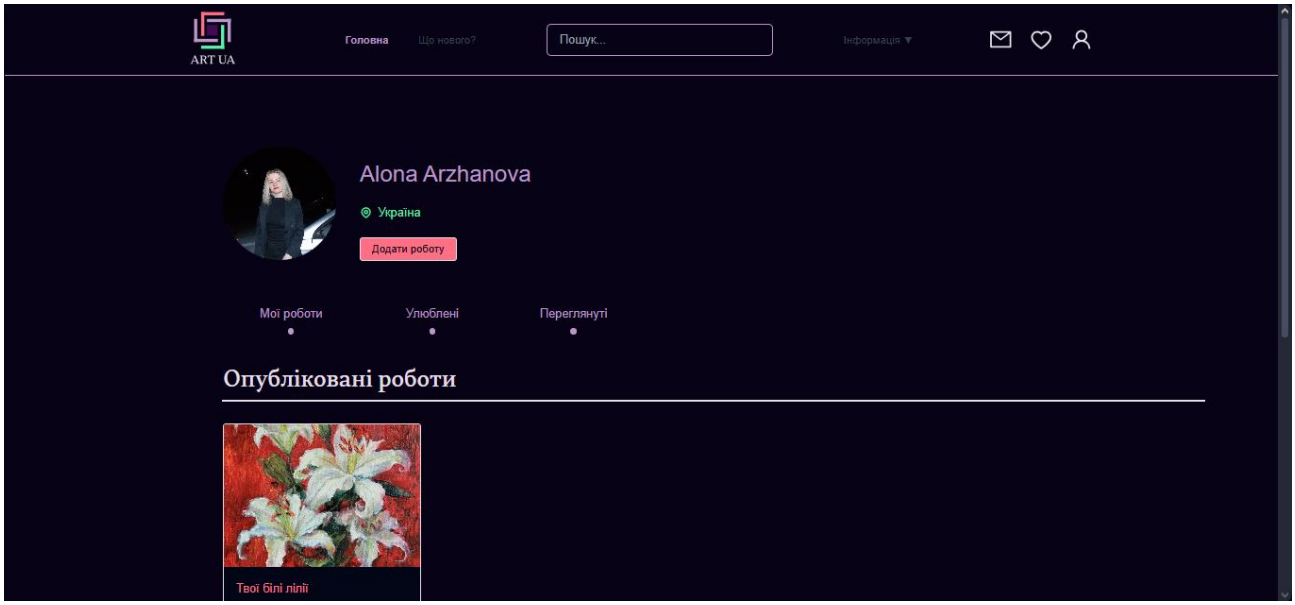

Рисунок Б.3 – Екран профілю користувача

Для зміни персональних даних передбачено окрему вкладку “Налаштування”.

Інтерфейс налаштувань складається з трьох вкладок: “Персоналізація”, “Акаунт”, “Розширені налаштування” (функціональні наразі лише “Персоналізація” та “Акаунт”). У підрозділі Персоналізація користувач може завантажити або змінити аватар, а також ввести чи оновити особисту інформацію: прізвище, ім’я та країну проживання. Усі зміни зберігаються після натискання кнопки “Зберегти зміни”. Інтерфейс реалізовано з урахуванням перевірки обов’язкових полів (позначені *), а також з обмеженням довжини введення.

