ДЕРЖАВНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ

УНІВЕРСИТЕТ

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка інформаційної системи з підтримки процесу організації художніх виставок»

Студентки 4 курсу, 8 групи,
спеціальності
122 «Комп'ютерні науки» _____ *підпис студента* Кондратенко Аліни Іванівни

Науковий керівник
кандидат технічних наук,
доцент _____ *підпис керівника* Томашевська Тетяна Володимирівна

Гарант освітньої програми
кандидат технічних наук,
доцент _____ *підпис керівника* Демідов Павло Георгійович

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«12» грудня 2022р.

Завдання
на випускну кваліфікаційну роботу (проект) студентці

Кондратенко Аліни Іванівни
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи (проекту)
«Розробка інформаційної системи з підтримки процесу організації художніх виставок»
Затверджена наказом ректора від *«09 » грудня 2022 р. № 3332*
2. Строк здачі студентом закінченої роботи *30 травня 2023 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: обґрунтування та розробка інформаційної системи з підтримки процесу організації художніх виставок.
Об'єкт дослідження: процес розробки автоматизованої інформаційної системи для забезпечення процесу організації художніх виставок.
Предмет дослідження: інформаційна система з підтримки процесу організації художніх виставок створена на основі web-технологій.
4. Перелік графічного матеріалу
_____
_____
_____
_____
5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Томашевська Т.В.	15.12.2022 р.	15.12.2022 р.
2	Томашевська Т.В.	15.12.2022 р.	15.12.2022 р.
3	Томашевська Т.В.	15.12.2022 р.	15.12.2022 р.

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛІЗ ІНФОРМАЦІЙНИХ ПРОЦЕСІВ.

1.1 Особливості використання web-застосунків для підтримки процесів в організації, види та принципи роботи web-застосунків

1.2 Огляд існуючих систем для організації виставок

1.3 Постановка задачі розробки інформаційної системи для організації виставок

РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ(ІС) ДЛЯ ОРГАНІЗАЦІЇ ВИСТАВОК

2.1 Формування вимог до інформаційної системи для забезпечення організації виставок

2.2 Розробка проекту інформаційної системи для підтримки процесу організації виставок

2.3 Архітектура web-застосунків

2.4 Технології для розробки клієнтської складової

2.5 Технології для розробки серверної складової

РОЗДІЛ 3. РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ З ПІДТРИМКИ ОРГАНІЗАЦІЇ ХУДОЖНІХ ВИСТАВОК У ВИГЛЯДІ WEB-ЗАСТОСУНКУ

3.1. Розробка інтерфейсу та дизайну

3.2 Розробка бази даних для ІС

3.3 Розробка функціонального наповнення web-застосунку

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*04.10.2022*	*04.10.2022*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*15.12.2022*	*15.12.2022*
3	*Вступ*	*03.02.2023*	*03.02.2023*
4	*РОЗДІЛ 1. Опис предметної області та аналіз інформаційних процесів*	*28.02.2023*	*28.02.2023*
5	*РОЗДІЛ 2. Проектування інформаційної системи(ІС) для організації виставок*	*06.04.2023*	*06.04.2023*
6	*РОЗДІЛ 3. Розробка інформаційної системи з підтримки організації художніх виставок у вигляді web-застосунку*	*12.05.2023*	*12.05.2023*
7	*Висновки*	*15.05.2023*	*15.05.2023*
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*30.05.2023*	*30.05.2023*
9	*Попередній захист випускної кваліфікаційної роботи*	*31.05.2023 - 01.06.2023*	*31.05.2023 - 01.06.2023*
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*02.06.2023*	*02.06.2023*
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*05.06.2023*	*05.06.2023*
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15» грудня 2022 р.

Керівник випускної кваліфікаційної роботи (проекту) Томашевська Т.В.
(прізвище, ініціали, підпис)

Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

Завдання прийняв студент-дипломник Кондратенко А.І.
(прізвище, ініціали, підпис)

9. Відгук керівника випускної кваліфікаційної роботи (проекту)

_____

Керівник випускної кваліфікаційної роботи (проекту)
_____ 30.05.2023 р.
(підпис, дата)

10. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента _____
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми_____Демідов П.Г.
(підпис, прізвище, ініціали)

Завідувач кафедри _____Пурський О.І.
(підпис, прізвище, ініціали)

«_____»_____2023 р.

Анотація

Випускна кваліфікаційна робота направлена на дослідження процесу розробки інформаційної системи з підтримки процесу організації художньої виставки. В ході роботи було теоретично обґрунтовано загальні положення, проведено аналіз предметної області, сформована концепція та чіткі вимоги до інформаційної системи. Розроблений проект інформаційної системи та створено web-застосунок для організації художніх виставок на основі інформаційної системи.

У перший розділ присвячений теоретичному дослідженню щодо інформаційних системи, описуються web-застосунки, проведено огляд систем-аналогів та аналіз предметної сфери, поставлено завдання і мету.

У другому розділі сформовано вимоги до інформаційної системи, розроблено технічне завдання, розглянуто процес створення художньої виставки, розроблено проект інформаційної системи.

У третьому розділі розроблено дизайн користувацького інтерфейсу, розроблено базу даних для інформаційної системи, розроблено клієнтську сторону й основні функції інформаційної системи.

Ключові слова: інформаційна система, база даних, MySQL, моделювання, HTML, CSS, Next.js, художні виставки, процес організації.

Anotation

The graduation qualification work is directed to the study of the process of developing an information system to support the process of organizing an art exhibition. During the course of the work, the general provisions were theoretically substantiated, the analysis of the subject area was carried out, the concept and clear requirements for the information system were formed. The project of information system was developed and the web application, that including the information system for the process organization of art exhibitions, was created.

The first chapter is devoted to theoretical research on information systems, web applications are described, an overview of similar systems and an analysis of the subject area are carried out, tasks and goals are set.

In the second section, the requirements for the information system were formed, the technical task was developed, the process of creating an art exhibition was considered, and the project of the information system was developed.

In the third section, the design of the user interface is developed, the database for the information system is developed, the client side and the main functions of the information system are developed.

Keywords: information system, database, MySQL, modeling, HTML, CSS, Next.js, art exhibitions, organization process.

ЗМІСТ

ВСТУП

Інформаційні технології розвиваються та розширюються на усі сфери людського життя. На момент написання роботи суспільство живе у період активного розвитку технологій, застосування програмних засобів є необхідністю задля виживання бізнесів на ринку. Зростання бізнесу є неможливим без введення в дію інформаційних систем, тому кожна організація має відділ, який впроваджує цифрові технології для прискорення і полегшення функціонування. Особливо активно використовують інформаційні системи для покращення управління компанією в цілому та її бізнес-процесам, а також для вдосконалення взаємодії персоналу всередині компанії, навчання робітників, комунікації та співпраці з клієнтами, збору, зберігання, обробки інформації, аналізу та візуалізації даних, що допомагає приймати кращі рішення. Інформаційні системи вводять в роботу компанії у вигляді web-додатків. Web-застосунки є поширеними програмними засобами, які активно використовуються для спрощення обліку роботи та даних, значно покращують і пришвидшують робочий процес, економлять людям час, зменшують витрати фінансів та полегшують процеси планування та проектування. Web-додатки у організаціях використовуються як сайт для інформування, корпоративна система, хмарний сервіс.

Для менеджменту заходів пов’язаних з мистецтвом розроблена велика кількість веб-сервісів і додатків, які допомагають у вирішенні питань організації, маркетингу, контролю продажів білетів і фінансів, роблять процес планування та контролю комфортним. Організація художньої виставки є складним та комплексним проектом, який містить у собі велику кількість задач таких як: вибір та бронювання місця проведення на певний період часу, керування бюджетом, організація аукціонів або продажів, рекламна діяльність, облік робіт, зберігання інформації про відвідувачів, митців, контроль прийому робіт та монтажу експозицій, організація відкриття виставки, збір відгуків, звітність і аналіз заходу, кожен з цих процесів потребує уваги і ретельного моніторингу. Для полегшення роботи з великим об’ємом

завдань, мінімізації фінансових витрат, поліпшення контролю та оптимізації пошуку місць для організації виставок має бути розроблена інформаційна система.

Актуальність тематики дослідження полягає в поширеності автоматизації процесів за допомогою інформаційних технологій, а також у необхідності оптимізації процесу організації художніх виставок, що дозволить планувати мистецькі заходи швидко, ефективно, а також допоможе компанії або художнику повноцінно контролювати весь процес.

Метою даного дослідження є обґрунтування та розробка інформаційної системи з підтримки процесу організації художніх виставок, яка дозволить митцям або їх менеджерам прискорити етап організації виставки, а саме: пошук місць для виставки, бронювання, розрахунок вартості білету, бюджету, контроль за додатковими послугами.

У ході розробки основними **завданнями** є:

- аналіз предметної області та продуктів-аналогів;
- визначення вимог до інформаційної системи, розробка технічного завдання;
- розробка проекту і структури інформаційної системи;
- розробка клієнтського інтерфейсу і функцій системи;
- розробка бази даних для системи.

Об’єкт дослідження: процес розробки автоматизованої інформаційної системи для забезпечення процесу організації художніх виставок.

Предмет дослідження: інформаційна система з підтримки процесу організації художніх виставок створена на основі web-технологій.

Методи дослідження: Для дослідження теми були використані: метод порівняння та протиставлення, методи спостереження, метод декомпозиції системи, моделювання бізнес-процесів та об'єктний аналіз, методи розробки веб-застосунків.

Практична значущість: Дані отримані у цьому дослідженні в подальшому можуть бути використанні для розробки повноцінної інформаційної системи з розширеним функціоналом, яка зможе оптимізувати процес організації виставок, за рахунок автоматизації і переведення усіх організаційних процесів й необхідної інформації в один веб-застосунок або додаток, що допоможе менеджерам у сфері мистецтва або художникам самостійно підвищить ефективність планування і управління виставками, взаємодії з учасниками і відвідувачами, вдосконалити контроль фінансів, аналітики, продажу квитків, а також вдосконалити маркетингову сторону процесу організації художньої виставки.

Структура та обсяг випускної дипломної роботи: Випускна дипломна робота складається з 3 розділів, висновків, списку літературних джерел, що складається з 26 найменувань. У повному обсязі робота займає 76 сторінок, містить 45 рисунків, 6 таблиць, 5 додатків.

РОЗДІЛ 1.
ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛІЗ ІНФОРМАЦІЙНИХ ПРОЦЕСІВ

Через активний розвиток інформаційних технологій, збільшення кількості даних, кадрів компаній та кількості функцій, кожна з організацій потребує використання сучасних інструментів для покращення роботи бізнесу. Управління процесами в організаціях відбувається за рахунок залучення в роботу інформаційних систем. Інформаційна системою називають сукупність методів і засобів персоналу, використовуваних для зберігання, обробки і видачі інформації в інтересах досягнення поставленої мети[1]. Інформаційні системи використовують для реалізації проектів, що складаються з великої кількості процесів, функцій, елементів, компонентів, які мають різноманітні цілі і властивості.

Основні переваги використання інформаційних систем для організації процесів:

- Інформаційні системи дозволяють зберігати велику кількість різноманітних даних, трансформувати їх і маніпулювати ними;
- Інформаційні системи використовують алгоритми, обчислення для обробки даних, через що широко використовуються для аналізу, перевірки, сортування та фільтрації інформації, ведення звітності, а також дозволяють візуалізацію даних, прогнозування та допомагають у налаштуванні бізнес-аналітики;
- Інформаційні системи допомагають з вирішенням проблеми повторюваних завдань у компанії, і роблять їх виконання більш швидким і точним;
- Інформаційні системи використовують клієнтські інтерфейси, які є зручними та інтуїтивно зрозумілими, для взаємодії користувача і системи, що підвищує ефективність роботи;

- Інформаційні системи гнучкі і адаптивні, і можуть змінюватися в залежності від потреб бізнесу.

Інформаційна система формується з таких основних складових: прикладне програмне забезпечення, технічні засоби, персонал, база даних і система управління базами даних(СУБД), організаційно-методичне (нормативне) забезпечення[5,12]. Система складається з усіх комп'ютерних програм, функцій, що допомагають у реалізації завдань у заданій для інформаційної системи предметній області, обчислювальної техніки, периферійного обладнання, засобів зчитування даних, засобів управління процесами, а також обладнання для забезпечення обміну інформацією. Всі інформаційні системи мають перелік організаційно-технічних заходів і дотримуються правових норм та документів. Життєвий цикл інформаційної системи поділяється на етапи: аналіз, проектування, розробка, тестування, впровадження, супровід, діагностика роботи, модернізація[5, 37].

Інформаційні системи є програмним продуктом, який реалізується за допомогою web-застосунків.

1.1 Особливості використання web-застосунків для підтримки процесів в організації, види та принципи роботи web-застосунків

Web-застосунки – це інтерактивні програми, які виконуються на сервері та взаємодіють із користувачами через веб-браузер та клієнтський інтерфейс. Додатки дають змогу користувачам виконувати різноманітні функції: зберігання інформації, обробка даних, спілкування з клієнтами, і взаємодія у команді, купівля або продаж товарів.

Web-застосунки розробляються за допомогою мов програмування, таких як Javascript, PHP або Python, мови гіпертекстової розмітки (HTML) та каскадних таблиць стилів (CSS), а також додаткових фреймворків та бібліотек.

Веб-застосунки створюють з різною метою: для управління бізнес-процесами в компанії, для оптимізації взаємодії з клієнтами, для обміну

інформацією, для автоматизації процесів, для покращення взаємодії робітників всередині команди, для керування файлами, що знаходяться в хмарному сховищі, для регулювання доступу до ресурсів, для управління фінансовими потоками.

Особливість використання web-застосунків для підтримки процесів в організації полягає у тому, що веб-додатки мають певну кількість вбудованих функцій, наприклад, робота з замовленнями, таблицями, графіками, аналітикою, розрахунками, авторизація користувача та зберігання особистих даних, на відміну від веб-сайтів, які мають в основному інформативний характер.

Використання web-застосунків для підтримки процесів в організації підвищує швидкість роботи у компанії. Доступ до веб-додатків можна отримати з мобільного або комп’ютерного пристрою, підключеного до Інтернету. Застосунки є зручними і доступними для кожної людини, оскільки їх не потрібно встановлювати на локальному комп'ютері, і можна використовувати на будь-якому пристрої або операційній системі, що підтримує браузер і має інтернет з’єднання. Веб-застосунки дозволяють зберігати інформацію та синхронізувати її між різноманітними пристроями, користувачами, забезпечують спільну роботу. Web-застосунки інтегрують у свої системи інші служби і додаткове програмне забезпечення, наприклад, платіжні системи, системи для забезпечення відео зв’язку або чату.

Web-застосунки працюють за принципом «клієнт-сервер», який представляє собою поділ системи на дві частини:

- Клієнтська частина: Код на стороні користувача працює з функціями інтерфейсу користувача, наприклад: функції кнопок, форм, вікон. Якщо користувач натискає посилання на застосунок, то веб браузер завантажує код клієнтської сторони та редагує графічні елементи, текст, текст для взаємодії з користувачами. Для розробки інтерфейсу використовують CSS, HTML, JS.

- Серверна частина: Відповідає на HTTP-запити і знаходиться на веб сервері. Серверний компонент складається з двох частин – логіки додатка(центр керування веб-застосунком) і бази даних(місце зберігання інформації)[7]. Сервер обробляє запити клієнта і надсилає відповідь. Запити можуть стосуватися питань збереження, виведення, редагування нових даних.

Web-сервер – програмне забезпечення у серверній частині веб-додатку, що виконує функції прийняття і обробки запитів користувача, у відповідь на них видає інформацію, яка згодом відображається у браузері клієнта. Сервер відповідає також за автентифікацію та безпеку користувача.

Базою даних називають набір взаємопов'язаних логікою даних спільного використання, що призначені для задоволення інформаційних потреб[4]. Бази даних є корпоративним ресурсом, що використовується багаторазово різними користувачами, який має у собі дані та їх опис.

Веб-застосунок працює за такою схемою(рис.1.1):

1) Користувач створює запит до веб-серверу, який складається з центру керування застосунком та бази даних.
2) Центр керування отримує запит і скеровує його на базу даних.
3) Інформація береться з бази даних та направляється до центру керування.
4) Веб сервер відправляє інформацію користувачу і вона з’являється у інтерфейсі[7].

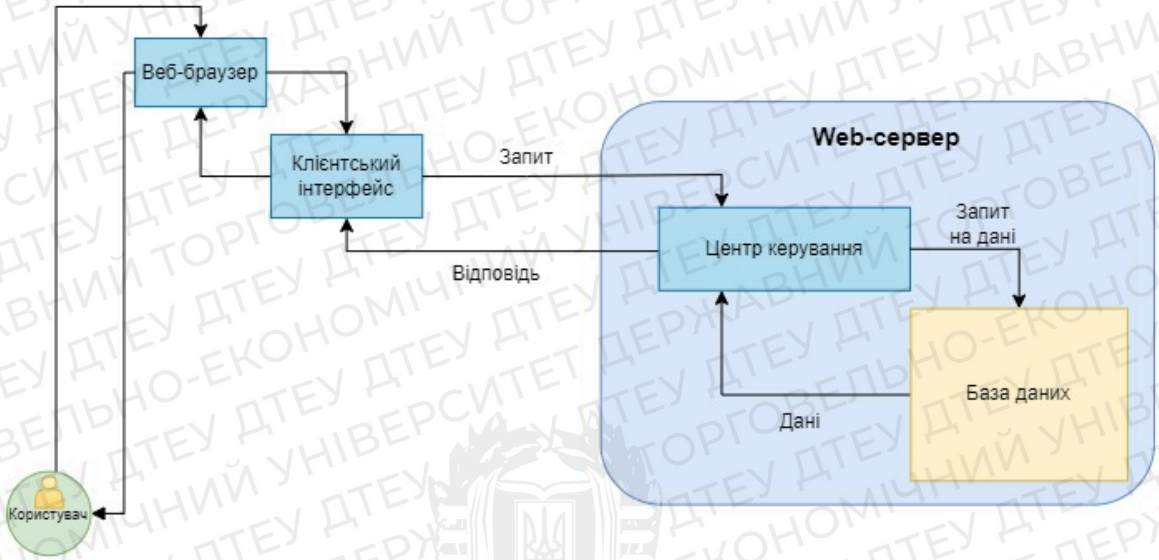

Рис.1.1 Схема роботи web-застосунку

Згідно з призначенням web-застосунки поділяються на види:

- Сайт – це найпростіший вид веб-додатків, що складається з HTML-сторінок, які можуть вміщувати у собі різні елементи, такі як медіа-контент(відео, анімації, аудіо та зображення), контактні форми і статичні текстові сторінки.
- E-commerce додаток – це застосунки, які використовуються для продажу-купівлі речей, послуг через інтернет-мережу. Веб додатки мають інтерфейс для клієнта та панель адміністратора для менеджера, в якій є функції для керування замовленнями.
- Хмарні сервіси – застосунки, призначені для зберігання або обробки інформації та різних типів даних.
- Соціальні мережі – застосунки, які забезпечують комунікацію людям через інтернет.
- CRM-системи – застосунки, які використовуються компаніями для забезпечення підтримки бізнес-процесів, таких як управління відділами, проектами, спілкування з клієнтами.
- Корпоративні портали – це комплексні платформи, які надають користувачам доступ до корпоративних даних та найрізноманітнішого

сервісу[7]. Ці портали дозволяють працювати одночасно з декількома додатками компанії (поштою, CRM, чатом), використовуючи один і той самий інтерфейс.

1.2 Огляд існуючих систем для організації виставок

Велика кількість компаній, які займаються організацією заходів різного типу, в тому числі й мистецьких виставок, розміщують інформацію про перелік їх послуги на веб ресурсі. Деякі організації мають свою систему(веб-додаток), яка допомагає виконувати певні функції та є доступною іншим користувачам тільки після реєстрації і оплати. На сайті про веб-застосунок зазвичай розміщена інформація про можливості системи, з чим працює система, відгуки користувачів та відповіді на постійні запитання. Після реєстрації та авторизації людина може користуватися у своїх цілях панеллю адміністратора, яка містить всі функції інформаційної системи.

В ході дослідження було знайдено декілька аналогічних прикладів інформаційних систем для організації різноманітних типів заходів, в тому числі і художніх виставок:

1) Web - застосунок Eventzilla. Даний сервіс є платформою для організації заходів, і включає в собі такі функції як забезпечення реєстрації на подію, продаж квитків, розробка веб-сайту події, просування по електронній пошті(розсилка). Також у ньому є функції аналітики та інші інструменти для маркетингу;

На головній сторінці основного сайту Eventzilla(рис.1.2) представлена інформація про функції платформи:

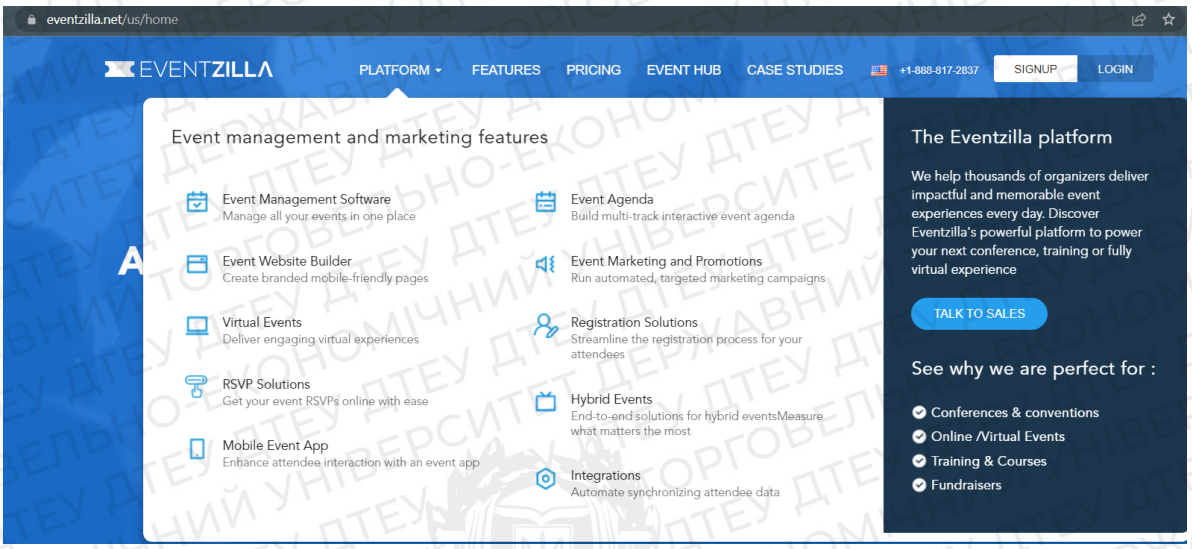

Рис. 1.2 Головна сторінка web-сайту Eventzilla

Після реєстрації у сервісі є можливість створити нову подію. Після натискання кнопки для користувача відкривається форма для вводу даних(рис.1.3), в якій потрібно вказати назву заходу, тип заходу, часовий пояс, зображення для анонсу заходу, організатора заходу, інформацію про організатора та опис самої події.