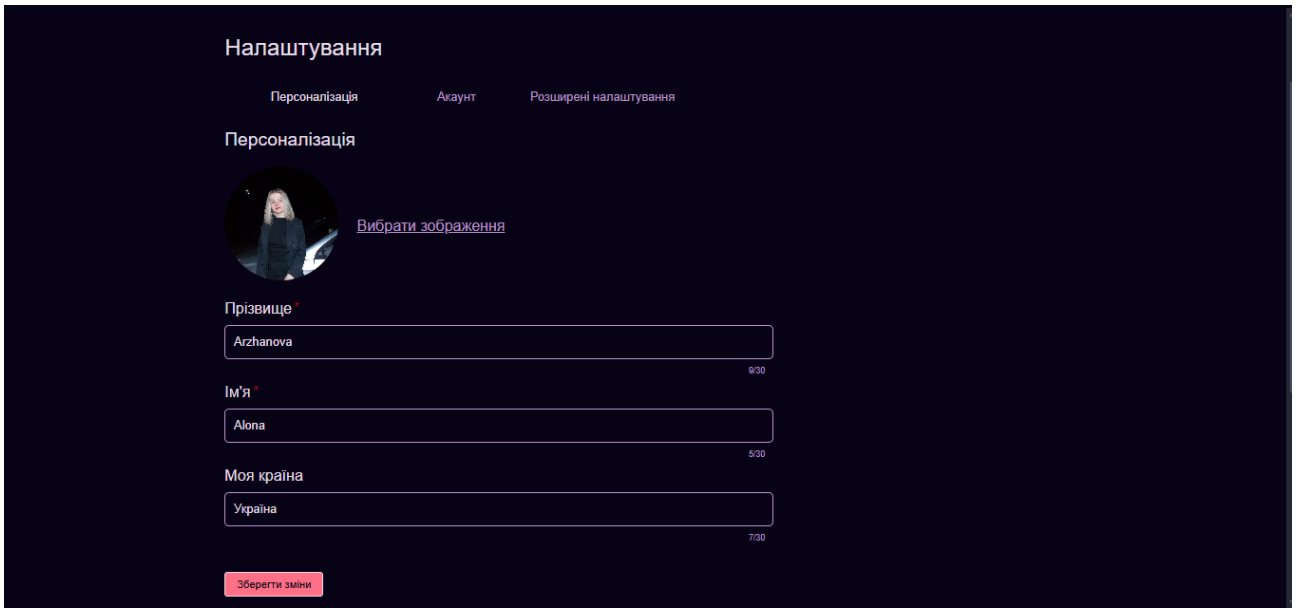

Рисунок Б.4 – Екран “Налаштування”

Для публікації нової роботи необхідно перейти до розділу “Профіль” та натиснути кнопку “Додати роботу”. Відкривається форма (рис. Б.5), у якій потрібно завантажити зображення, ввести назву, опис, вибрати категорію та вказати ціну.