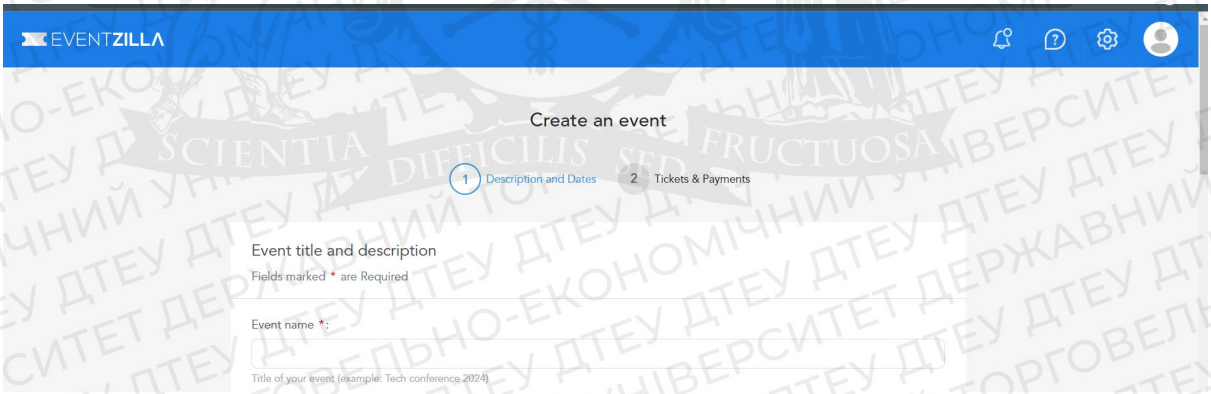

Рис. 1.3 Форма для створення події в Eventzilla

Випадаючий список для вибору типу події(рис.1.4) пропонує велику кількість варіантів: концерт, жива музика, художня виставка, тренінг, спортивна подія і т.д.

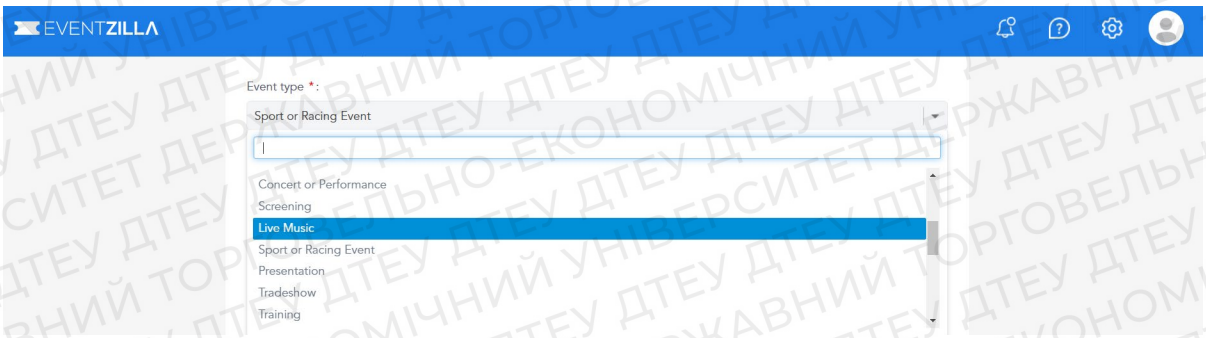

Рис. 1.4 Випадаючий список для вибору типу події в Eventzilla

У вкладці «Description and Dates» користувач може обрати місце проведення заходу з трьох варіантів(рис.1.5): офлайн подія, онлайн подія, гібридна(поєднує онлайн та живий захід) подія. Якщо обрано офлайн подію, тоді у користувача додатково просять вказати дані про місце проведення події. У вкладці «Tickets and Payments» користувач може обрати тип білету: безкоштовний білет, платний білет, благодійний внесок або через товар. На кожен тип заповнюється форма у якій є опис білету, доступна кількість, його вартість, термін продажу білетів, а також код доступу(промо-код), за яким можна отримати доступ до події безкоштовно.

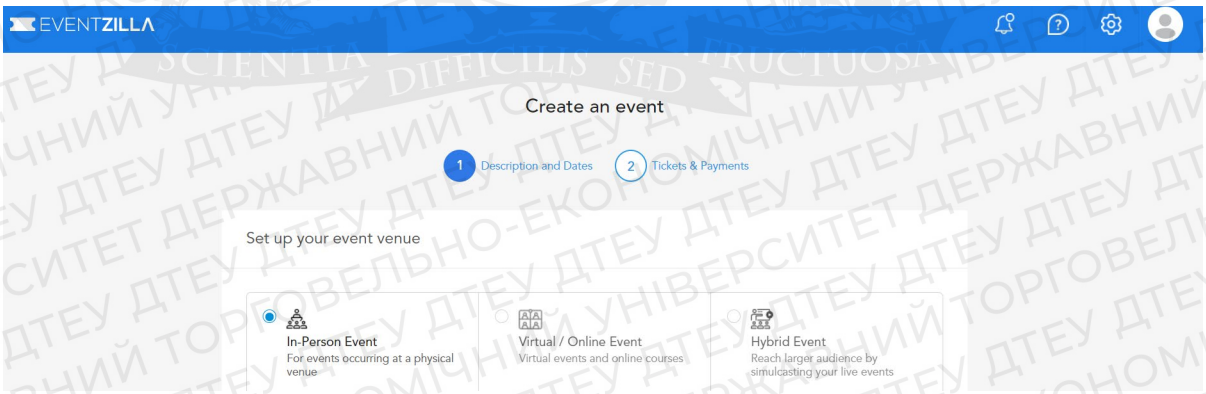

Рис. 1.5 Вибір місця проведення

Після оформлення події користувачу відкритий доступ до панелі адміністратора(рис.1.6), у якій є безліч функцій для контролю заходу, відвідувачів, реєстрації, фінансів, розкладу події, реклами та аналітики, в системі є звітність і інструменти для створення збору відгуків(анкетування, опитування).

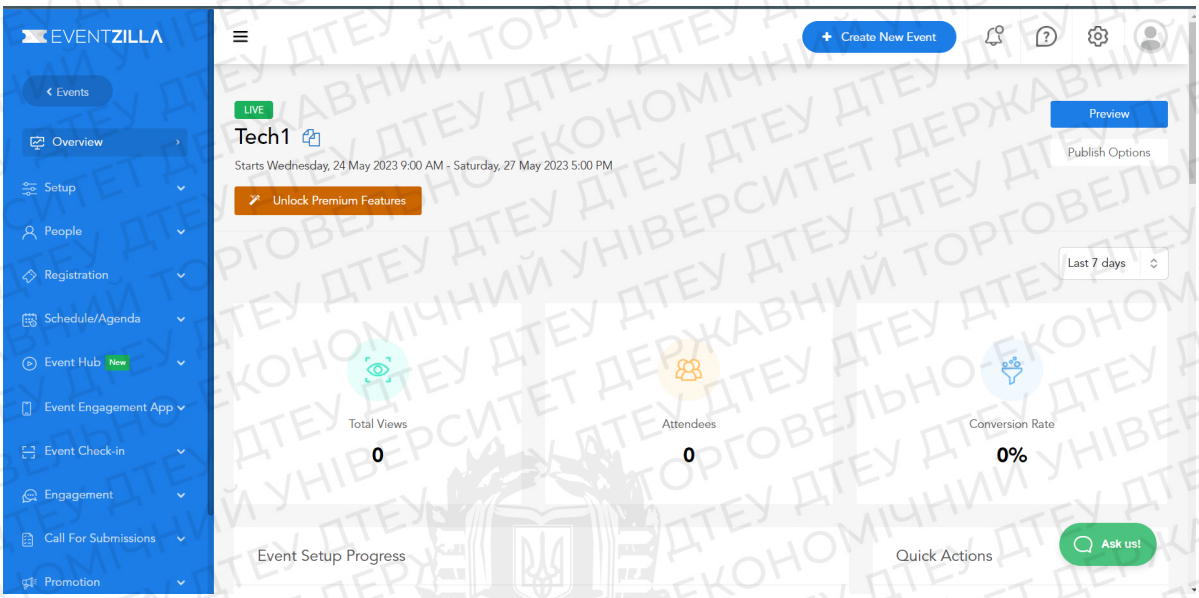

Рис. 1.6 Панель адміністратора у платформі Eventzilla

Недоліками цієї системи з точки зору організації виставок:

- відсутній контроль експонатів;
- немає контролю за додатковими послугами(кейтеринг, монтаж експонатів, транспортування, продаж).

Посилання на web-сервіс: https://www.eventzilla.net/us/home.

2) Web-сервіс Artsy. Онлайн платформа виступає великою базою художніх робіт, галерей, художників, виставок. Для художників у веб-застосунку після реєстрації, є можливість виставляти свої роботи на продаж або просувати свої виставки у арт-суспільстві. Цей застосунок також призначений для колекціонерів, скориставшись ним колекціонер може придбати якийсь витвір мистецтва, або організувати аукціон.

На головній сторінці Artsy(рис.1.7) у навігаційній панелі є розділи: Художники(«Artists»), Художні роботи(«Artworks»), Аукціони(«Auctions»), Кімнати для перегляду(«Viewing Rooms»), Галереї(«Galleries»), Ярмарки(«Fairs»), Шоу(«Shows»), Музеї(«Museums»). У верхній панелі сайту є кнопки для входу до облікового запису або реєстрації.

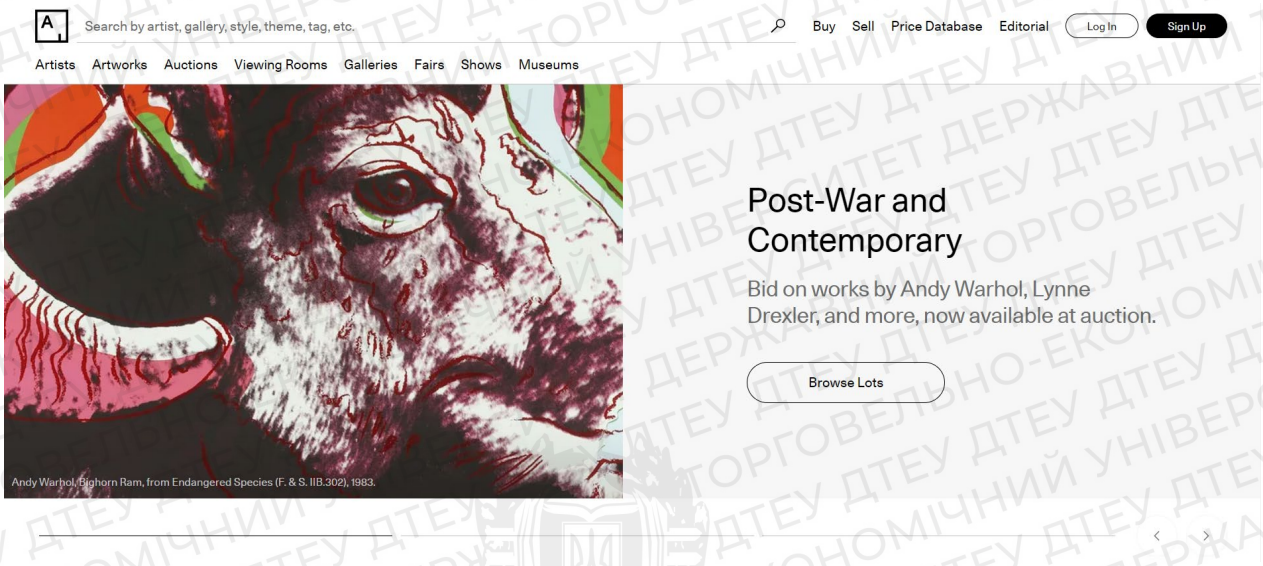

Рис. 1.7 Головна сторінка web-сайту Artsy

У верхній частині сайту є функція «Buy», при натиску на яку відкривається сторінка з усіма доступними для покупки картинами(рис.1.8), зліва знаходиться панель, що дозволяє налаштувати фільтр пошуку за автором, типом мистецтва, рідкістю, ціною, розміром, шляхом покупки, матеріалом, національністю митця, локацією, кольором, періодом, галереєю, також є інструмент для пошуку робіт «Price Database», де щоб знайти певну роботу варто ввести ім'я, фамілію автора, розмір і тип роботи.

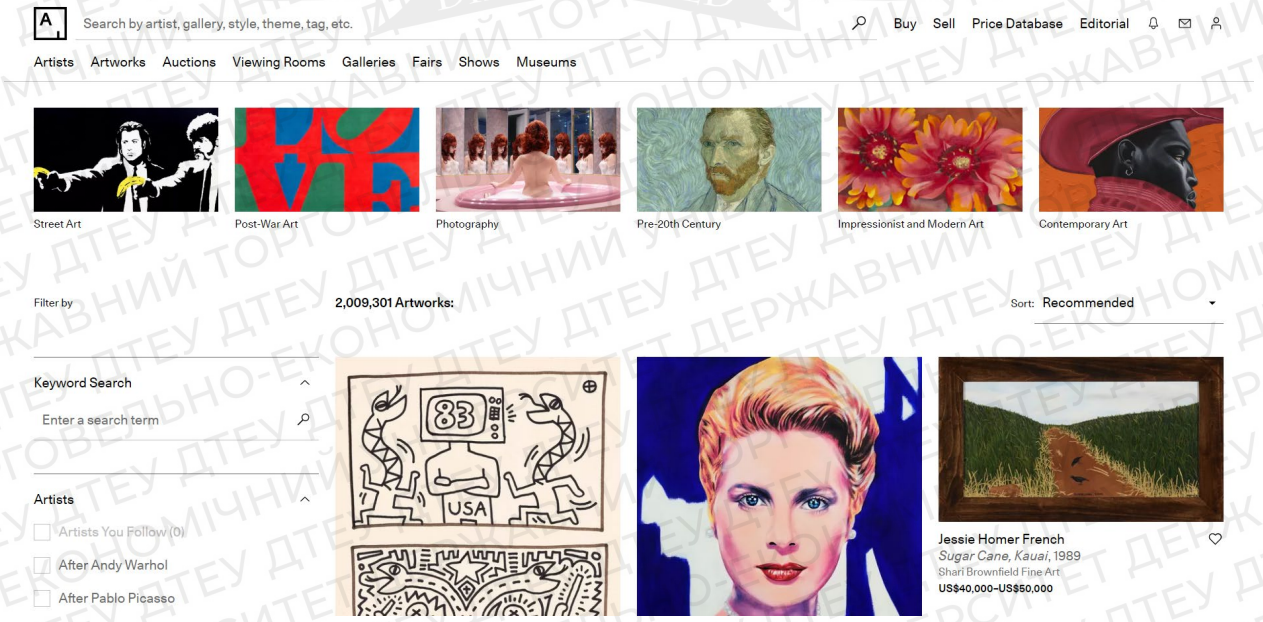

Рис. 1.8 Сторінка з усіма доступними для покупки картинами Artsy

Сервіс дозволяє художникам створювати власний обліковий запис, формувати колекції своїх картин, продавати роботи через сайт, рекламувати свої заходи та приймати участь у актуальних виставках.

Недоліками цієї системи з точки зору організації виставок:

- відсутність функціоналу планування виставки.

Посилання на web-застосунок: https://www.artsy.net/

3) Web-застосунок Eventbrite. Цей сервіс складається з сайту-лендінгу та сайту, який виступає інструментом для організації подій. За допомогою цього інструменту управління подіями можна створювати і рекламувати свої заходи, а також продавати квитки і відстежувати відвідуваність. Крім того, він надає можливості для email-маркетингу та аналітики подій. На головній сторінці основного сайту Eventbrite (рис.1.9) можна побачити інформацію про функціональні можливості інструменту для планування, його переваги, відгуки користувачів, для організації, яких заходів цей застосунок підходить.

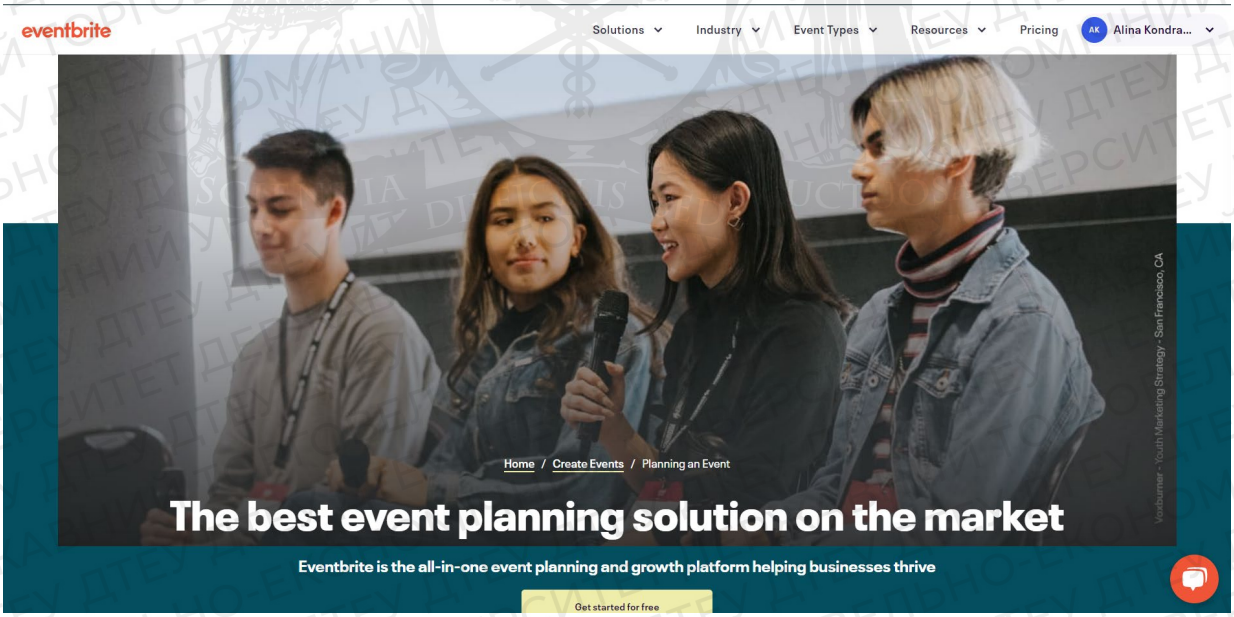

Рис. 1.9 Головна сторінка сайту Eventbrite

Сайт для управління заходами має вигляд CRM-системи у вигляді дошки(рис.1.10). На головній сторінці є можливість створити івент, а також є чек-лист для послідовної організації події. Система має панель управління у якій присутні розділи: Події, Замовлення, Маркетинг, Звіти, Фінанси, Налаштування організації, Інформація.

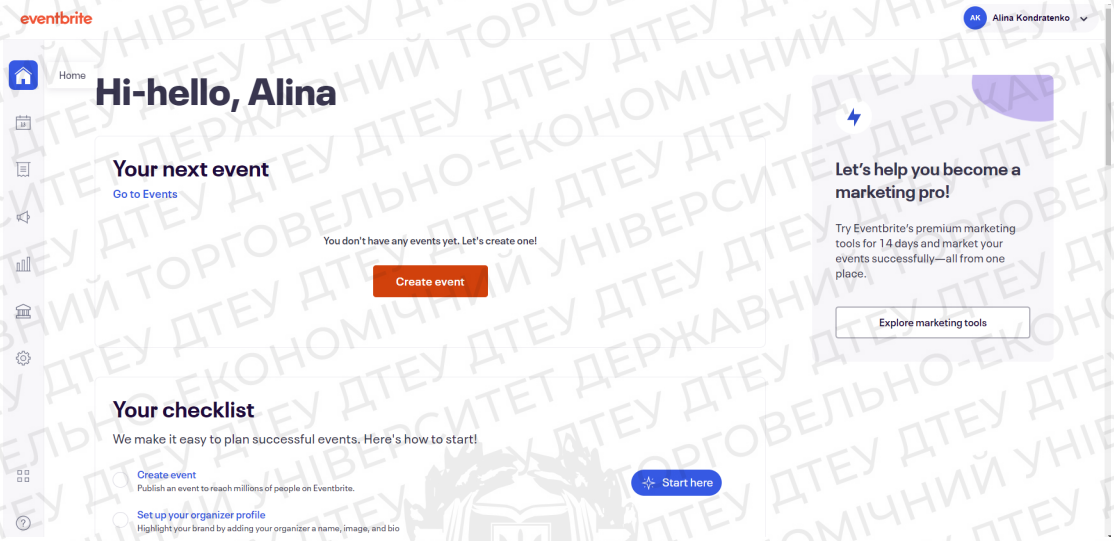

Рис. 1.10 Головна сторінка дошки Eventbrite

В панелі управління представлена функція для створення квитків(рис.1.11). Користувач(менеджер) може вказати доступну кількість білетів, тип білету, ім'я білету, ціну білету, термін дії і додаткові параметри: мінімальна/максимальна кількість білетів, формування eTicket(електронного квитка).

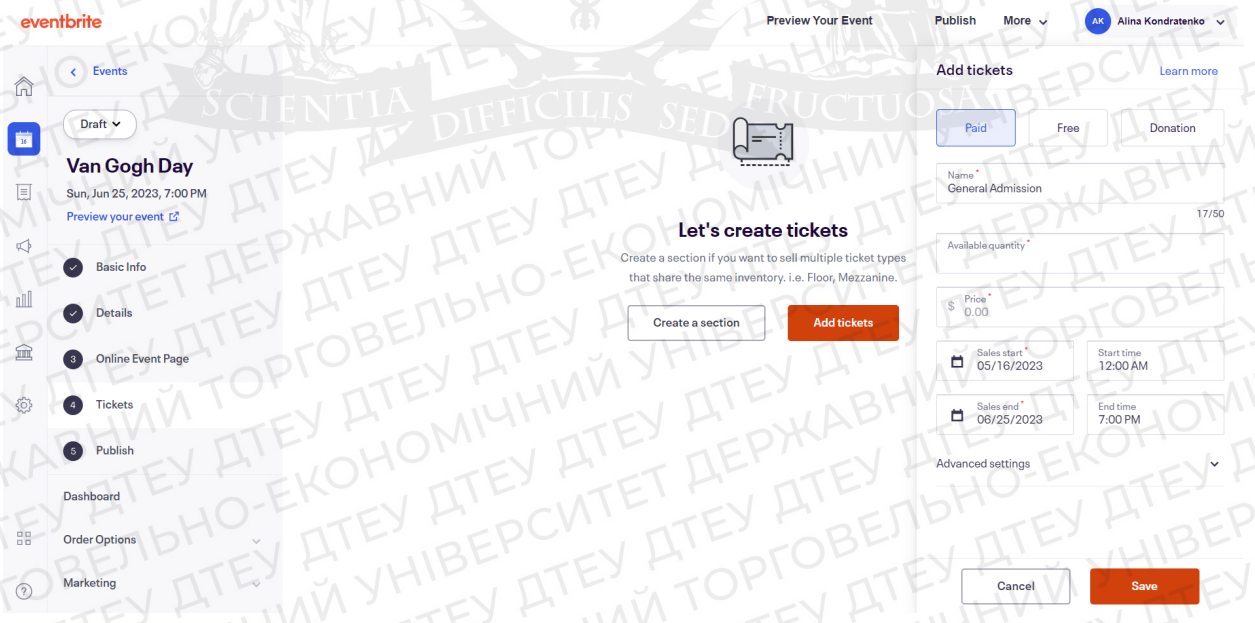

Рис. 1.11 Створення квитків для події Eventbrite

На дошці пропонується відредагувати параметри запиту на інформацію від користувача(рис.1.12), налаштувати, яку необхідну інформацію вводитиме користувач при реєстрації на виставку або при покупці білетів, можна обрати

зберігання інформації тільки про покупця або кожного відвідувача, окрім імені, фамілії та електронної пошти можна запросити: префікс(Містер або Місіс), стать, домашній телефон, мобільний телефон, адреса доставки білетів, веб сайт або блог, професія, компанія, робочий телефон і адреса.

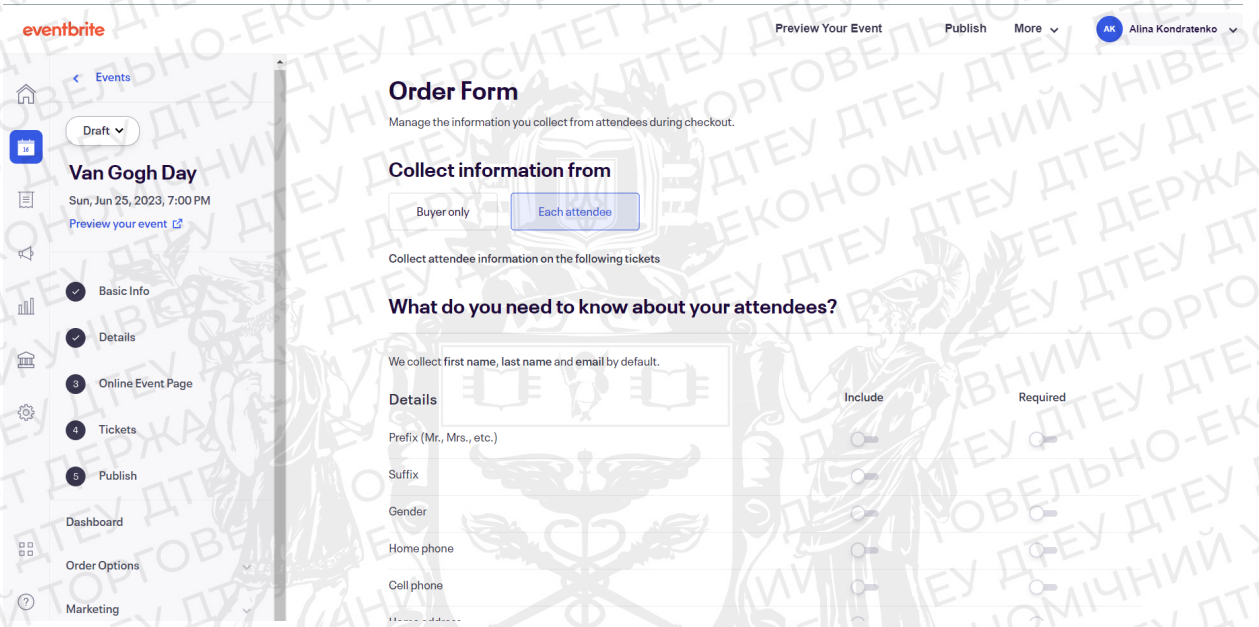

Рис. 1.12 Запит на дані про відвідувача в Eventbrite

Ця платформа має велику кількість функцій та дозволяє контролювати продажі білетів, створювати аналітику заходів, контролювати рекламну діяльність.

Переваги системи:

- Платформа є безкоштовною та доступною будь-кому;
- Платформа дає можливість створити сайт події.

Недоліками цієї інформаційної системи з точки зору організації виставок:

- відсутність контролю експонатів;
- відсутність контролю додаткових послуг;

- недостатня взаємодія з клієнтами, відсутність зворотного зв’язку з відвідувачами;
- відсутність переліку місць для проведення.

Посилання на web-застосунок: https://www.eventbrite.com/organizer/overview/

На основі розглянутих аналогів складено порівняльну таблицю функцій досліджуваних систем(табл.1.1).

Табл. 1.1 Порівняльна таблиця функцій аналогів

Функція	Eventzilla	Artsy	Eventbrite
Перегляд переліку місць проведення	-	+	-
Обирання типу виставки(online, offline, гібрид)	+	-	+
Реєстрація користувача в системі	+	+	+
Робота з квитками	+	-	+
Створення власних заходів	+	-	+
Продаж/покупка картин/мистецьких виробів	-	+	-
Контроль колекції картин для виставки	-	+	-
Реклама і просування виставки	-	+	+
Контроль бюджету	-	-	+

Продовження табл.1.1

Керування додатковими послугами	-	-	+
Управління учасниками і відвідувачами	+	-	+
Розрахунок вартості виставки	-	-	-
Керування відгуками	-	-	+
Перегляд авторів і картин	-	+	+

Отже, виділивши спільні ознаки систем-прикладів можна зробити висновок, що для конкурентоспроможності інформаційна система має обов'язково мати функцію реєстрації користувача, обирання типу виставки, управління учасниками і відвідувачами, робота з квитками також необхідно впровадити нові функції, наприклад, контроль бюджету, розрахунок вартості виставки, керування додатковими послугами, перегляд місць проведення.

1.3 Постановка задачі розробки інформаційної системи для організації виставок

Організація художніх виставок є комплексним процесом, який вміщує у собі велику кількість завдань, які мають бути детально розглянуті, а також потребують багато часу, контролю, ресурсів. Організацією виставки зазвичай займається менеджер митця, компанія або сам митець.

Метою проведення виставки часто є інформування суспільства про проблеми, поширення ідей, обмін досвідом, комерція, популяризація. Групи за інтересами, професійні товариства бізнесові товариства вважають, що

виставки є важливим місцем для комунікації. Художні виставки мають широкий вплив на культуру, суспільство, маркетинг, технології, вони реалізують функцію формування чи підтримки певного настрою у суспільстві[3].

Процес організації художньої виставки вміщує в собі значну кількість етапів:

1) Визначення теми та концепції виставки. На цьому етапі організатор виставки продумує тематику та концепцію виставки, хто є цільовою аудиторією.
2) Обрати місце проведення, яке підходить під розмір і тип картин, перевірка чи є потрібне обладнання для монтажу картин.
3) Обрати твори мистецтва, які прийматимуть участь у художній виставці. Перевірка творів на відповідність теми, розмір, кількість.
4) Розробити бюджет компанії. Планування і розрахунок витрат, до яких входить оренда приміщення, реклама, пошук спонсорів або грантів, монтаж та демонтаж художніх картин, організація транспортування картин та додаткові послуги: напої, екскурсоводи, кейтерінг.
5) Оформлення виставки. На цьому етапі вирішується як будуть розміщуватися картини, яким буде освітлення та інші деталі.
6) Регулювання реклами та просування виставки. Організація просування інформації в інтернеті, електронною поштою, сайт для інформування, бланки реклами, промо-відео, банери.
7) Організація проведення виставки. На цьому етапі підготовляють приміщення, встановлюють картини, планують день відкриття та додаткові заходи(аукціон, благодійні збори).
8) Закриття виставки. До цього процесу входить: організація демонтажу і повернення картин авторам, збір відгуків.

Існують етапи пов'язані з організацією виставки, які можна зробити спростити та підвищити ефективність, точність та контроль процесу за

допомогою застосування інформаційних технологій, що також вивільнить значну кількість часу на виконання задач з вищим пріоритетом.

Після аналізу виведено узагальнюючий перелік функцій, які можна автоматизувати за допомогою інформаційної системи:

- Керування створенням виставки(тема, місце проведення, концепція, напрямок, дати проведення);
- Керування виставкою, і розробка чек-листу для виставки;
- Керування переліком учасників(організаторів) виставки;
- Керування картинами(експонатами);
- Керування квитками;
- Керування бюджетом і розрахунок витрат;
- Керування маркетингом, рекламою і просуванням;
- Керування інформацією про відвідувачів;
- Керування додатковими послугами;
- Керування продажу картин;
- Керування запланованими заходами(аукціон) на виставці.

Враховуючи вищенаведене, основною цілю роботи є розробка інформаційної системи з підтримки процесу організації художніх виставок у вигляді web-застосунку. Подібна система дозволить художникам або їх менеджерам оптимізувати процес створення художньої виставки, дозволить контролювати картини, авторів, кількість витрат і прибутку, відвідувачів, а також надавати інформацію про компанію, галереї.

Основні можливості, що мають бути у розробленій інформаційній системі:

- Реєстрація користувача;
- Створення виставки;
- Чек лист для організації виставки;
- Перегляд доступних місць проведення;
- Розрахунок бюджету;
- Перегляд списку картин, авторів виставки.

Задля досягнення поставленої задачі потрібно виконати такі пункти:

- визначити основні вимоги до інформаційної системи з підтримки організації художніх виставок і провести їх аналіз;
- обрати методи розробки та технології для web-застосунка;
- розробити технічне завдання;
- спроектувати інформаційну систему;
- розробити прототип системи;
- спроектувати базу даних;
- розробити дизайн інтерфейсу користувача;
- розробити основні функції інформаційної системи.

РОЗДІЛ 2.
ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ(ІС) ДЛЯ ОРГАНІЗАЦІЇ ВИСТАВОК

Перед початком розробки інформаційної системи завжди заздалегідь потрібно побудувати її проект, щоб покращити розуміння галузі, проблеми та уникнути розбіжності думок з клієнтом, і покращити взаємодію з ним. Проектуванням інформаційної системи називають чіткий підхід, під час якого розробляється проект для розробки особливого програмного продукту. За допомогою цього підходу відбувається забезпечення виконання поставленого завдання, а також пошук вирішення конкретної проблеми[4].

Проектування інформаційної системи поділяється на три частини: проектування об’єктів даних, що будуть реалізовані у базу даних, проектування форм, програм, звітів, які будуть забезпечувати виконання запитів до даних, облік конкретного середовища або технологій, а саме використовуваної архітектури, конфігурації апаратних засобів[5]. Процес проектування повноцінної інформаційної системи має включати дослідження з обґрунтуванням необхідності розробки інформаційної системи, вибір концепції і цілей проекту, формування вимог до системи і підсистем, технічного завдання, розробка ескізу системи з усіма функціями, завданнями, побудова моделей бізнес-процесів і даних.

Початок роботи над проектом інформаційної системи полягає у визначенні її мети та завдань, необхідної функціональності, й надалі створення моделей, діаграм, формування вимог до архітектури, вибір програмного забезпечення та технологій.

2.1 Формування вимог до інформаційної системи для забезпечення організації виставок

Web-застосунок, який містить у собі інформаційну систему з підтримки процесу організації виставок призначений для інформування клієнтів організації про доступні місця для проведення, допомагати в оформленні детальної інформації про виставки, картини, розрахунку бюджету і організації додаткових заходів.

Система має бути загальнодоступною для будь-яких користувачів і у мережі Інтернет. Інформаційна система призначена в основному для менеджерів, малих організацій і митців. Для використання веб-застосунку користувач має мати комп'ютерний пристрій з клавіатурою та мишкою, а також доступ до Інтернет мережі.

На основі проведеного аналізу були сформульовані функціональні вимоги до інформаційної системи, яка розробляється.

Веб-застосунок має містити такі функціональні можливості як:

- реєстрація в системі;
- авторизація в системі;
- переглядання місць проведення;
- створення виставки
- розрахунок бюджету(інтерфейс);
- створення списку картин.

Структура web-застосунку складається з сайту та панелі управління. Початковий сайт має основні задачі показати інформацію про сервіс, розповісти про його можливості і переваги відвідувачу, а також дати можливість зареєструватися або зайти в систему з наявним обліковим записом, підписатися на розсилку з інформацією про найближчі художні виставки. Веб-додаток має містити просту та зрозумілу для користувача навігаційну панель, яка містить у собі функціонал для організації виставки.

Вимоги до дизайну для веб-застосунку:

- основний сайт, який бачить користувач до реєстрації має бути мінімалістичний та без зайвих деталей;

- основний сайт, легкий для завантаження, містить структуровану інформацію, яку легко читати;
- панель управління має простий дизайн.

Задля кращого розуміння та структурування інформації, конкретизації необхідних деталей про інформаційну систему варто розробити технічне завдання.

Технічне завдання являється важливим документом, що містить у собі цілі, вимоги і основні дані для початку, які необхідні для розробки автоматизованої системи управління[1,40]. Розробка технічного завдання допомагає вирішити проблему різноманітного розумінням задач, відмінностей думок щодо системи та визначити чіткий послідовність процесів. Цей документ містить у собі призначення системи, мету, встановлює загальні вимоги до web-застосунку, висуває функціональні та нефункціональні вимоги, детально прописує етапи розробки, структуру додатку.

На основі попередніх досліджень було повністю сформульоване технічне завдання в Додатку А.

2.2 Розробка проекту інформаційної системи для підтримки процесу організації виставок

IDEF0 - набір методів, правил і процедур, призначених для створення функціональної моделі системи. Метою цієї методології є створення ієрархічної системи. Спочатку необхідно зробити загальний опис системи і визначити її взаємодію з навколишнім середовищем (контекстна діаграма).

Потрібно детально розглянути з чого складається художня виставка як бізнес процес, для кращого розуміння, який функціонал має бути пріоритетнішим і які процеси буде доречно автоматизувати, для цього створена структурно-функціональна модель художньої виставки за допомогою програмного забезпечення Erwin.

Контекстна діаграма відображає процеси оформлення виставки, організаційні процеси, процеси управління, реклами і проведення художньої виставки. Для її побудови були визначені вхідні дані, вихідні дані, механізми та елементи управління.

Контекстну діаграму на якій відображено бізнес-процес діяльності художньої виставки, зображено на рис.2.1.

До вхідних даних контекстної діаграми належить:

- Дані про білети;
- Запит на організацію художної виставки;
- Дані про учасниіків;
- Фінанси;
- Місця проведення виставки;
- Художні роботи;
- Інформація про захід та авторів.

До елементів управління контекстної діаграми належать:

- Правила проведення виставки;
- Правила організації виставки;
- Правила системи;
- Стандарти монтування експозицій.

До механізмів контекстної діаграми належать:

- Інформаційна система;
- Організатор;
- Технічний персонал.

До вихідних даних контекстної діаграми належить:

- Дані про заплановану виставку;
- Дані про продані білети;
- Список картин;
- Розрахований бюджет виставки;
- Створений чек лист організації виставки;
- Дані про зареєстрованих відвідувачів.

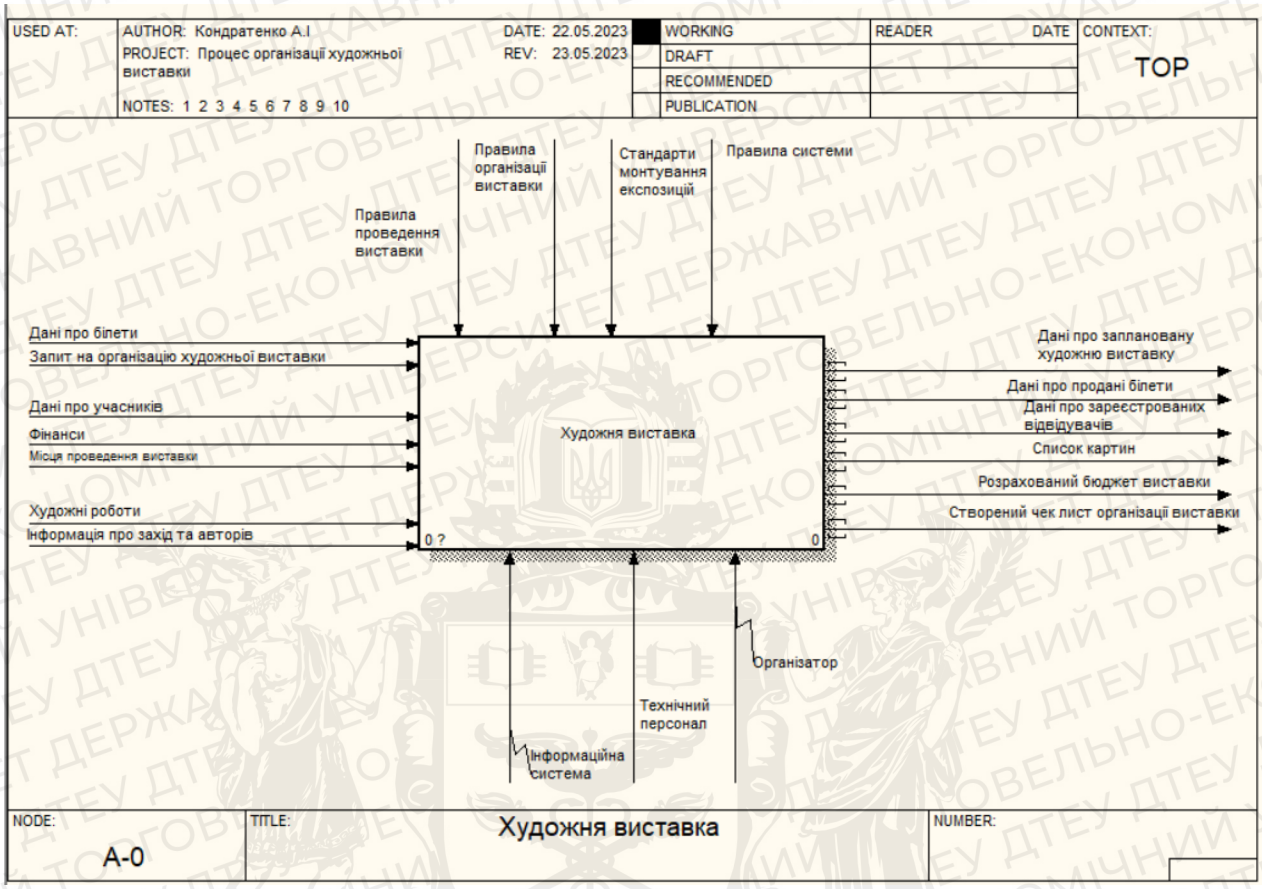

Рис. 2.1 Модель бізнес-процесу художньої виставки

Після того як контекстна діаграма побудована необхідно провести її функціональну декомпозицію. Для декомпозиції процес необхідно розділити на підпроцеси, кожен з яких має бути описаний відокремлено від інших (діаграма декомпозиції). Створена декомпозиція моделі художньої виставки зображена на рис.2.2.

Модель розділена на такі підпроцеси: оформлення виставки, організація процесу виставки, розрахунок бюджету виставки, контроль відвідувачів, проведення виставки та аналіз. Після завершення останнього підпроцесу рахується, що виставка була проведена і створюється звітність про виставку, яка складається з відгуків відвідувачі, фінансового звіту, аналізу ефективності рекламної діяльності.

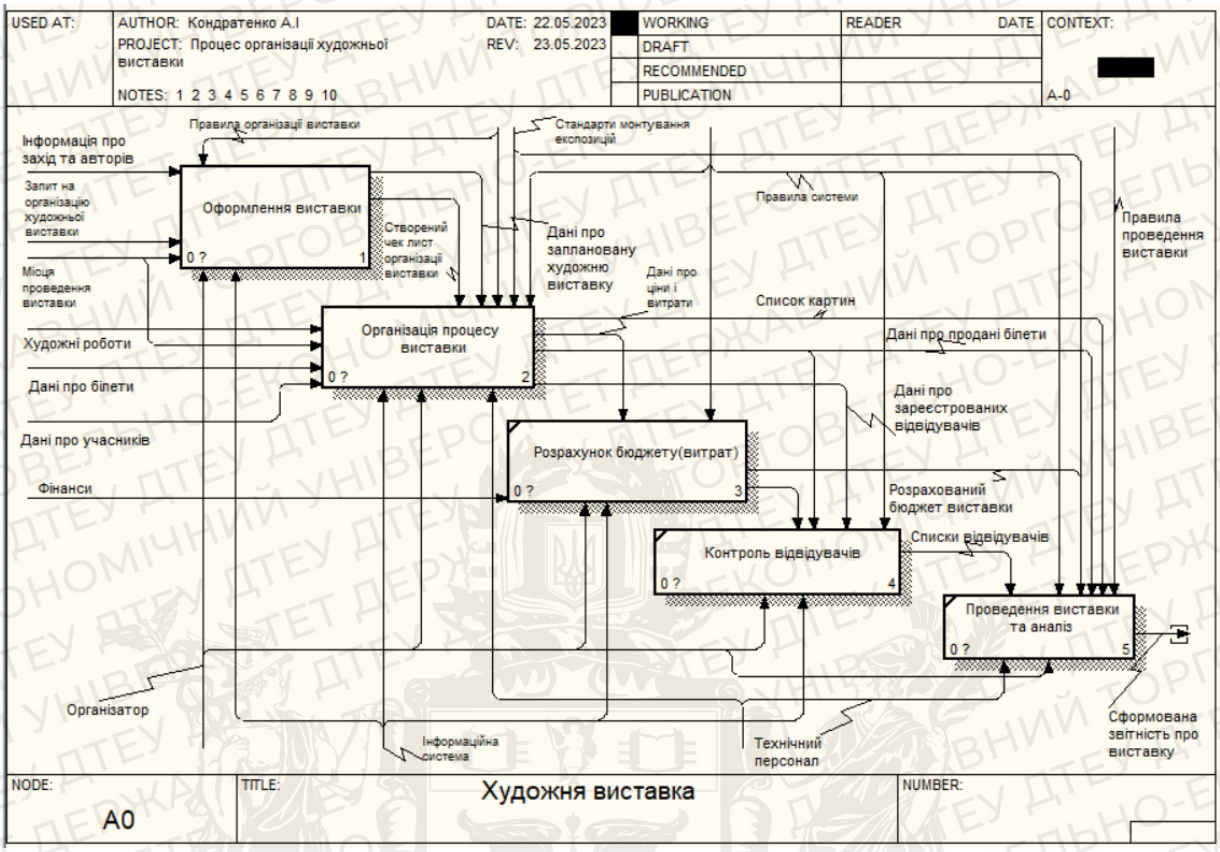

Рис. 2.2 Модель декомпозиції бізнес-процесу художня виставка

Після аналізу бізнес-процесів організації художної виставки була проведена декомпозиція двох з підпроцесів, а саме процесу «Оформлення виставки»(рис.2.3) та процесу «Організації процесу виставки» (рис.2.4).