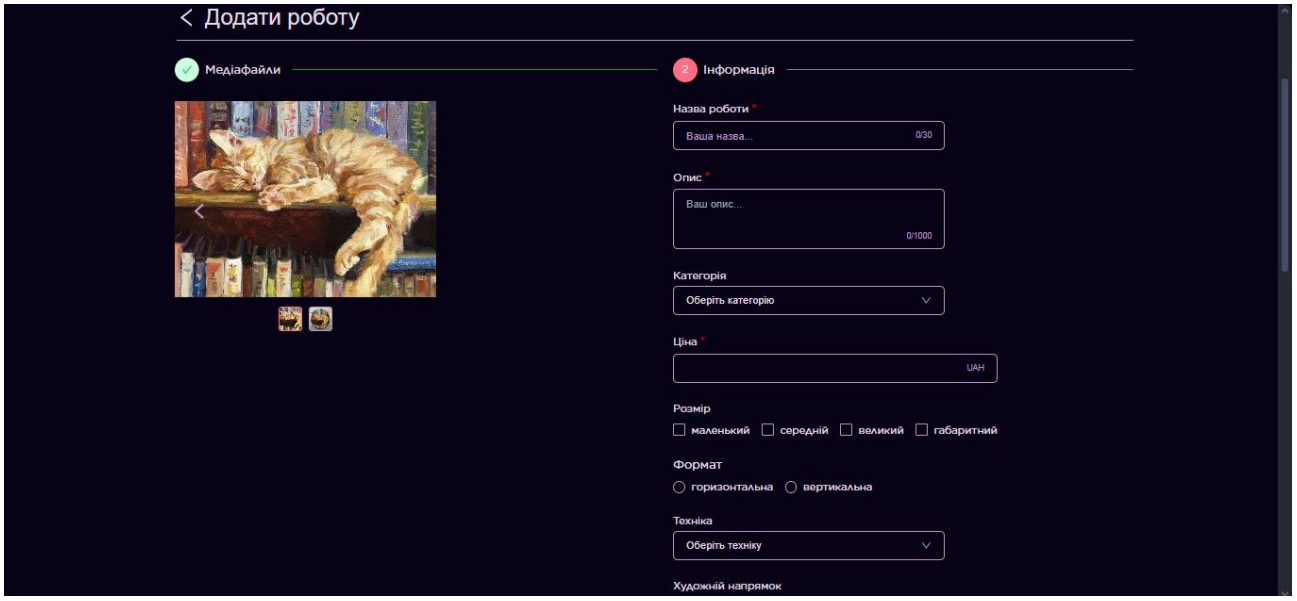

Рисунок Б.5 – Форма створення нового допису

Після збереження робота з'являється у публічному каталозі та доступна для перегляду іншими користувачами.

Для комунікації реалізовано модуль особистих повідомлень (рис. Б.6),

який підтримує обмін текстовими повідомленнями у реальному часі. Також існує система сповіщень (рис. Б.7), яка інформує про нові повідомлення, вподобання та інші активності.