Декомпозиція підпроцесу оформлення виставки розділились на чотири процеси:

- Вибір тематики, опис концепції, назви;
- Вибір авторів і робіт;
- Вибір місця проведення художньої виставки;
- Вибір дати проведення і етапи виставки.

В наслідку цих процесів має бути отримано всі необхідні основні дані про виставку, що вподальшому будуть використовуватися для організації та рекламної діяльності і створений чек лист процесів організації.

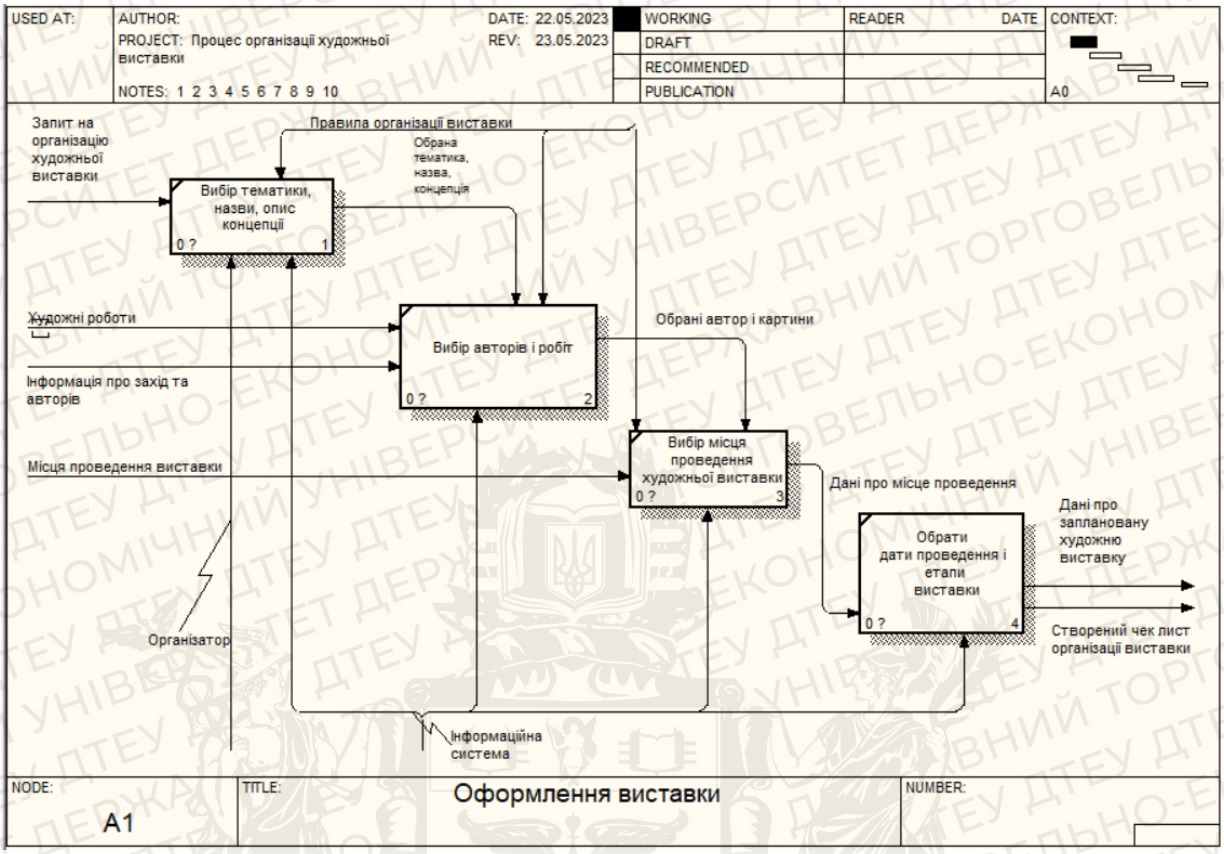

Рис. 2.3 Модель декомпозиція процесу «Оформлення виставки»

Процес «Організація виставки» після декомпозиції діаграми був поділений на підпроцеси:

- Створення списку художніх робіт та учасників виставки;
- Організація додаткових та технічних послуг(технічні прослуги – виставлення композиції, монтаж робіт, транспортування робіт, додаткові послуги - гіди, аукціон, кейтерінг);
- Створення продажу білетів, характеристик та типів білетів, відкриття реєстрації користувачів;
- Реклама та просування;
- Зіставлення переліку усіх витрат на організацію та рекламу.

В наслідку всіх процесів ми отримаємо дані про всі білети, ціни, витрати, перелік послуг і пройдемо чек лист організаційних процесів.

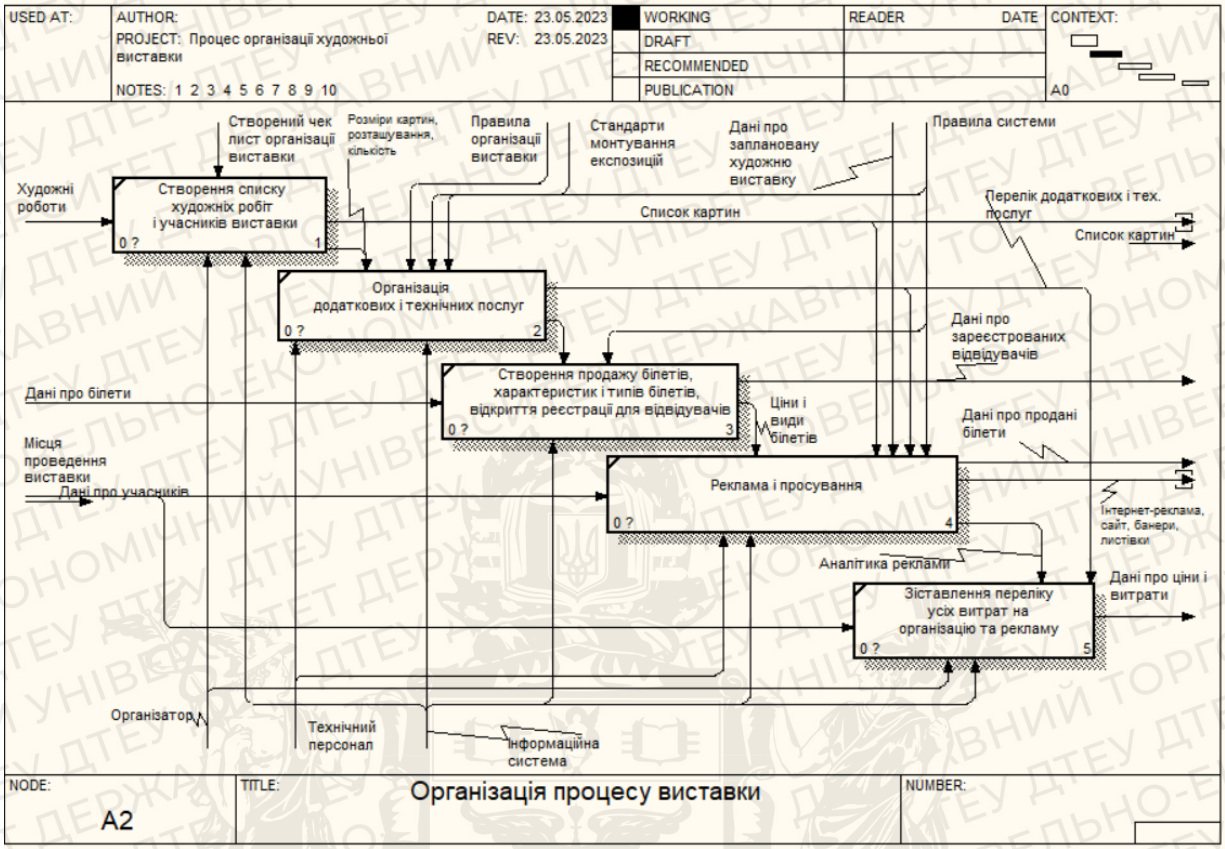

Рис. 2.4 Модель декомпозиція процесу «Організація процесу виставки»

Мова UML(Unified Modeling Language) являється загальною мовою графічного моделювання, що використовується для створення представлення проекту інформаційної системи у вигляді візуальної моделі, що дає повне уявлення про функціональні можливості і конфігурацію. Головною ідеєю UML є можливість моделювати будь-яке програмне забезпечення або інші системи як набори взаємодіючих об’єктів[1,76].

UseCase UML діаграму або діаграму використання застосовують для відображення функціоналу в загальному вигляді[1]. Для створення діаграми використовуються 2 типи елементів: функцій описують прецеденти(варіанти використання), користувачів, пристрої, бази даних описують як дійових осіб(акторів). Визначити загальні границі, конкретизувати предметну область, сформувати вимоги до функціоналу майбутньої інформаційної системи, підготувати документацію на початок, що відображає взаємодію творців

системи з користувачами і клієнтами – це все є метою створення діаграм використання.

Для розробки UseCase діаграми потрібно для початку визначити акторів та прецеденти інформаційної системи. Розроблено таблицю з усіма дійовими особами та варіантами використання(табл.2.1).

Дійовими особами системи (акторами) інформаційної системи обрано:

- Організатор – користувач, який займається організацією художньої виставки, і який є основним організатором виставки. (Митець, менеджер художника)
- Організація – виступає в ролі користувача з розширеними можливостями для організації масштабніших виставок.
- Адміністратор – особа, яка займається вирішенням питань з роботи системи.
- Користувач – особа, яка не зареєстрована в системі і просто переглядає сайт.
- База даних(БД) – місце, де зберігаються усі дані системи.

Табл.2.1 Списки елементів UseCase діаграми інформаційної системи

Список акторів	Список прецедентів
Користувач Організатор Організація Адміністратор БД	Реєстрація Вхід в систему Створити виставку Створити чек лист Розрахунок бюджету Створити список картин Вибір типу виставки Вписування характеристик виставок Додавання пункту у чек лист

Для представлення функціоналу інформаційної системи та взаємодії з ним користувачів створена UseCase діаграма(рис.2.5).

Дійові особи мають перед реєстрацією на головному сайті з інформацією або авторизацією і входом у систему мають характеристики звичайного користувача, тобто наслідують його.

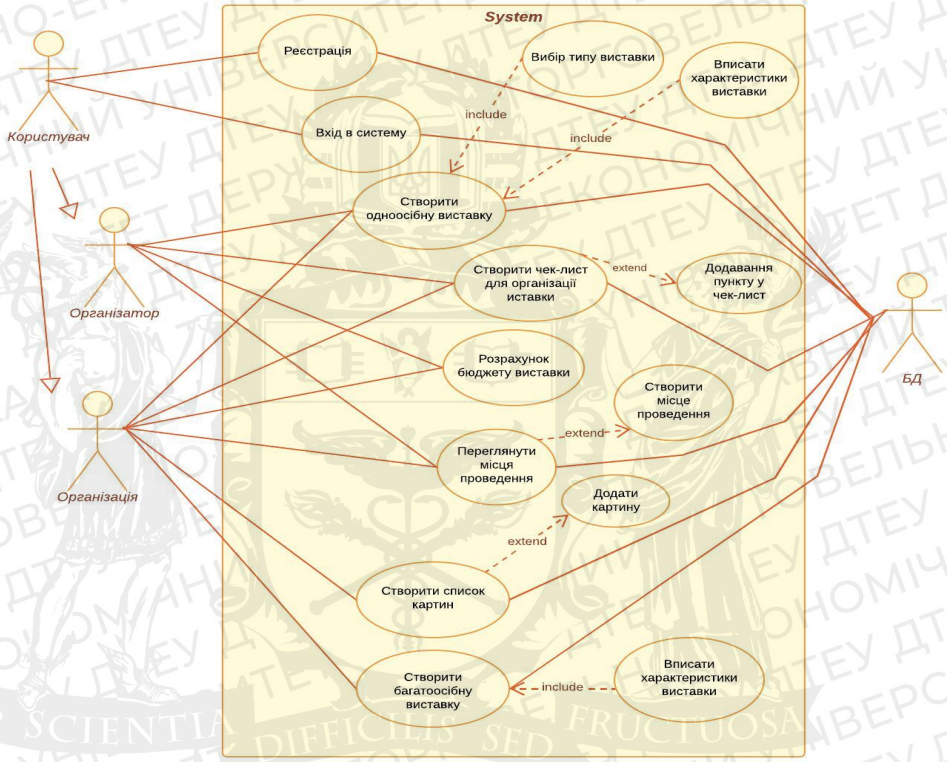

Рис. 2.5 Діаграма використання для інформаційної системи

Для опису прецедентів високого рівня було розроблено таблиці з такими параметрами(табл.2.2-2.5): ім’я прецеденту, дійові особи, номер, опис прецеденту, головний сценарій та інший сценарій.

Табл. 2.2 Опис прецеденту високого рівня «Вхід в систему»

Ім’я прецеденту	Вхід в систему
Дійові особи	Користувач БД

Продовження табл.2.2

Номер	1
Опис прецеденту	Після кліку по кнопці «Вхід», відкривається форма для входу у систему. Користувач вводить електронну пошту і пароль для того, щоб зайти до системи.
Головний сценарій	Дані про користувача знайдені. Вхід в систему реалізовано.
Інший сценарій	Даних користувача немає в БД. Операція не виконана, вхід до системи не здійснений.

Табл.2.3 Опис прецеденту високого рівня «Створити виставку»

Ім’я прецеденту	Створити виставку
Дійові особи	Організатор або Організація БД
Номер	2
Опис прецеденту	Користувач певної ролі переходить у розділ «Виставки», натискає на кнопку «Створити виставку» для того, щоб створити запис з даними про художню виставку.
Головний сценарій	Користувач створив запис про художню виставку.
Інший сценарій	Введено недостатньо даних, або дані не є коректними, операція не виконана.

Табл.2.4 Опис прецеденту високого рівня «Створити список картин»

Ім’я прецеденту	Створити список картин
Дійові особи	Організатор або Організація БД
Номер	3
Опис прецеденту	Користувач переходить у розділ «Картини» », натискає на кнопку «Створити список» для того, щоб створити дошку на яку можна додавати картини з даними, для цього варто ввести ім’я списку, назва виставки.
Головний сценарій	Список картин для певної виставки створений.
Інший сценарій	Введені неправильні дані або неналежна інформація, запит не реалізовано.

Табл. 2.5 Опис прецеденту високого рівня «Переглянути місця проведення»

Ім’я прецеденту	Переглянути місця проведення
Дійові особи	Організатор або Організація БД
Номер	3
Опис прецеденту	Користувач переходить у розділ «Місця проведення», і переглядає перелік місць проведення, і їх характеристики.
Головний сценарій	Користувач певної ролі переглянув список місць проведення та обрав певне для своєї виставки.
Інший сценарій	Помилка системи або користувач не знайшов місця яке підходить для його виставки.

Для представлення прецеденту «Вхід у систему» та прецеденту «Створення виставки» були розроблені діаграми послідовності(sequence diagrams).

Діаграми послідовності являються способом відображення поведінки, за допомогою зображення повідомлень, що передаються у системі почергово.

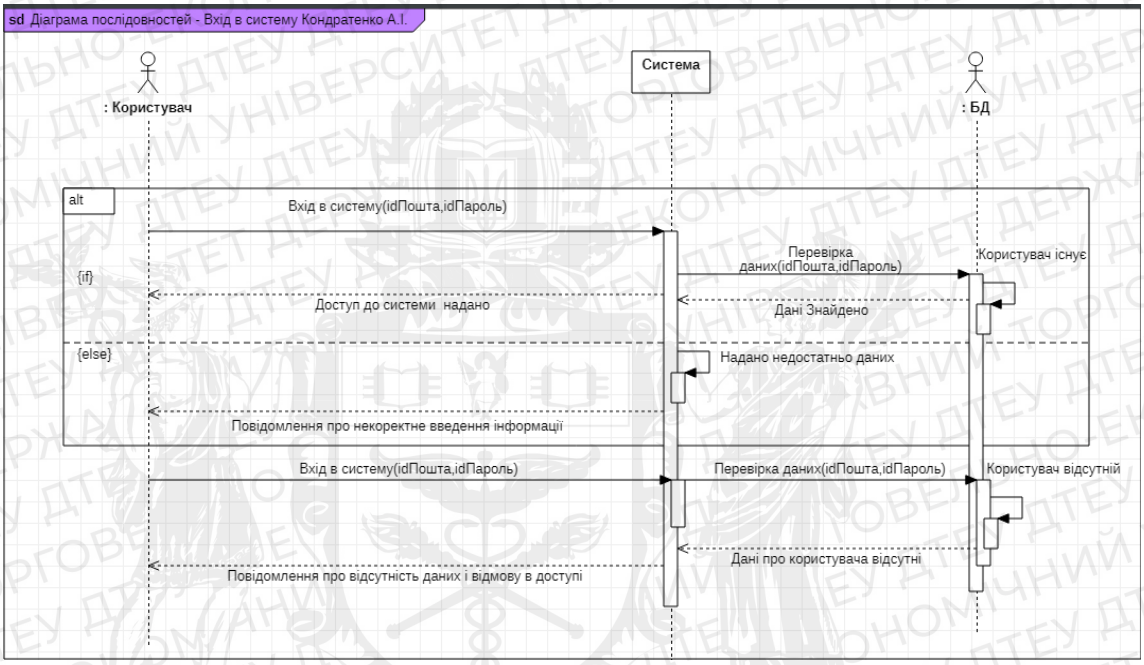

Рис. 2.6 Діаграма послідовності для прецеденту «Вхід в систему»

Алгоритми дій користувача у варіанті дії «Вхід в систему» (рис.2.6) в залежності від правильності вводу даних входу:

1. Якщо користувач хоче зайти в систему і вводить дані(пошта, пароль), відправляє їх в систему. Система перевіряє дані за рахунок відправлення запиту в базу даних. База даних знаходить потрібні дані та відправляє відповідь системі, а система надає права доступу.
2. Якщо користувач хоче зайти в систему і вводить дані(пошта, пароль), але недотримується правил, та вводить недостатню кількість символів, або не правильно оформлену електронну пошту, відправляє їх в систему. Система перевіряє дані з стандартом, відправляє користувачу повідомлення про те, що дані введені не правильно.

3. Якщо користувач хоче зайти в систему і вводить дані(пошта, пароль), відправляє їх в систему. Система перевіряє дані за рахунок відправлення запиту в базу даних. База даних не знаходить даних користувача та відправляє негативну відповідь системі, а система відправляє повідомлення про відсутність введених даних у системі і відмову у доступі.

Для представлення процесу «Створення виставки» була розроблена діаграма послідовності (рис.2.7), яка має такий алгоритм:

1) Користувач натискає кнопку «Створити виставку». Система відправляє користувачу запит на дані: назва виставки, опис виставки, організатор.
2) Користувач вводить в потрібні поля усі потрібні дані і натискає кнопку «Далі», система відправляє дані на збереження у базу даних. База даних видає системі відповідь про успішність операції. Система відправляє запит на введення типу виставки.
3) Користувач обирає тип виставки з трьох варіантів(онлайн, офлайн, гібрид) та натискає далі. Система відправляє отримані дані на збереження, база даних надсилає відповідь про успішне збереження. Система відсилає запит користувачу на дані про місце проведення, дати проведення.
4) Користувач вводить місце проведення, дати проведення і натискає кнопку «Створити». Система відправляє отримані дані в базу даних, база даних надсилає відповідь про успішне збереження. Система показує створений новий запис про художню виставку у таблиці.

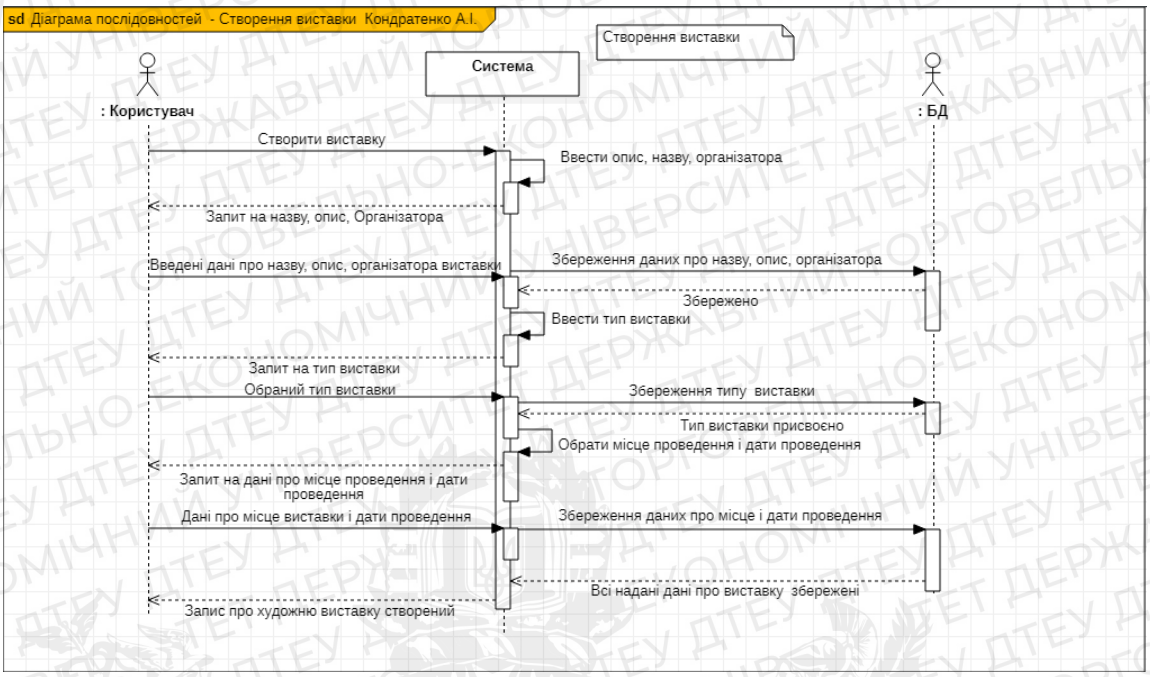

Рис. 2.7 Діаграма послідовності для прецеденту «Створення виставки»

2.3 Архітектура web-застосунків

Архітектура веб-застосунків являє собою схему, що відображає взаємозв'язки між складовими системи: база даних, користувацькі інтерфейси, веб сервер, API.

Архітектура веб-застосунків поділяється на рівні(рис.2.8):

- Рівень презентації або представлення. Цей рівень використовується задля відображення клієнтського інтерфейсу, містить у собі усі UI(User Interface) елементи, які співпрацюють з серверним рівнем, доступний у браузері, обробляє і надає усі необхідні дані для сторони користувача. Метою рівня є обробка запитів користувачів, їх відправлення, надання відповідей, результатів, а також відображення їх у застосунку.
- Рівень бізнес логіки. Цей рівень є відповідальним за обмін даними, бізнес-логіку операцій, правила, має в собі інструкції для взаємодії

серверної частини веб-застосунку з клієнтською, як обробляти запити клієнтів та дані[20].

- Рівень служб обслуговування даних (також відомий як DSL) передає дані, оброблені рівнем бізнес-логіки, на рівень презентації. Цей рівень забезпечує безпеку даних шляхом ізоляції бізнес-логіки від клієнтів.
- Рівень доступу до даних(DAL), керує CRUD-операціями(створення, читання, видалення, оновлення), а також пропонує полегшений доступ до баз з XML-файлами.

Рис. 2.8 Рівні архітектури веб-застосунків

Web-застосунки поділяються на три типи за архітектурою:

– SPA (Single Page Application) – веб-додатки для бізнесу, що завантажують одну HTML-сторінку, яка оновлюється автоматично за допомогою JS. SPA використовує API(Application Programming Interface) для зв'язку з сервером, в разі необхідності можна використовувати таку саму API у мобільному застосунку[17]. Цей тип архітектури добре підходить для створення інтерактивного веб-застосунку, забезпечує швидку візуалізацію після повного завантаження програми у браузері. Основні недоліки SPA полягають в тому, що для першого завантаження застосунку потрібен тривалий час, також цей тип

є дорогим, має обмежену підтримку застарілих браузерів і погану маршрутизацію. SPA архітектуру застосовують Facebook, Gmail, GitHub, Twitter;

– MPA (Multi Page Application) – це багатосторінкові веб-додатки, що працює при постійній взаємодії з веб-сервером. MPA є більш складними, дорогими масштабними та потребують більше часу на розробку, аніж SPA. Кожна сторінка MPA відображає різний вміст, який потрібно оновлювати щоразу, коли користувач взаємодіє з нею. Особливо часто MPA використовується для сайтів з великим об'ємом даних, наприклад, застосунки для новин, сайти для комерції. MPA використовують Amazon, The New York Times та eBay[18].
– PWA (Progressive Web App) – це гібрид web-сторінки та додатка[]. Прогресивні веб-додатки підтримуються Windows, Android та iOS (для iOS офлайн-режим вимкнено). PWA є безпечним, оскільки використовує HTTPS. Розробники можуть додавати оновлення до веб-програми віддалено. Серед недоліків цього типу архітектури – необхідність вибору браузера та ОС, які його повністю підтримують[17].

Архітектура застосунків може ділитися на види:

- Монолітна архітектура(Складається з бази даних, клієнтської та серверної сторони);
- Архітектура мікросервісів (Складається з набору малих сервісів, які поєднуються утворюючи застосунок);

У веб-застосунку, що містить інформаційну систему для підтримки процесу організації художніх виставок застосовується клієнт-сервер архітектура і веб-додаток буде типу SPA з монолітною архітектурою.

2.4 Технології для розробки клієнтської складової

Дизайн процес – це підхід, який використовують дизайнери веб-застосунків і додатків, в його основі лежить розбиття великого проекту на

менші керовані частини. Цей підхід активно використовують також науковці, інженери.

Дизайн процес поділяється на етапи:

- Визначення проблеми;
- Збір інформації;
- Мозковий штурм та аналіз ідей;
- Розробка рішення;
- Збір відгуків;
- Вдосконалення.

Процес розробки клієнтської частини веб-застосунку складається з:

1. Перегляд вимог. Цей етап віщує в собі аналіз вимог проекту, його функціональності, цільової аудиторії.
2. Розробка користувацького інтерфейсу (UI) та формування взаємодії з користувачем(UX). Етап передбачає дослідження взаємодії користувача та інтерфейсу, створення дизайну та макету клієнтського інтерфейсу.
3. Верстка інтерфейсу за допомогою HTML та CSS.
4. Програмування функціоналу за допомогою JavaScript.
5. Оптимізація та тестування.

Для розробки дизайну клієнтського інтерфейсу використовують Figma.

2.4.1 Figma

Figma – це сервіс у хмарі, що має вигляд графічного редактора, з яким можна працювати прямо у браузері, призначений для UI/UX дизайнерів та веб-розробників, працює на багатьох платформах, має онлайн та десктопну (комп'ютерну версію).

Призначенням цього сервісу є:

- розробка прототипів для додатків і веб-сайтів;
- розробка елементів інтерфейсу, таких як іконки, форми, кнопки;
- розробка векторних зображень.

Figma дозволяє багатьом користувачам редагувати один файл і залишати коментарі, має власне хмарне сховище з проектами, є продуктивним і добре підходить для реалізації прототипування проекту, а тож для створення user story map.

Основними інструментами та елементами за рахунок яких створюється макет у Figma є:

- фрейми(артборди, на яких створюється проект);
- зображення;
- векторні форми;
- текст;
- модульна сітка(використовується для упорядкування елементів на фреймі);
- криві(лінії для створення простих векторних форм);
- ефекти і маски.

Сервіс дозволяє розширювати список доступних функцій за рахунок плагінів. Існує велике різноманіття плагінів в залежності від необхідного функціоналу. Наприклад, існує плагін Unsplash, задача якого імпорт зображень веб-застосунку, який слугує складом картинок.

2.4.2 HTML

Мова розмітки гіпертексту або HTML(HyperText Markup Language) - це поширена мова розмітки, що застосовується для створення веб-сторінок. Використовуючи компоненти HTML, такі як теги та атрибути, вона дозволяє розробляти та структурувати розділи, абзаци, посилання та елементи.

HTML використовується в розробці веб-сайтів, для відображення різноманітних елементів на веб-сторінці. Мову розмітки гіпертексту також для забезпечення навігації між сторінками за рахунок вбудовування гіперпосилань. HTML, як Microsoft Word, дозволяє організовувати і форматувати документи, особливо важливу документацію. HTML

використовує теги (символи, які знаходяться всередині кутових дужках), щоб надати інформації, особливого значення інформації, яку вони оточують[12].

Кожна HTML сторінка обов'язково містить такі структурні теги:

- <html> є кореневим елементом, який визначає весь HTML-документ;
- <head> містить мета дані, такі як заголовки, кодування;
- <body> тег, у якому заходиться весь вміст сторінки.

Елементи в HTML поділяються на:

- Блочні елементи(block) – елементи, які починаються з нового рядка(, <section>, <li>, <table>);

- Вбудовані елементи(inline) – елементи, які займають стільки місця скільки буде потрібно(<a>,,
,<button>).

HTML5 – сучасна версія HTML, яка вміщує в собі більше нових функцій, тегів, представляє кілька елементів для покращення інтерактивності, мультимедійних можливостей і семантичної ефективності. Медіа можна розміщувати в коді HTML замість використання плагінів. Елементи HTML5 включають:

- семантичні елементи(<header>, <footer>, <nav>);
- графічні елементи(<canvas>, <svg>);
- мультимедійні елементи(<video>, <audio>).

2.4.4 CSS

CSS - це мова каскадних стилів, яку використовують для стилізації веб-сайту, а саме для додавання фону, шрифтів, кольорів, анімацій, інтерактивних елементів.

CSS дотримується правил стилів, що вміщає собі стилізацію за допомогою селекторів, властивостей і значень. Браузер інтерпретує файл CSS, що розроблений за правилами стилів, і закодовані стилі застосовуються до елементів сайту. CSS може використовуватися: як окремий файл, що підключається до HTML файлу з елементами за допомогою тегу <link>, як

стиль у рядку, який застосовується тільки для конкретного рядка за допомогою тегу <style>, як внутрішній стиль сторінки, який має завантажуватися кожного разу як сторінка оновлюється.

Селектори CSS застосовуються для конкретизації компонентів сторінки, до яких згодом будуть додаватися певні характеристики стилів. Селектори бувають:

1. Класові селектори(стилізують елементи, за певним класом);
2. Id-селектори (стилізують елементи, за унікальним ідентифікатором);
3. Селектор за елементом(стилізує HTML елементи на сторінці за їх тегом);
4. Універсальний селектор(стиль застосовується до усієї сторінки);
5. Груповий селектор(стилізує одразу декілька типів елементів за тегами).

CSS підтримує успадкування, тобто коли властивості, застосовані до батьківського елемента, можуть передаватися і застосовуватись до його дочірніх елементів, що дозволяє ефективно та узгоджено застосовувати стилі до різних елементів. Крім того, CSS дотримується каскадного порядку, що означає, що стилі можна комбінувати у залежності від їх порядку застосування або специфіки застосування.

2.4.5 React.js

React - це популярна бібліотека, яка використовується для створення користувацьких інтерфейсів. React.js був розроблений компанією Facebook, щоб вирішити деякі проблеми, пов'язані з масштабними веб-сайтами і керованими даними[11]. За допомогою фреймворку React можна розробляти свої додатки, створюючи багаторазові компоненти, які можна порівняти з блоками Lego. Компоненти React.js є повноцінними окремими частинами кінцевого інтерфейсу, які, будучи зібраними до купи, формують весь користувацький інтерфейс застосунку.

Головна роль і задача React у веб-застосунку в обробці рівня представлення цього додатку. Фреймворк React.js поєднує швидкість та ефективність JavaScript з більш ефективним методом маніпулювання DOM для швидшого рендерингу веб-сторінок та створення динамічних та адаптивних веб-додатків.

React.js використовує віртуальний DOM (Document Object Model), який є абстракцією реального DOM, React оптимізує продуктивність, завдяки віртуальному DOM оновлюючи лише необхідні зараз частини інтерфейсу користувача, коли відбуваються зміни, замість того, щоб змінювати всю сторінку. І така стратегія призводить до більш плавного використання застосунку, підвищення продуктивності веб-продукту та швидшої обробки.

React має велику кількість різноманітних бібліотек, інструментів. Існують бібліотеки додаткові бібліотеки такі як: Redux – для управління глобальним станом, Axios - для створення HTTP-запитів, а також бувають окремі бібліотеки компонентів для інтерфейсу користувача, наприклад Material-UI, Ant Design, Chakra UI, які заздалегідь надають спроектовані та кастомізовані компоненти. React використовують у багатьох веб-застосунках, наприклад Khan Academy, Instagram, Netflix.

2.4.6 Chakra UI

Chakra UI - це сучасна бібліотека компонентів для React.js. Бібліотека надається користувачам з численним набором багаторазових та комбінованих React-компонентів, які можна використовувати для створення додатків з клієнтським інтерфейсом. ЇЇ основні переваги в модульності, простоті застосування та доступності. Chakra UI використовується для створення React-застосунків та для прискорення процесу розробки веб-продукту.

2.6.7 Tailwind CSS

Бібліотека Tailwind CSS має відкритий код та пропонує колекцію класів, яка додається напряму до рядка з елементами HTML і стилізує їх. Службові класи цієї бібліотеки прискорюють роботу та спрощують її, дозволяють створювати багаторазові компоненти для веб-сторінок.

2.5 Технології для розробки серверної складової.

Після ретельного огляду доступних технологій, для реалізації бази даних було обрано MySQL, для відображення бази даних ORM Prisma.

2.5.1 Next.js

Next.js - фреймворк, заснований на React, який дозволяє створювати веб-додатки з покращеною зручністю для користувача та підвищеною продуктивністю завдяки додаванню функцій попередньої обробки, таких як статична генерація сторінок (SSG) і повний рендеринг на стороні сервера (SSR). Next.js має маршрутизацію на основі файлів, що означає, що кожен файл у каталозі представляє певний маршрут у програмі. Next.js забезпечує легкий спосіб створення кінцевих точок API, а також містить вбудовану підтримку бібліотек CSS-in-JS, вбудовані можливості для оптимізації зображень.

2.5.2 База даних MySQL

SQL є стандартизованою мовою для доступу до баз даних. MySQL — це популярна система керування базами даних, що дозволяє керувати реляційними базами даних і являється програмним забезпеченням з відкритим кодом, яке підтримується Oracle. У порівнянні з іншим програмним забезпеченням для баз даних, таким як Oracle Database або Microsoft SQL Server, MySQL досить легка система для вивчення. Реляційні бази даних на момент написання дослідження активно використовуються для представлення складних взаємозв’язків між елементами даних і обчислення їх з швидкістю, необхідною для прийняття ефективних та продуктивних рішень у сучасних організаціях[23].

MySQL може працювати на різних платформах, таких як UNIX, Linux і Windows. MySQL підтримує методи, що гарантують безпеку даних і їх

цілісність : автентифікація користувачів, контроль доступу та механізми шифрування. В MySQL існують спеціальні системи привілеїв, що забезпечує регулювання прав доступу облікових записів.

2.5.3 ORM(Object Relational Mapping) Prisma

ORM(Object Relational Mapping) являє собою спеціальну технологію програмування, призначену для створення віртуальної бази даних, що має на меті пов’язати бази даних з принципами об’єктно-орієнтованого програмування.

Prisma - це інструмент об'єктно-реляційного відображення (ORM) для Node.js та TypeScript. Prisma пропонує більш комфортний підхід до управління базами даних й усуває необхідність написання SQL запитів вручну. Цей інструмент є безпечним і легко інтегрується в набір технологій для моделювання баз даних, міграцій та зв'язку між додатками та базами даних.

Prisma у веб-технологіях застосовується для:

- Створення запитів до баз даних за допомогою конструктора;
- Визначення моделі даних застосунку, за допомогою вмонтованої мови опису схем, що допомагає визначати типи даних, обмеження, зв’язки;
- Спрощення основних операцій з базами даних за допомогою API та методів, наприклад, читання, видалення даних, оновлення;
- Спрощення роботи з зв’язками даних.

РОЗДІЛ 3.
РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ З ПІДТРИМКИ ОРГАНІЗАЦІЇ ХУДОЖНІХ ВИСТАВОК У ВИГЛЯДІ WEB-ЗАСТОСУНКУ

3.1 Розробка інтерфейсу та дизайну

Для розробки дизайну інтерфейсу необхідно було розробити user story map(Карта користувача), що мала слугувати відображенням послідовності та варіацій дій користувача на сайті та в системі. Для інформаційної системи з підтримки процесу організації художніх виставок у програмі Miro було створено User Story Map(рис.3.1), де користувачем може бути організація або організатор(митець, менеджер), карта показує активності або діяльності(червоні картки), які користувач може виконувати у веб-застосунку, дії(рожеві картки) і завдання нижчого рівня(жовті картки).

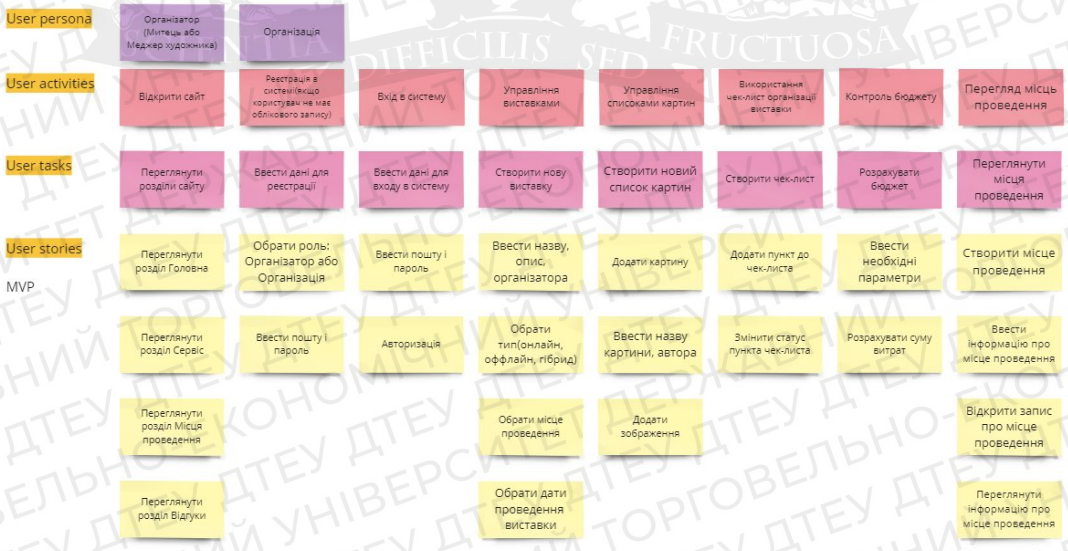

Рис. 3.1 User Story Map

В Adobe illustrator був розроблений брендінг сервісу “Exhibit.Mastery”(рис.3.2), який складається з: особливого логотипу та палітри

кольорів, які в подальшому застосовуються в макетах сайту та панелі навігації, підібраних шрифтів.

Рис. 3.2 Логотип “Exhibit.Mastery” і обрана палітра кольорів

Задля початкового схематичного представлення структури веб-застосунку, було розроблено wireframe або прототип дизайну, на якому зображено положення елементів, тексту, без додавання стилів. Wireframe в розробці інтерфейсу користувача слугує каркасом для майбутнього повноцінного макету програмного продукту. Прототип відображає основні сторінки і розділи головного сайту та панелі управління інформаційною системою, а також форми для реєстрації, входу.

Зображення прототипу сайту та основних розділів розміщено в Додатку Б.

Для панелі управління або дешборду був розроблений прототип, який зображає положення компонентів дошки, необхідні текстові поля та поля вводу, таблиці. На рисунку 3.2 зображений wireframe головної сторінки панелі управління.

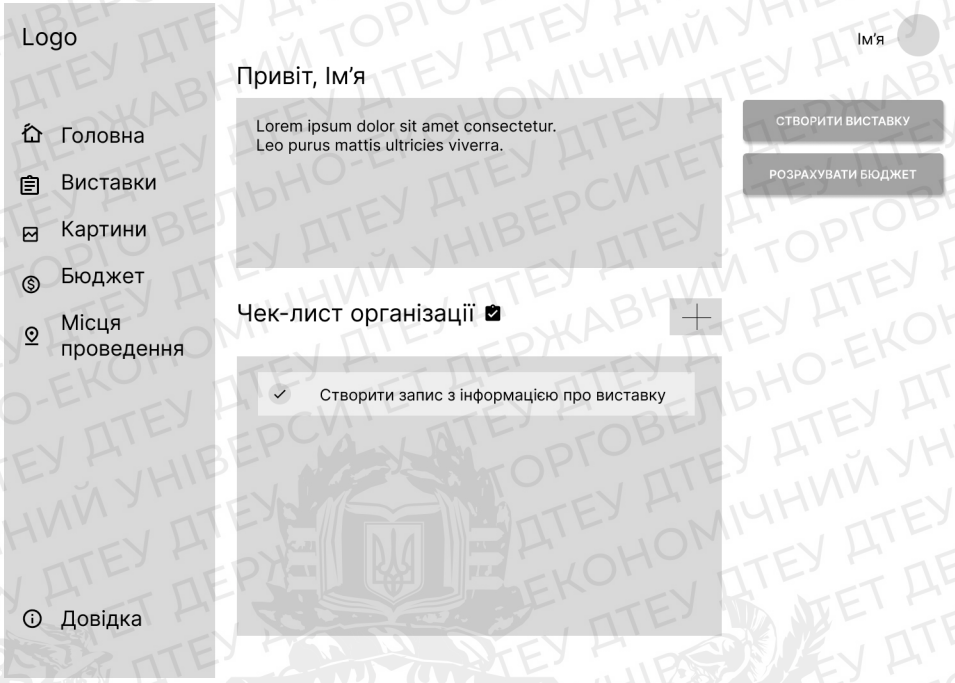

Рис. 3.3 Прототип панелі управління

Для панелі управління було розроблено навігацію, яка складається з розділів: Головна, Виставки, Картини, Бюджет, Місця проведення, Довідка. На головній сторінці можна побачити робоче поле з блоком чек лист , а також повідомлення для користувача. Зверху знаходиться зображення облікового запису користувача та ім'я, воно слугує меню, яке можна відкрити і побачити три опції «Профіль», «Налаштування» та «Вихід». З правого боку знаходяться кнопки «Створити виставку», «Розрахувати бюджет» для швидкого доступу до функцій.

Слідом за головною сторінкою панелі управління були розроблені прототипи для таких розділів: Виставки, Картини, Бюджет, Місця проведення.

Розділ Виставки має вигляд блоку з таблицею, де вказані усі створені виставки з їх назвою та датою проведення, і є кнопка для створення виставки. Прототип вигляду розділу «Виставки» представлено на рисунку 3.4.

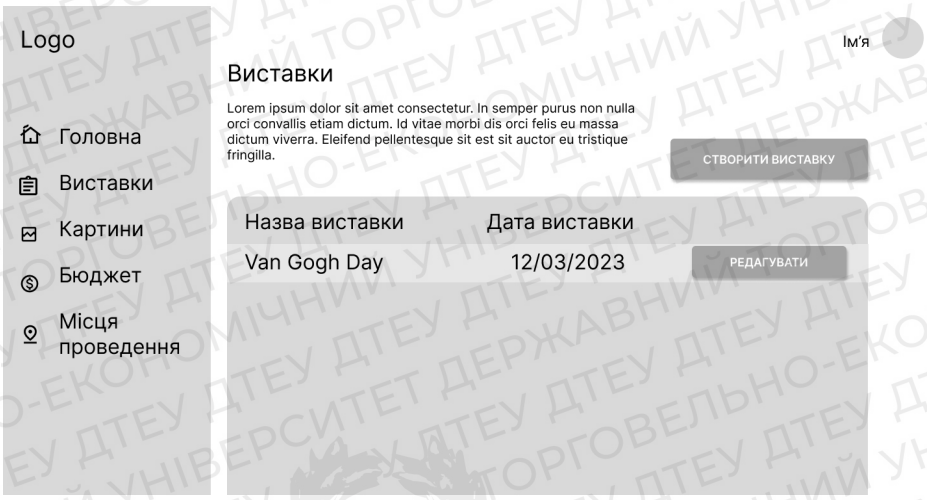

Рис. 3.4 Прототип розділу «Списки картин» панелі управління

В доповнення до всього за пунктами вказаними у технічному завданні були розроблені прототипи форм реєстрації(для Організатора та Організації) та входу до системи переглянути їх можна в Додатку Б. В розділі «Бюджет» створено прототип калькулятору для розрахунку витрат на організацію(рис.3.5)

Рис. 3.5 Прототип розділу «Бюджет» панелі управління

Коли прототипи сайту були завершені, був розроблений повноцінний макет майбутнього застосунку, який включає сайт для інформування користувачів і панель керування(дешборд), що містить функціонал інформаційної системи. На головній сторінці сайту(рис.3.6) знаходиться меню, логотип, кнопки для автентифікації, і основний блок з текстом, зображенням та кнопкою переходу до реєстрації.