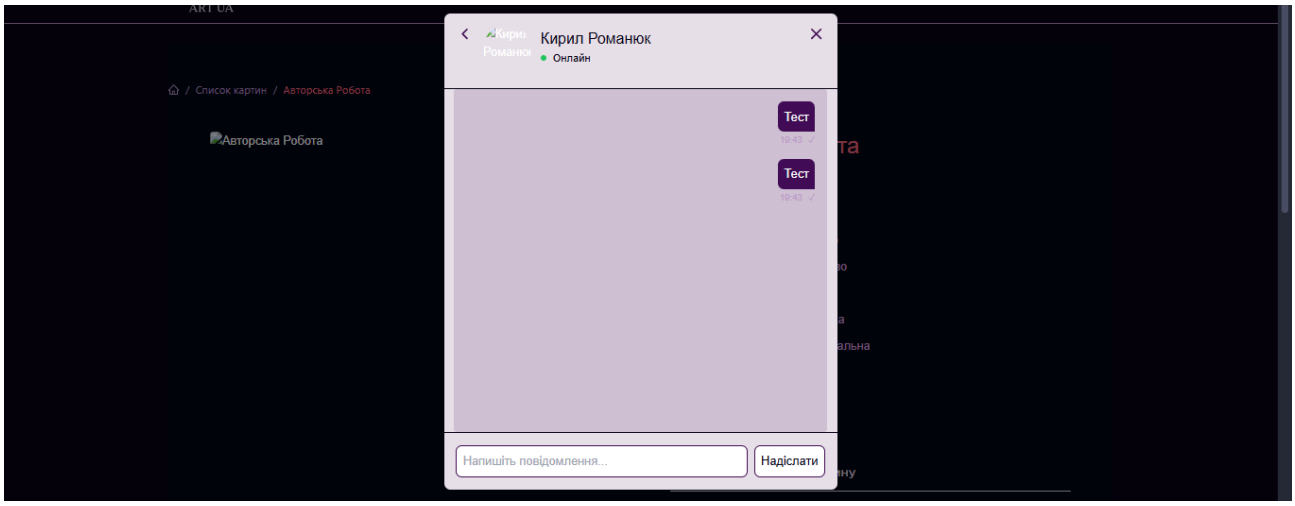

Рисунок Б.6 – Екран чату

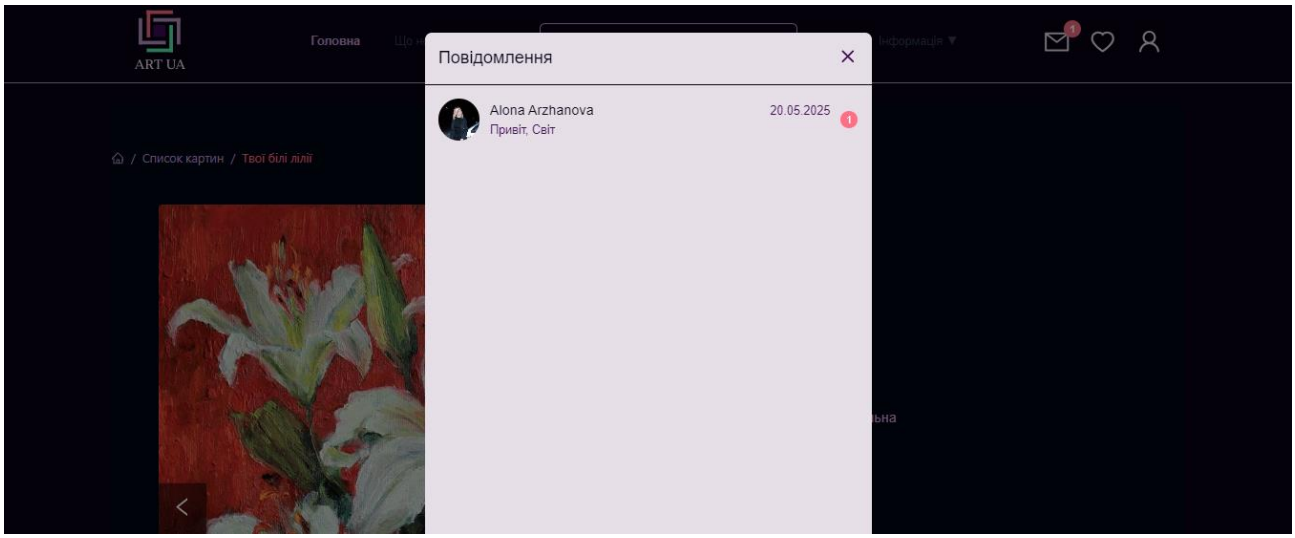

Рисунок Б.7 – Демонстрація сповіщення про нове повідомлення

АНОТАЦІЯ

Аржанова А. С. Розробка вебплатформи для підтримки українських митців та художників засобами вебтехнологій. Рукопис

У роботі розглянуто питання цифрової підтримки художників і митців шляхом створення сучасної вебплатформи. Програмний засіб “ArtUA” реалізований як клієнт-серверна система, що забезпечує інтерактивну взаємодію між авторами творчого контенту та відвідувачами платформи. Проєкт спрямований на популяризацію українського мистецтва в інтернет-середовищі, розвиток цифрових спільнот та полегшення доступу до творчих робіт.

Теоретична частина містить аналіз впливу цифрових технологій на культурну сферу, дослідження потреб митців у цифровому просторі, характеристику архітектурних підходів до побудови платформ, методи забезпечення інтерактивності, безпеки, продуктивності та огляд аналогічних рішень. Особлива увага приділена соціальним мережам як інструменту просування творчості.

У практичній частині здійснено постановку задачі, визначено призначення, функціональні та нефункціональні вимоги до системи. Обґрунтовано вибір інструментів розробки (React, ASP.NET Core, PostgreSQL, SignalR, Entity Framework, JWT). Детально описано реалізацію основних функціональних модулів: авторизації, керування профілями, галереї творчих робіт, системи чату на основі WebSocket, а також методів тестування й налагодження. Надано рекомендації щодо впровадження платформи та її масштабування в майбутньому.

У процесі реалізації використано такі інструменти: React, ASP.NET Core, PostgreSQL, SignalR, Entity Framework, JWT, Visual Studio, GitHub.

Ключові слова: вебплатформа, митці, цифрова підтримка, React, ASP.NET Core, SignalR, PostgreSQL, культура, інтерактивність, користувацька взаємодія.