Весь макет був розроблений згідно кольорами бренду і логотипу.

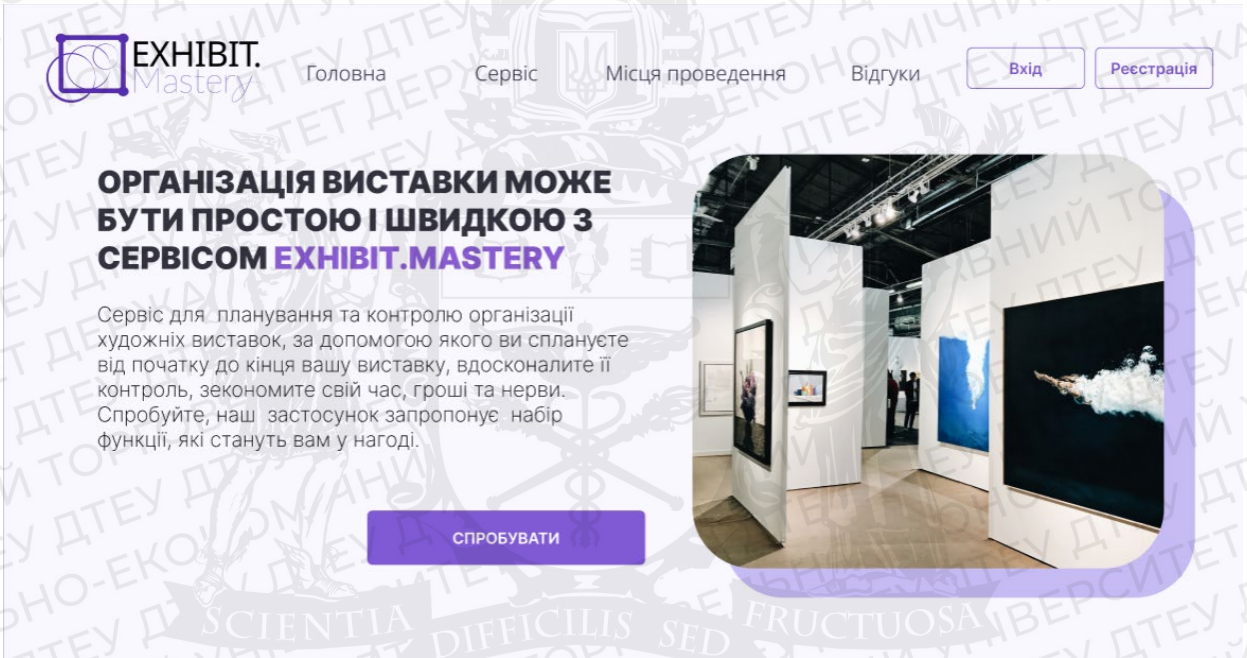

Рис. 3.6 Розроблений дизайн сайту Головна сторінка

В блоці «Сервіс» (рис.3.7) сайту міститься короткий опис панелі керування та її функцій у вигляді карток з функціями. Кожна картка має назву та опис застосування функції.

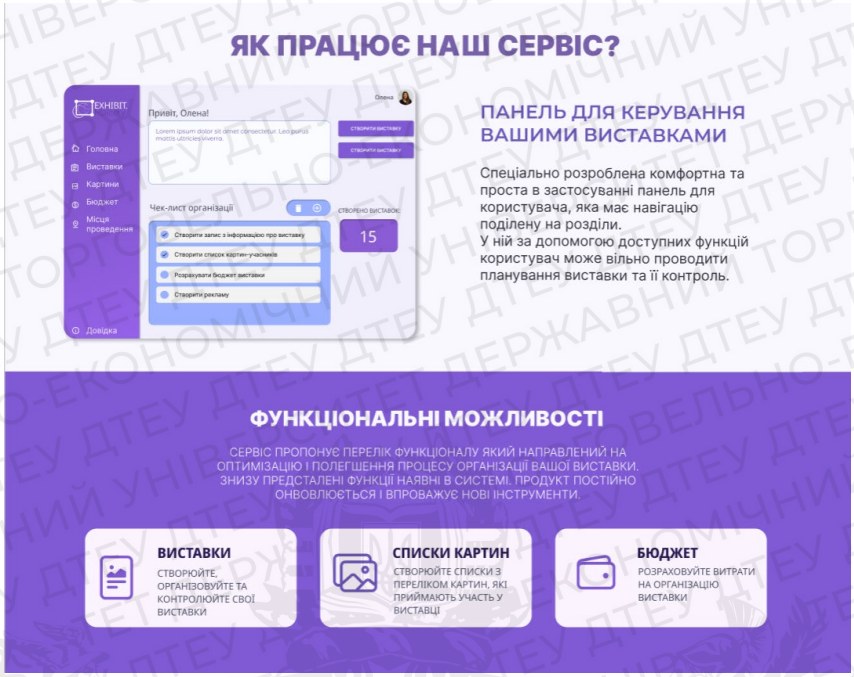

Рис. 3.7 Розроблений дизайн сайту розділ Сервіс

В розділі «Місця проведення»(рис.3.8) користувач може побачити, які види місць проведення виставки він може обрати та вказувати в панелі управління, в блоці також є кнопка для швидкого переходу до реєстрації.

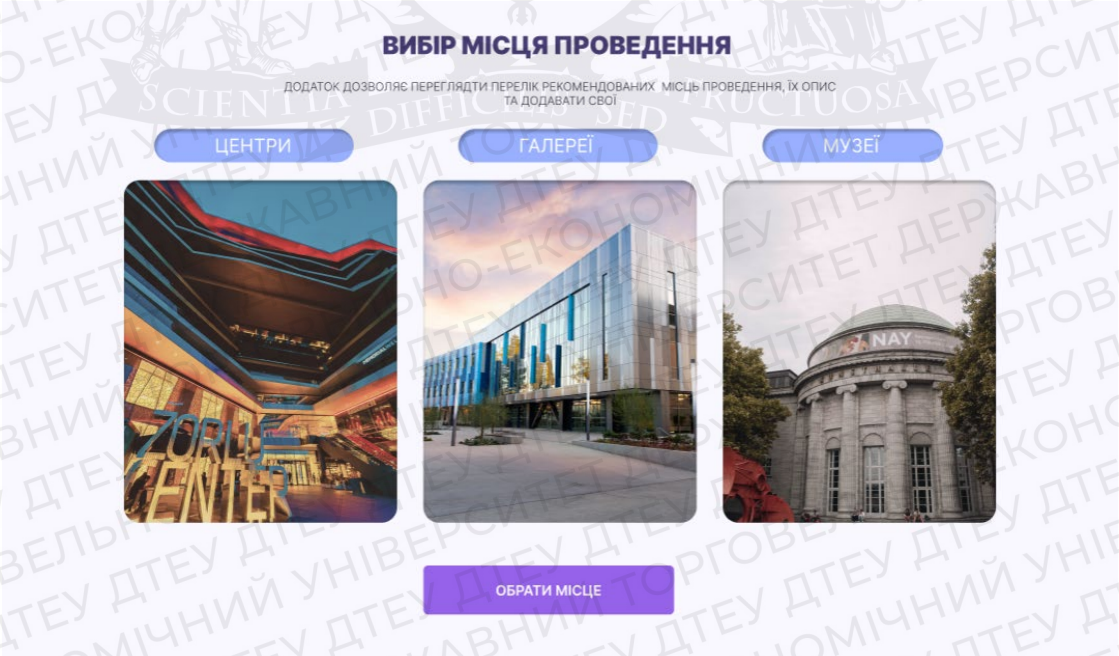

Рис. 3.8 Розроблений дизайн сайту розділ Місця проведення

Далі міститься розділ з відгуками користувачів системи, блок для підписки на розсилку від сервісу, а також нижня панель з навігацією, кнопками для автентифікації і контактною інформацією.

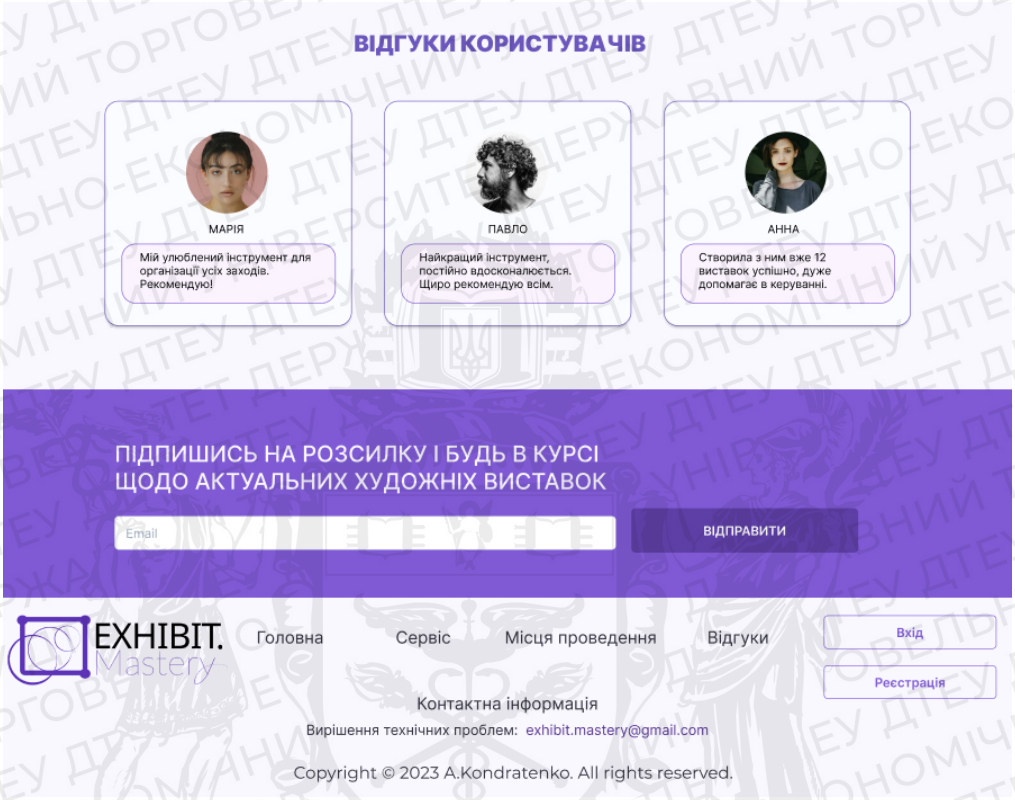

Рис. 3.9 Розроблений дизайн сайту розділ «Відгуки» та нижня панель сайту

Згодом був розроблений дизайн панелі керування(рис.3.10), який включає верхню частину сайту у якій міститься відображення профілю користувача з ім'ям, зображення профілю, а також привітання «Привіт, <Ім’я користувача>. Також застосунок налічує панель навігації, за допомогою якої можна переміщуватися по наявним розділам, чек лист, який призначений для фіксування завдань для організації виставки, і дозволяє створювати і видаляти завдання. Зверху міститься блок повідомлення для користувача від системи, і кнопки швидкого доступу до функції «Створити виставку», «Перегляд місць проведення».

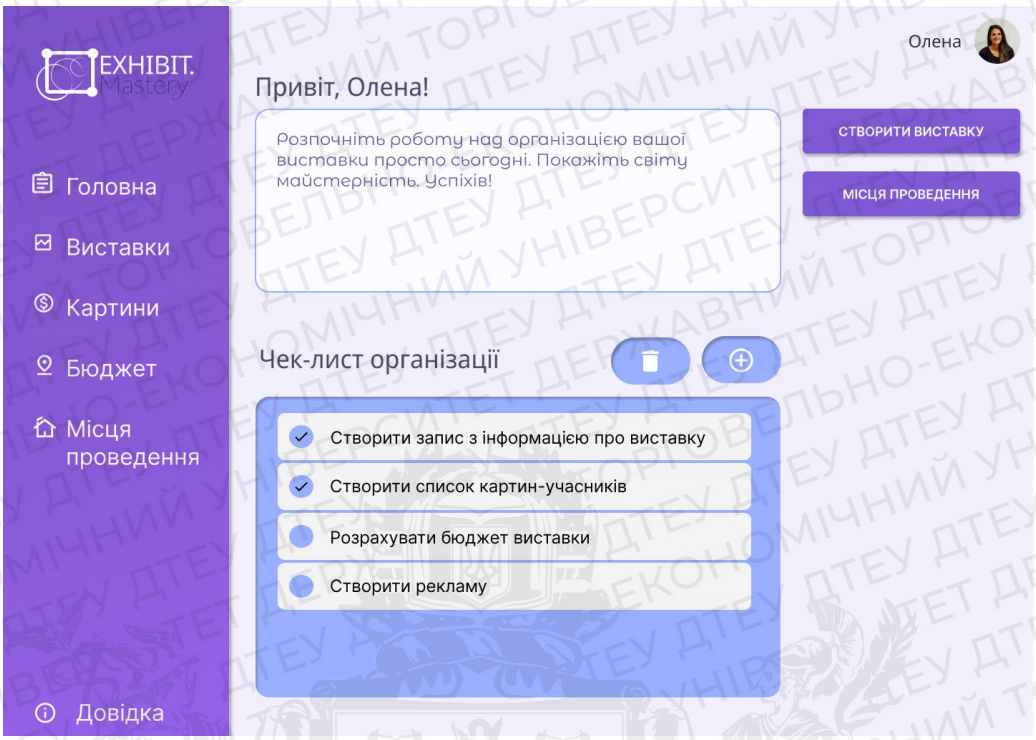

Рис. 3.10 Розроблений дизайн панелі керування з інформаційною системою

Розроблено вигляд всіх сторінок з меню та наведено приклад сторінки з Виставки(рис.3.11), яка складається з таблиці виставок, де є колонки: назва виставки, дата виставки та перегляд виставки. Над таблицею є кнопка «Створити виставку» для переходу на форму сторінку з формою для створення нового запису про виставку. Аналогічні таблиці, з різними параметрами наведені на сторінках «Списки картин», «Місця проведення».

На сторінці «Місця проведення» у таблиці можна натиснути на запис і переглянути всі дані про конкретне місце для проведення виставок зі списку.

Детальніше ознайомитися з макетами усіх сторінок панелі управління інформаційною системою для організації виставок можна у Додатку В.

Сайт з інформацією та панель управління повністю пройшли етап верстки і були розроблені за допомогою Next.js, Tailwind CSS, Chakra UI.

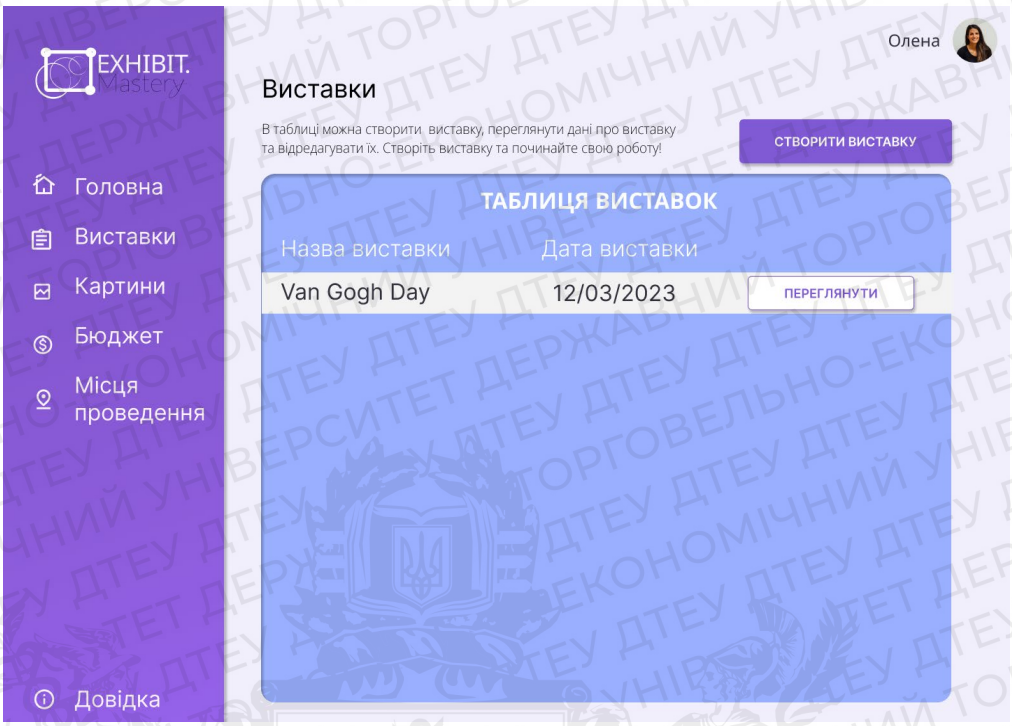

Рис. 3.11 Розроблений дизайн сторінки «Виставки» з панелі управління

Усі розроблені прототипи, макет сайту, форми та усі сторінки панелі навігації наведені у програмі Figma за посиланням:

https://www.figma.com/file/3SwMaW94TvbCCHbFZMqtij/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0-%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B0?type=design&node-id=0%3A1&t=OGlDmMGsrQ7E8X55-1

3.2 Розробка бази даних для ІС

Модель бази даних розроблялася в програмному забезпеченні MySQL Workbench Model. На початку роботи було важливо визначити, які дані є най важливішими і на їх основі розробити першу таблицю для моделі бази даних. Найважливішими даними веб-застосунку є дані, які створюються користувачем, а отже дані про Організатора або Організацію. Було розроблено таблиці з даними про користувачів, які надалі будуть доповнюватися даними нових користувачів після їх реєстрації в системі.

В розроблених таблицях наведені дані, які користувачі вводять при реєстрації в системі. В таблиці про організатора існують поля є user_Organizator_id (перший ключ це ідентифікатор таблиці), ім'я організатора (name_Organizator), фамілія (lastname_Organizator), пароль (password_Organizator), електронна пошта (email_Organizator), а також додаткова інформація про псевдонім організатора (митця) (nickname_Organizator), діяльність (менеджер, художник) (activity_Organizator).

В таблиці інформації про організацію існує user_Organization_id(перший ключ), назва організації (name_Organization), пароль (password_Organization), пошта (email_Organization), країна організації (country_Organization) та діяльність організації (activity_Organization). За пріоритетністю даних системи далі слідують дані про виставки, тому додано ще одну таблицю з даними, які має мати кожна виставки.

У таблиці «Виставки»(рис.3.12) вказуються дані про назву виставки, опис виставки(має обмеження в 150 символів), тип виставки, дата проведення, і організатор або організація, що створює запис про художню виставку. Виставки поєднуються з Організаторами та Організаціями лініями зв'язку типу one-to-many, так як організатор виставки має бути один, а виставок створити може багато.

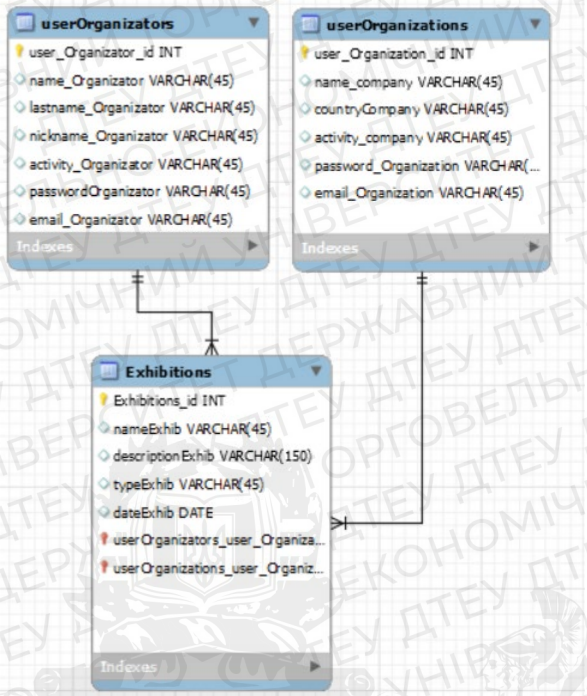

Рис. 3.12 Таблиця бази даних «Виставки»

На рисунку 3.13 зображено розроблені таблиці «Списки картин» і «Картини» у базі даних.

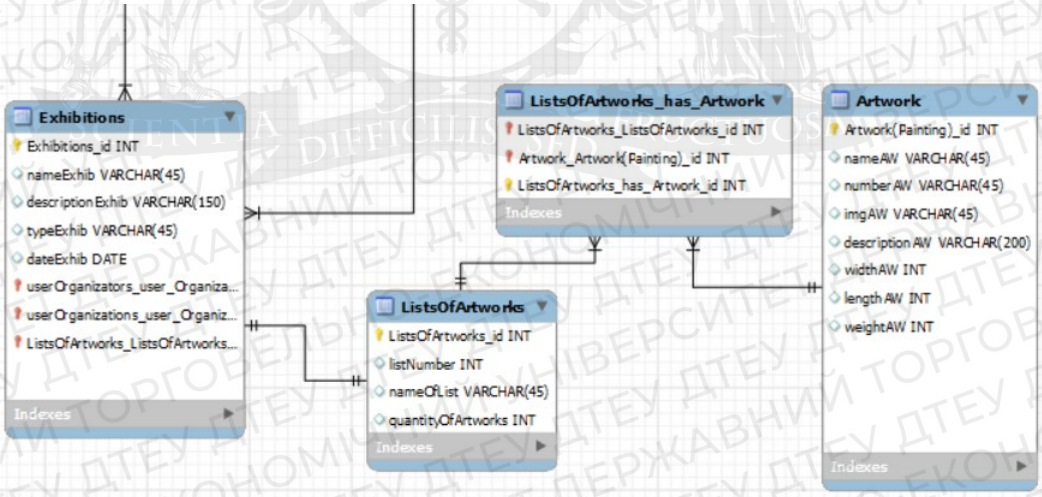

Рис. 3.13 Таблиці бази даних «Списки картин» і «Картини»

Таблиця Картини складається з даних назви, номеру, зображення, опису, ширини, довжини та ваги картини. Таблиця у якій відображені списки картин складається з параметрів ідентифікатор списку картин, номер списку, ім’я списку, кількість робіт. Ця таблиця поєднується типом зв’язку Many-to-Many

з таблицею Картина або арт-робота, що означає що арт-робота може бути у декількох списках, і в списках може існувати більше ніж одна картина. Таблиця ListOfArtworks_has_Artwork має лише два параметри, які вказують на ідентифікатори таблиці арт-роботи та таблиці список картин, та також має власний перший ключ.

Таблиці «Картини» та «Автори» мають взаємозв'язок типу One-to-Many, тому що автор може мати багато різних робіт, а картина зазвичай має лише одного автора. В таблиці з даними автора вказані його ім'я, фамілія, псевдонім, країна, і ще номер телефону, так як його дуже часто вказують на виставках на випадок купівлі картини, або він може бути необхідним для вирішення технічних питань, транспортування або правильного виставлення експозиції виставки.

Далі додано дані про місце проведення, а саме назва місця проведення, країна місця проведення виставки, адреса, кількість доступних для проведення залів. Таблиця «Місця проведення» поєднується з таблицею «Виставки» типом зв'язку One-to-One, так як в основному звичайні художні виставки мають лише одне місце проведення, а місця проведення також досить рідко поєднують у собі велику кількість художніх виставок в один час.

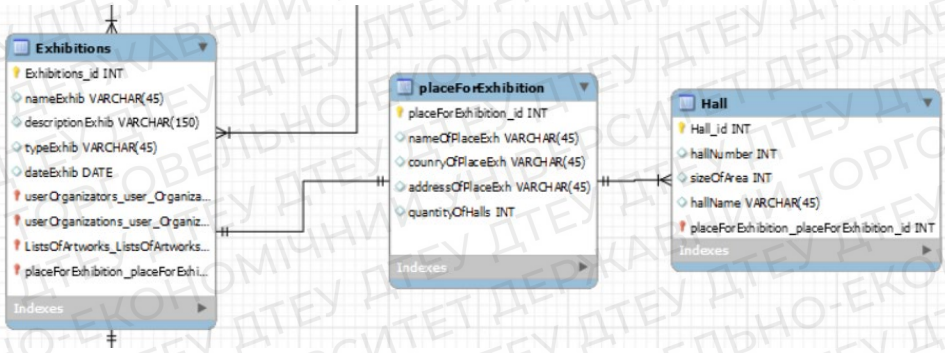

Рис. 3.14 Таблиці бази даних «Місця проведення виставки» та «Зали»

В результаті була розроблена повноцінна модель бази даних(рис.3.15), на основі, якої далі розроблялася база даних для інформаційної системи.

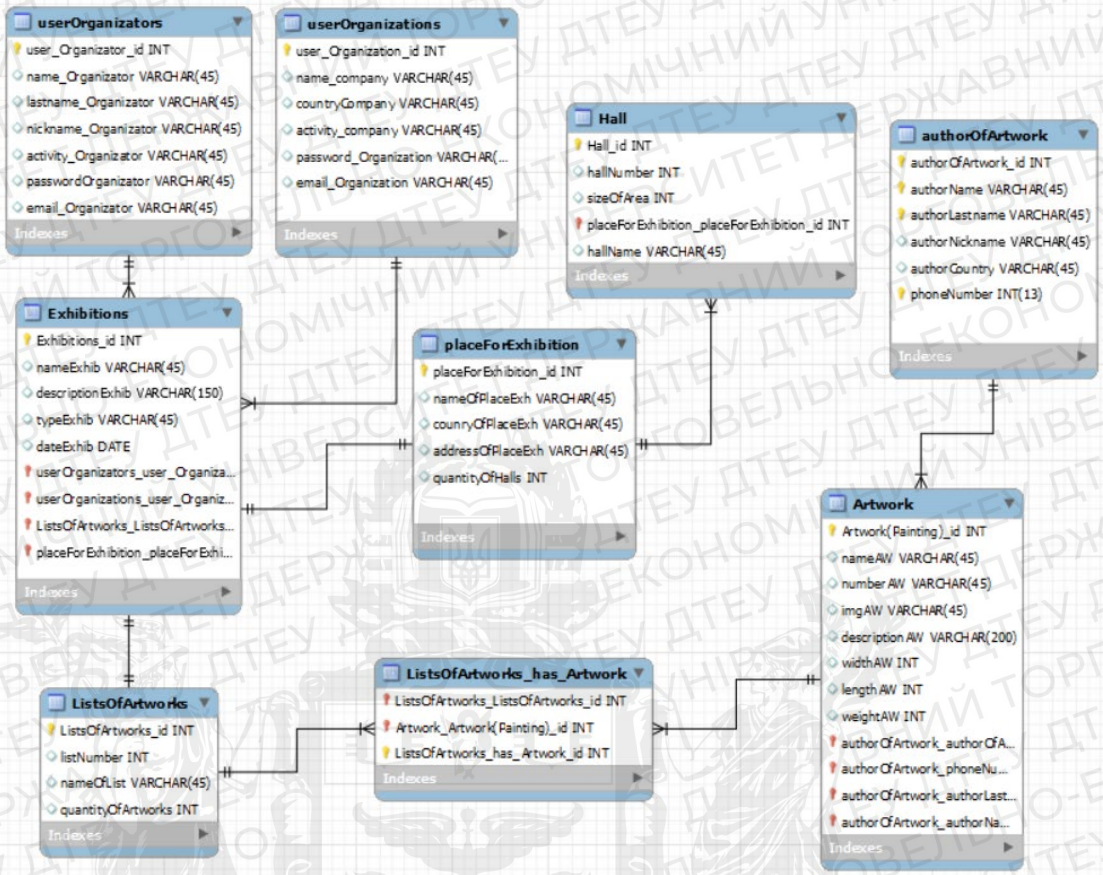

Рис. 3.15 Загальний вигляд розробленої моделі бази даних

Після того як була розроблена модель даних, далі розроблялася сама база даних, за допомогою MySQL Workbench. Щоб розробити базу даних потрібно було скористатися мовою SQL і створити схему за допомогою команди CREATE SCHEMA назва бази даних.

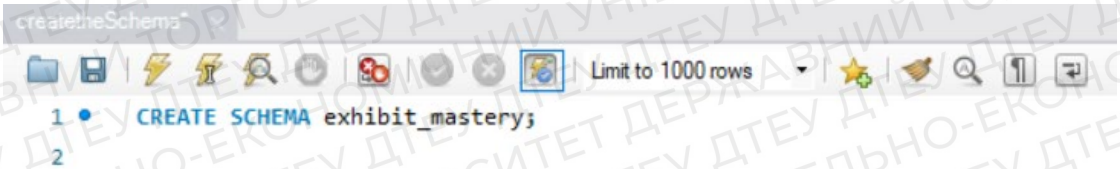

Рис. 3.16 Команда для розробки бази даних

Створення таблиці в MySQL реалізується за допомогою команди CREATE TABLE IF NOT EXISTS ‘назва бази даних’, ‘назва таблиці’. На рисунку 3.0 наведено приклад створення таблиці у вигляді коду:

```
CREATE TABLE IF NOT EXISTS `exhibit_mastery`.`userOrganizators` (
  `user_Organizator_id` INT NOT NULL,
  `name_Organizator` VARCHAR(45) NULL,
  `lastname_Organizator` VARCHAR(45) NULL,
  `nickname_Organizator` VARCHAR(45) NULL,
  `activity_Organizator` VARCHAR(45) NULL,
  `passwordOrganizator` VARCHAR(45) NULL,
  `email_Organizator` VARCHAR(45) NULL,
  PRIMARY KEY (`user_Organizator_id`))
ENGINE = InnoDB;
```

Рис. 3.17 Створення нової таблиці у базі даних

В результаті розроблена база даних, якою можна оперувати, додавати, видаляти, змінювати дані, та проводити з ними операції.

Весь код, який був написаний для розробки бази даних в MySQL Workbench наведений в Додатку Г.

3.3 Розробка функціонального наповнення web-застосунку

Інтерфейси головного сайту та панелі керування були розроблені за допомогою Next.js та стилізований за допомогою бібліотеки Tailwind CSS, з додаванням компонентів з Chakra UI.

Маршрутизація у системі створена за допомогою Next.js Router. Принцип якої заснований на файлах, де кожен файл є сторінкою додатку. Навігація між сторінками у веб застосунку створена за допомогою компоненту Link.

За допомогою OPM Prisma у Visual Studio Code IDE за допомогою команди npx prisma db pull було згенеровано моделі даних, і розроблена міграція, використовується для керування змінами, на основі бази даних MySQL, яка розроблялась раніше(див.п.п.3.2). Створені моделі в подальшому використовувались для розробки реєстрації та входу до системи користувача, кожна модель даних була розкладена на складові або атрибути, тип кожного був прописаний(рис.3.18).

Моделі даних в системі застосовуються для створення, оновлення, читання даних безпосередньо.

```
model placeforexhibition {
  placeForExhibition_id Int          @id
  nameOfPlaceExh        String?      @db.VarChar(45)
  counryOfPlaceExh      String?      @db.VarChar(45)
  addressOfPlaceExh     String?      @db.VarChar(45)
  quantityOfHalls       Int?
  exhibitions           exhibitions[]
  hall                  hall[]
}
```

Рис. 3.18 Приклад створеної моделі даних для виставки

3.3.1 Навігація на сторінці панелі управління

Було розроблене меню для користувача або навігація системи, що знаходиться з лівого боку екрану та дозволяє отримати доступ до всіх сторінок з переліку. В системі представлено розділи: Головна, Виставки, Картини, Бюджет, Місця проведення, Довідка.

Код для створення панелі навігації(меню), а саме з файлу Sidebar.tsx, наведено у додатку Д.

3.3.2 Реєстрація в системі:

Для реалізації реєстрації в системі додатково застосовувались NextAuth.js – система для забезпечення автентифікації користувачів. Щоб зареєструватися в системі користувач має натиснути на кнопку «Реєстрація» на головному сайті і після цього заповнити форму для реєстрації, вказавши в ній свою електронну пошту, пароль, ім’я та фамілію.

Вигляд форми реєстрації наведено на рисунку 3.19.

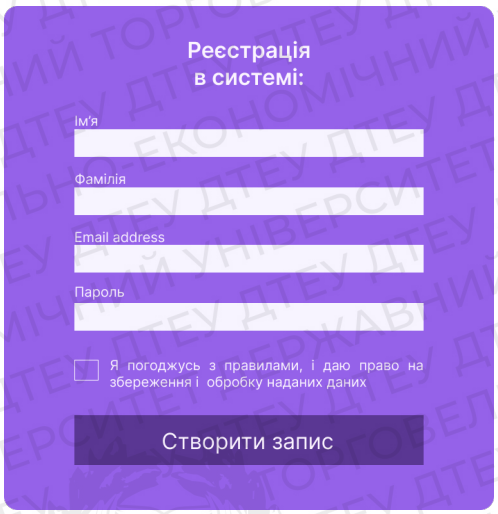

Рис. 3.19 Форма для реєстрації користувача у системі

Форма розроблена за допомогою компонентів Chakra UI, та додаткової стилізації з Tailwind CSS(рис.3.20).

```
<div className="justify-center font-sans place-content-center w-md p-12 block rounded-lg bg-vioex-medium ">
  <div className="flex flex-col justify-center ">
    <h2 className="justify-center text-center font-semibold text-vioex-light">
      Реєстрація в системі:
    </h2>
    <div className="bg-vioex-tx-light">
      <h5 className="bg-vioex-medium text-vioex-light text-[15px] font-normal">
        Ім'я
      </h5>
      <Input size="xs"></Input>
      <h5 className="bg-vioex-medium text-vioex-light text-[15px] font-normal">
        Фамілія
      </h5>
      <Input size="xs"></Input>
      <h5 className="bg-vioex-medium text-vioex-light text-[15px] font-normal">
        Email
      </h5>
      <Input size="xs"></Input>
      <h5 className="bg-vioex-medium text-vioex-light text-[15px] font-normal">
        Пароль
      </h5>
      <Input size="xs"></Input>
    </div>
    <div className="flex flex-row items-center">
      <Checkbox className="m-2 "></Checkbox>
      <p className="text-vioex-light m-4 text-[10px]">
        Я погоджусь з правилами, і даю право на збереження і обробку
        наданих даних
      </p>
    </div>
    <button className="rounded-md text-vioex-light p-3 bg-vioex-dark">
      Зареєструватися
    </button>
```

Рис. 3.20 Код для оформлення форми для реєстрації

3.3.3 Меню користувача

Випадаюче меню в верхній частині системи для користувача(рис.3.21), було розроблено за допомогою компонентів з бібліотеки Chakra UI. Принцип

роботи є простим, користувач натискає на іконку профілю і відкривається меню з переліком пунктів: «Профіль», «Налаштування», «Вихід».

```
<Menu>
  <MenuButton
    className="font-light font-normal"
    as={Button}
    rightIcon={<ChevronDownIcon />}
  >
    Ліна Костенко
  </MenuButton>
  <Image
    className=""
    src={'./assets/icon-user-header.svg'}
    alt={'icon-user-Header'}
    width={60}
    height={60}
  ></Image>
  <MenuList>
    <MenuItem>Профіль</MenuItem>
    <MenuItem>Налаштування</MenuItem>
    <MenuItem>Вихід</MenuItem>
  </MenuList>
</Menu>
```

Рис. 3.21 Код для створення випадаючого меню

Розроблено за допомогою компонентів з бібліотеки Chakra UI: Menu, MenuButton(кнопка меню), Image(зображення профілю), MenuList(список пунктів у меню), MenuItem(пункт у меню).

3.3.4 Чек лист

На головній сторінці панелі управління був розроблений чек лист, за допомогою введення даних для нового завдання при натисканні на кнопку «Додати»(+) відкривається у форма у яку вводяться дані і додаються до таблиці.

```
export default async function CheckList() {
  const tasks = await getAllTodos();

  return (
    <div className="justify-center">
      <h2 className="m-5 justify-center items-center text-2xl font-sans font-normal px-[300px]">
        Чек ліст організації
      </h2>
      <div className='w-full'>
        <AddTask />
        <Box className="max-w-lg flex mx-auto pl-4 box-border place-center justify-center
        bg-blupan-panel rounded-[20px] shadow-pan-1">
          <TodoList tasks={tasks} />
        </Box>
      </div>
      <div></div>
    </div>
  );
}
```

Рис. 3.22 Основний код для чек листа

Для чек листу були розроблені компоненти AddTask.tsx та TodoList.tsx. AddTask містить в собі функції і кнопку для створення нового завдання, TodoList містить у собі таблицю до якої підключається додатковий компонент Task який передає характеристики введеного завдання, такі як текст і стан чек листу. Нове завдання чек-листу реалізується за допомогою форми Modal, що виникає на екрані після натискання кнопки, і має заголовок, поле для вводу – Input, і кнопку для створення запису. Повноцінний код файлу AddTask наведено в Додатку Д.

```
const Task: React.FC<TaskProps> = ({ task }) => {
  return (
    <tr
      className="bg-vioex-tx-light text-blupan-texti p-1 rounded-[20px] m-4 w-full"
      key={task.id}
    >
      <td>
        <Checkbox className="bg-blupan-panel" defaultChecked></Checkbox>
      </td>
      <td>{task.text}</td>
    </tr>
  );
};

export default Task;
```

Рис. 3.23 Код для створення нового завдання(Task.tsx) в чек листі

Завдання має таку структуру: рядок з id завдання, у якому є дві комірки зі чек-боксом, для позначення статусу роботи та введений попередньо користувачем текст.

3.3.5 Таблиця «Виставки»

При переході на сторінку “Виставки” можна побачити таблицю у якій вказано перелік рекомендованих місць проведення. Ця функція була реалізована за допомогою таблиці даних та компонентів. Для створення таблиці розроблений компонент OneExhibition.tsx, відповідає за відображення рядків з даними у таблиці з виставками. Цей файл імпортує дані з файлу exhib.json, в якому знаходиться потрібна інформація про виставку, далі вносить її у змінну, яка вміщає в собі стрілкову функцію з параметром exh. Таблиця містить комірку з яка має конкретний id виставки(exh), і далі окремо в кожний комірку додається запит на певну властивість виставки з вказаним id. Також додано комірку яка імпортує кнопку для перегляду виставки.

```
const OneExhibition: React.FC<ExhibProps> = ({ exh }) => {
  return (
    <tr
      className="bg-vioex-tx-light text-blupan-texti p-1 rounded-[20px] m-4 w-full"
      key={exh.id}
    >

      <td>{exh.text}</td>
      <td>{exh.date}</td>
      <td><ViewTheExh/></td>
    </tr>
  );
};

export default OneExhibition;
```

Рис. 3.24 Код для рядка у таблиці «Виставки»

Дані для компонентів беруться з файлу у форматі JSON. Файл з рядками приєднано до компоненту TableWithExh.tsx, що містить всю таблицю, і він у свою чергу доєднаний до сторінки з виставками.

```
export default async function Exhibitions() {
  const exhs = await getAllExhib();
  return (
    <div className="m-[300px] border font-sans p-8 bg-vioex-light text-black">
      <h1 className="text-normal">Виставки</h1>
      <p>
        В таблиці можна створити виставку, переглянути дані про виставку та
        відредагувати їх. Створіть виставку та починайте свою роботу!
      </p>
      <div>
        <Button
          className="bg-vioex-medium w-304 h-63 "
          colorScheme="purple"
          variant="solid"
        ></Button>
      </div>
      <div>
        <Box
          className="max-w-lg flex mx-auto pl-4 box-border place-center justify-center
        bg-blupan-panel rounded-[20px] shadow-pan-1"
        >
          <TableWithExh exhs={exhs} />
        </Box>
      </div>
      <Link href={'/'}>Назад на головну</Link>
    </div>
  );
}
```

Рис. 3.25 Код сторінки з виставками

Отже, реалізована інформаційна система з підтримки процесу організації виставок має простий користувацький інтерфейс та легка у користуванні. Може допомагати з організацією виставок та їх плануванням, а також має багато шляхів для майбутнього вдосконалення системи за допомогою модифікації функцій та імплементації додаткових підсистем та інструментів, що зроблять роботу менеджера ефективнішою і покращать контроль над процесом організації.

ВИСНОВКИ

У випускній кваліфікаційній роботі було проведено дослідження процесу розробки інформаційної системи для автоматизації процесу організації художніх виставок. Протягом розробки була розглянута предметна область, здійснений аналіз та порівняння веб-застосунків схожої тематики і функціональності до системи, що розроблялася, виявлено їх переваги та недоліки. Детально розглянуто процес організації художньої виставки і на основі аналізу виведено основні задачі інформаційної системи для організації виставок. Засновані на раніше поставлених задачах, були сформовані функціональні та нефункціональні вимоги до інформаційної системи. Створене технічне завдання, де визначена концепція застосунку, цільова аудиторія, етапи розробки, всі вимоги, структура веб-застосунку і його вміст. Розроблено проект інформаційної системи за допомогою моделюванням бізнес-процесів та засобів проектування. Створено брендінг для інформаційної системи, проведено прототипування сторінок веб-застосунку, дизайн для застосунку. Розроблено базу даних для системи, клієнтський інтерфейс та основний функціонал.

В результаті усіх проведених операцій отримано такі **висновки**:

1. Процес організації виставок є комплексним і вміщує велику кількість завдань, з різним рівнем складності. Автоматизація цього процесу за допомогою інформаційної системи значно покращує планування і полегшує задачі організатора, допомагає контролювати виконання завдань. В такій системі можуть міститися різноманітні функції, що дозволяє в довгостроковій перспективі вдосконалювати інформаційну систему добудовуючи додаткові інструменти.
2. Моделювання бізнес-процесів допомагає краще ознайомитися з процесом, який буде автоматизований і зрозуміти, який шлях краще обрати для розробки, краще провести розподіл завдань за пріоритетом, с точки зору реалізації функцій.

3. Попередня розробка проекту із формулюванням усіх вимог щодо інформаційної системи, значно спрощує вибір технологій для її реалізації. Зважаючи на цільову аудиторію та маючи розуміння якими будуть дії користувача у застосунку, легше розробити комфортний інтерфейс та уникнути багатьох майбутніх проблем з системою.
4. Розроблена інформаційна система може слугувати корисним інструментом в діяльності менеджера або митця та допомагати організовувати презентацію мистецьких робіт швидше.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Авраменко В. С. Проектування інформаційних систем: навчальний посібник : навч. посіб. / Валентин Семенович Авраменко, Артем Сергійович Авраменко. – Черкаси : Черкас. нац. ун-т ім. Б. Хмельн., 2017. – 434 с.

2. Бардус І. О. Бази даних у схемах(на основі фундаменталізованого підходу) : навч. посіб. / Ірина Олександрівна Бардус. – Харків : «Дісаплюс», 2017. – 133 с.

3. Вдовічен А. А. Організація виставкової діяльності : навч. посіб. / Анатолій Анатолійович Вдовічен, Ольга Геннадіївна Вдовічена. – Чернівці : Технодрук, 2018. – 264 с.

4. Добролюбова М. В. Програмування баз даних: конспект лекцій : навч. посіб. / Марина Валеріївна Добролюбова. – Київ : КПІ ім. Ігоря Сікорського, 2021. – 275 с.

5. Проектування інформаційних систем: Загальні питання теорії проектування ІС (конспект лекцій) : конспект лекцій / уклад.: О. С. Коваленко, Л. М. Добровська. – Київ : КПІ ім. Ігоря Сікорського, 2020. – 192 с.

6. Табунщик Г. В. Проектування та моделювання програмного забезпечення сучасних інформаційних систем : навч. посіб. / Г. В. Табунщик, Т. Каплієнко, О. Петрова. – Запоріжжя : Дике Поле, 2016. – 250 с.

7. Ткаченко А. Як створити веб-додаток: типи, переваги, принцип роботи - Wezom [Електронний ресурс] / Аліна Ткаченко // IT-компанія повного циклу програмних продуктів WEZOM - Київ, Україна. – Режим доступу: https://wezom.com.ua/ua/blog/kak-sozdat-veb-prilozhenie (дата звернення: 12.01.2023).

8. Омельчук Є. Що таке Figma: функції, інструменти та переваги - академія Wezom [Електронний ресурс] / Євген Омельчук // Академия Wezom. – Режим доступу: https://wezom.academy/ua/chto-takoe-figma-funktsii-instrumenty-ipreimuschestva/ (дата звернення: 22.02.2023).

9. Штаєр Л. Технології розробки програмного забезпечення / Л. Штаєр. – Івано-Франківськ : ІФНТУНГ, 2017. – 139 с.

10. Як організувати виставку живопису | UBI Конференц Хол [Електронний ресурс] // UBI Конференц Хол. – Режим доступу: https://ubi-hall.com.ua/uk/2018/03/14/iak-organizyvati-vistavky-jivopisy/ (дата звернення: 10.02.2023).

11. Banks A. Learning React: Functional Web Development with React and Redux / Alex Banks, Eve Porcello. – [S. l.] : O'Reilly Media, 2017. – 350 p.

12. Duckett J. HTML & CSS: Design and build websites / Jon Duckett. – [S. l. : s. n.], 2014. – 490 p.

13. Haverbeke M. Eloquent Javascript / Marijn Haverbeke. – San Francisco : No Starch Press, 2009.

14. How to set up an art exhibition | The Jotform Blog [Electronic resource] // The Jotform Blog. – Mode of access: https://www.jotform.com/blog/how-to-set-up-an-art-exhibition/ (date of access: 12.03.2023).

15. Learning JavaScript: JavaScript Essentials for Modern Application Development. – Sebastopol, California, USA : O'Reilly, 2016. – 334 p.

16. Luchaninov Y. Web Application Architecture: Choosing the Right Type in 2023 [Electronic resource] / Yurii Luchaninov // MobiDev. – Mode of access: https://mobidev.biz/blog/web-application-architecture-types (date of access: 11.04.2023).

17. Single Page Application (SPA) vs Multi Page Application (MPA): Which Is The Best? | Clean Commit [Electronic resource] // Application Development & Headless eCommerce | Clean Commit. – Mode of access: https://cleancommit.io/blog/spa-vs-mpa-which-is-the-king/ (date of access: 11.03.2023).

18. Sotnik S. Development Features Web-Applications [Electronic resource] / S. Sotnik, T. Shakurova, V. Lyashenko // International Journal of

Academic and Applied Research (IJAAR). – 2023. – Vol. 7, no. 1. – P. 79–85. – Mode of access: https://openarchive.nure.ua/handle/document/21600.

19. Web Application Architecture Fundamentals: The Full Roadmap [Electronic resource] // Cleveroad Inc. - Web and App development company. – Mode of access: https://www.cleveroad.com/blog/web-application-architecture/ (date of access: 11.03.2023).

20. What is a Database? Definition, Meaning, Types with Example [Electronic resource] // Guru99. – Mode of access: https://www.guru99.com/introduction-to-database-sql.html (date of access: 10.03.2023).

21. What is a Web App? - Web Application Explained - AWS [Electronic resource] // Amazon Web Services, Inc. – Mode of access: https://aws.amazon.com/what-is/web-application/?nc1=h_ls (date of access: 12.04.2023). – Title from screen.

22. Grippa V. M. Learning MySQL / Vinicius M. Grippa, Sergey Kuzmichev. – [S. l.] : O'Reilly Media, Incorporated, 2021.

23. CSS: Cascading Style Sheets | MDN [Electronic resource] // MDN Web Docs. – Mode of access: https://developer.mozilla.org/en-US/docs/Web/CSS (date of access: 02.05.2023).

24. Docs | Next.js [Electronic resource] // Документація Next.js. – Mode of access: https://nextjs.org/docs (date of access: 01.04.2023).

25. Documentation - Tailwind CSS [Electronic resource] // Tailwind CSS. – Mode of access: https://tailwindcss.com/docs/installation (date of access: 04.03.2023).

26. Documentation [Electronic resource] // Chakra UI. – Mode of access: https://chakra-ui.com/getting-started (date of access: 02.03.2023).

ДОДАТКИ

Додаток А

Технічне завдання для розробки інформаційної системи з підтримки процесу організації художніх виставок

1. Загальні відомості

Назва проекту: Проект «Exhibit.Mastery» - web-застосунок для організації художніх виставок. Слоган проекту: Manage and emphasize your mastery(Керуй та підкреслюй свою майстерність).

Мета проекту: вдосконалення та полегшення процесів планування та організації художньої виставки.

Призначення проекту: Призначенням проекту є розробка інформаційної системи для автоматизації процесів і функцій, які зазвичай виконуються особистими менеджерами художників(арт менеджерами), організаціями або художниками самостійно для планування та організації виставки художніх робіт за певною тематикою та з певною ціллю.

Вміст проекту: інформаційна система з підтримки процесу організації художніх виставок у вигляді web-застосунку

Строки проекту: до 30 днів

Цільова аудиторія

Вік цільової аудиторії: Люди різного віку від 16-100.

Переважаюча стать: будь-яка.

За видом професійної діяльності цільова аудиторія поділяється на такі групи:

- Митці;

- Колекціонери;
- Менеджери митців;
- Компанії, організації;
- Агенції;
- Упорядники та розпорядники;
- Виставкові центри, галереї.

Специфіка діяльності користувача:

Користувач пов'язаний з виставковою діяльністю у сфері арт мистецтва та має бажання оптимізувати процес планування і організації виставки.

Задачі, які хоче оптимізувати користувач ІС за допомогою застосунку:

- Пришвидшити процес оформлення виставки
- Систематизувати процес організації роботи
- Створити контроль продажу білетів
- Спростити розрахунок бюджету
- Спостити спосіб зберігання списків картин для виставки
- Спостити організацію додаткових послуг

Вимоги до інформаційної системи в цілому:

Інформаційна система має бути реалізована за допомогою веб-технологій у вигляді веб-застосунку. ІС потребує простого та зрозумілого клієнтського інтерфейс.

Застосунки-аналоги:

- https://www.eventzilla.net/us/home
- https://www.artsy.net/ ;
- https://www.eventbrite.com/organizer/overview/

Вимоги до інтерфейсу користувача

- Інтерфейс має бути веб-орієнтованим;
- Web-додаток має бути комфортним та простим для застосування

- Веб-застосунок має мати кольори, що мають нормальну контрастність і яскравість.
- Інтерфейс має бути інтуїтивно зрозумілим.

Вимоги, що описують функціональність проекту

Функції, має інформаційна система:

- **Створення виставки** – функція, яка дозволяє користувачу створити запис з даними про нову виставку. Форма для розміщення нової виставки включає поля вводу характеристик виставки, таких як: назва, опис, організатор, країна, тип виставки, автор, місце проведення (адреса, кімната(необов'язково)), дати проведення. Щоб створити виставку варто натиснути кнопку «Створити» інформація про виставку зберігається в базі даних.
- **Реєстрація в системі** – створення облікового запису та внесення даних користувача в систему(форма реєстрації);
- **Авторизація** - вхід в обліковий запис (форма входу в систему), який реалізується коли користувач вводить електронну пошту та пароль і натискає «Вхід»;
- **Переглядання місць проведення** – список місць з переліком залів для виставок;
- **Розрахунок бюджету** – розділ у якому знаходиться калькулятор для розрахунку необхідного бюджету виставки;
- **Створення списку картин** – створення та наповнення списку картин в панелі управління, для подальшого використання в назначених вистаках.

Вимоги до збереження інформації

Вся інформація, що знаходиться у системі має розміщуватися в базі даних.

Вимоги до розмежування доступу

Розташована на сайті інформація про систему і її можливості є загальнодоступною для усіх відвідувачів сайту. В панелі управління, яка містить інформаційну систему користувачі застосунку за правами доступу поділяються на групи:

1. **Користувач** – особа, яка тільки зайшла на сайт ще не пройшла реєстрацію та авторизацію, має доступ до інформації, має можливість зареєструватися, ввійти в систему, підписатися на розсилку вказавши попередньо електронну пошту.
2. **Організатор –** особа, яка зареєстрована у системі, має доступ до панелі управління, що містить функціонал системи. Організатор має доступ до функцій створення одноосібних виставок, перегляду створених виставок, перегляду місць проведення, може проводити розрахунки, змінювати інформацію про виставку, створювати списки картин.
3. **Організація** – користувач, обліковий запис якого належить певній компанії, агенції, галереї, має доступ до загальної частини застосунку. Додатково мають функцію для створення корпоративної виставки(у якій приймають участь декілька художників).
4. **Адміністратор** – особа, яка має доступ до системи, може видаляти неналежні дані, змінювати інформацію у профілях інших користувачів за запитом, вирішувати проблеми та несправності системи. Доступ здійснюється за допомогою використання унікальних логіна і паролю.

Вимоги до надійності

Інформаційна система має зберігати цілісність, конфіденційність даних та повинна витримувати навантаження в максимум 50 користувачів.

Технічні та апаратні вимоги

Для стабільної роботи веб-застосунку, техніка користувача має мати приблизно такі мінімальні характеристики:

1) Процессор – Intel Core 2 Duo 2GHz ;
2) Оперативна пам’ять – 2 GB та більше;
3) Мережева карта – 100 Mbit.
4) Графічна карта – 265 MB;
5) Мишка або сенсорна панель;
6) Клавіатура;
7) Монітор, 1024x768 і більше.

Кожен користувач має мати доступ до мережі Інтернет та пристрій який підтримує веб-додаток.

Етапи роботи:

1 Етап: “Дослідження”(1-2 дні)

- аналіз цільової аудиторії;
- розробка user story map;
- дослідження ринку і конкурентів;
- сегментація цільової аудиторії.

2 Етап: “Прототипу веб-застосунку з урахуванням всіх блоків та вікон” (2-3 дні)

- проектування інтерфейсу та архітектури сайту;
- детальне прототипування ключових розділів і сторінок у Figma;
- розробка wireframe-ів дизайну навігаційної панелі, форм для: реєстрації, авторизації.
- узгодження прототипу (ескізу);

3 Етап: “Повноцінний дизайн веб-застосунку з урахуванням всіх блоків та форм(2-3 дні)

- розробка повноцінного макету сайту у Figma;
- наповнення сайту контентом(текст, фото);

4 Етап: “Верстка сайту за дизайном”(5-7 днів)

- верстка усіх розділів сайту за допомогою HTML, CSS і Next.js;
- верстка панелі управління сервісом за допомогою Next.js;
- верстка форм для реєстрації, авторизації користувача;

5 Етап: “Реалізація серверної частини”(5-7 днів)

- створення маршрутизації для сайту;
- розробка бази даних в MySQL;
- створення) API для сайту (менеджер/митець або організація;
- використання бази даних MySQL для збереження даних та інструмент для інтерпретації даних Prisma.

6 Етап: “Тестування застосунку”(1-2 дні)

- тестування функціоналу: робота сайту, панелі управління, форм, кнопок;
- тестування відповідності тз.

Структура веб-застосунку:

Web-застосунок складається з сайту та панелі управління, яка містить інформаційну систему з підтримки процесу організації художніх виставок та її функції. Доступ до панелі управління стає можливим після реєстрації користувача в системі.

Структура сайту:

Верхня панель сайту (Header) містить:

- Логотип;
- Навігація. Розділи: «Головна», «Сервіс», «Місця проведення», «Відгуки»;
- Кнопка для реєстрації;
- Кнопка для входу.

Головна сторінка сайту:

На сторінці є такі елементи:

- Заголовок
- Текст
- Кнопка «Спробувати»

Розділ «Сервіс»:

- Заголовок «Інструменти застосунку»
- Сітка з переліком доступних функцій
- Заголовок «Можливості»
- Текст-опис функції

Розділ «Місця проведення»:

Містить картки з популярними місцями проведення виставки з зображенням місця, назвою, кількістю залів, містом.

Розділ «Відгуки»:

Містить картки з відгуками користувачів.

Нижня панель сайту(Footer):

- Логотип;
- Навігація. Розділи: Головна, Сервіс, Місця проведення, Відгуки;
- Контакти(Електронна пошта, соціальні мережі);
- Пошта для технічних питань;
- Логотипи партнерів(необов’язково або за наявності).

Форма для підписки розсилку про виставки:

- Електронна пошта;
- Кнопка «Надіслати».

Форма для реєстрації організатора:

- Ім’я;
- Фамілія;
- Пседонім (необов’язкове поле);
- Вибір діяльності (менеджер, митець, інший)(необов’язкове поле);
- Електронна пошта;
- Пароль;

- Кнопка «Створити».

Форма для реєстрації організації:

- Назва організації;
- Країна(необов’язкове поле);
- Вид діяльності (менеджмент, мистецтво, інше);
- Електронна пошта;
- Пароль;
- Кнопка «Створити».

Форма для авторизації:

- Електронна пошта;
- Пароль;
- Кнопка «Увійти».

Структура панелі управління:

Веб-застосунок має панель управління, де є роль організатора, організації та адміністратора.(Адміністратор має повний доступ, а менеджер частковий (вакансії, замовлення, новини).

Навігаційна панель інформаційної системи має такі розділи:

- Головна сторінка
- Виставки
- Місця проведення
- Картини
- Бюджет
- Довідка(допоміжний розділ з інформацією про систему)

Верхня панель:

- Логотип ;

- Ім’я та зображення користувача(випадаюче меню з розділами: «Профіль», «Налаштування», «Вихід»).

Головна сторінка: містить інформацію про користувача: ім’я, фамілію, діяльність, електронна пошта, зображення.

Розділ «Виставки»: має таблицю, де відображаються з наявними виставками, та кнопку «Створити виставку».

Розділ «Місце проведення»: має таблицю з наявними місцями проведення і кнопкою «Додати місце».

Розділ «Картини»: має таблицю з наявними списками картин і кнопкою «Новий список».

Розділ «Бюджет»: містить калькулятор для розрахунку бюджету з параметрами(оренда на день, кількість днів оренди, транспортування картини вага та кількість, монтування картин, вартість додаткових послуг, кнопка «Розрахувати», Загальна сума витрат)

Розділ «Довідка»: знаходиться інформація про можливості системи для користувача панелі керування.

Додаток Б

Прототипи головного сайту застосунку «Exhibit.Mastery»

Рис.1 Прототип головного сайту системи

Рис.2 Протип форми входу в систему

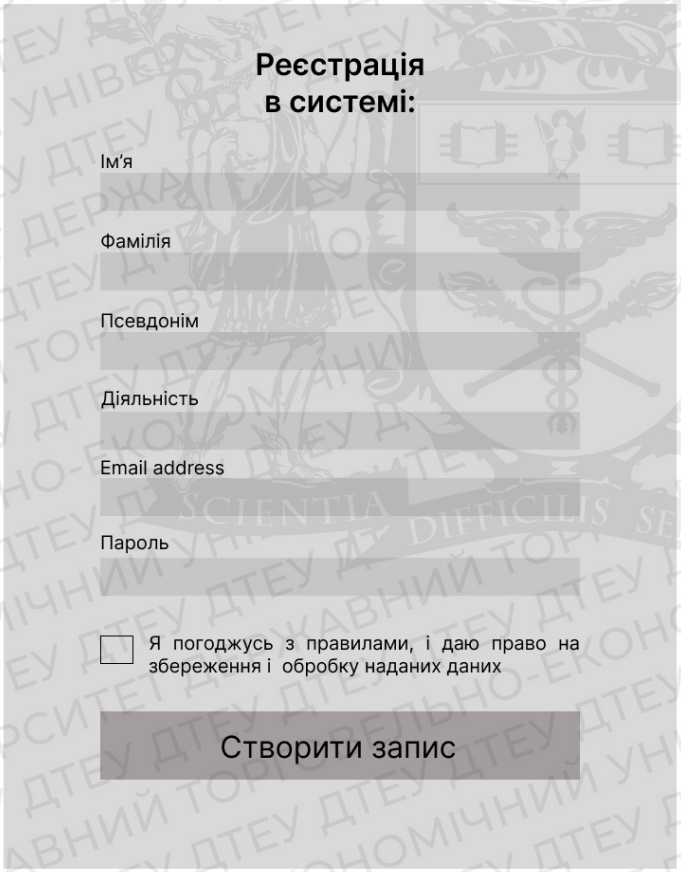

Рис.3 Прототип форми реєстрації користувача

Додаток В

Макети дизайну інтерфейсу застосунку «Exhibit.Mastery»

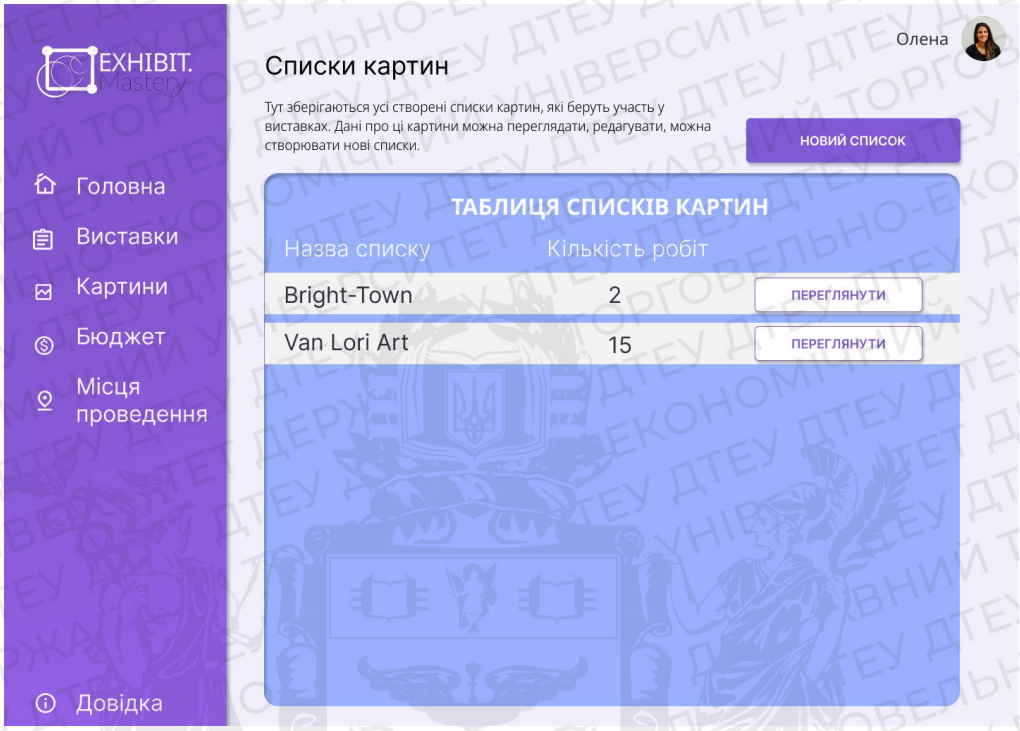

Рис.1 Сторінка «Списки картин» панелі управління системи

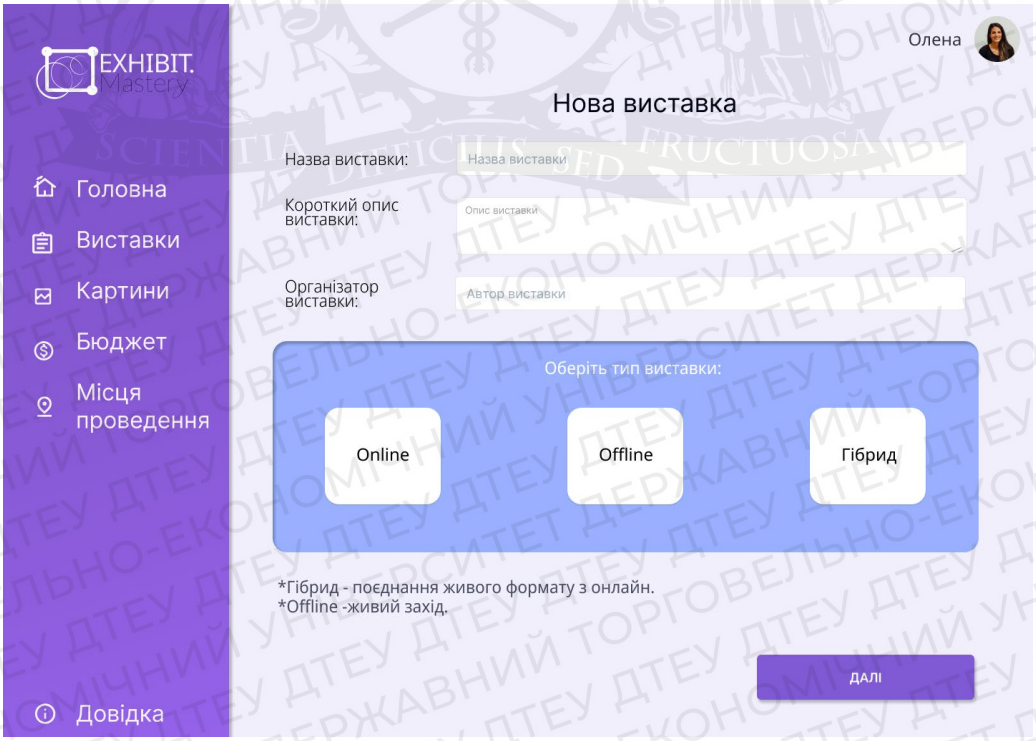

Рис.2 Форма для створення виставки

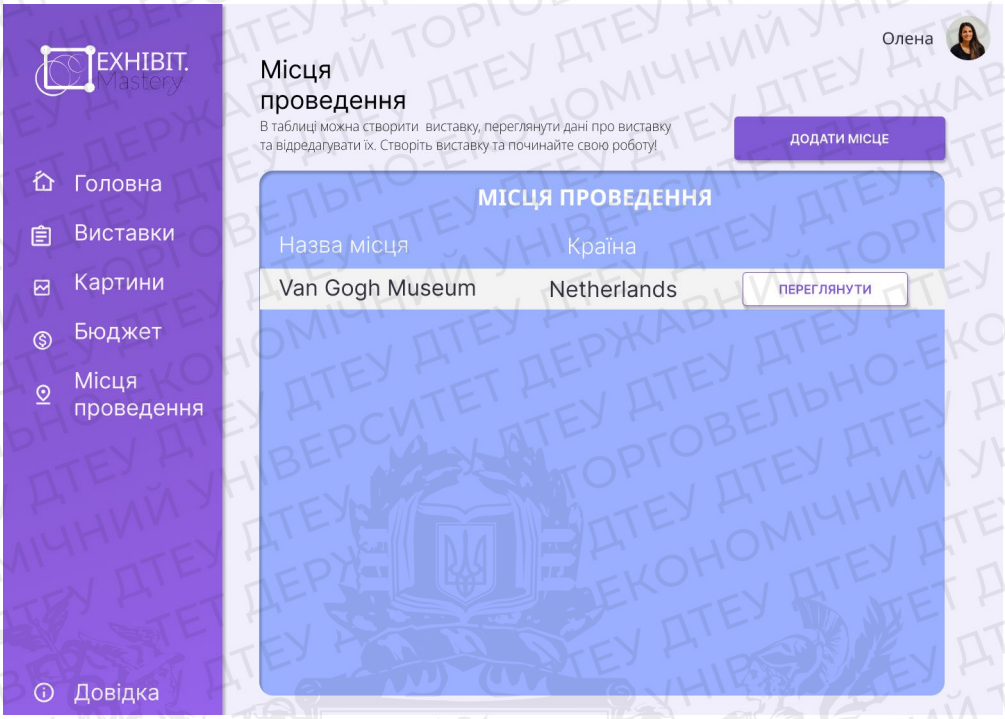

Рис.3 Сторінка «Місця проведення» у панелі управління системою

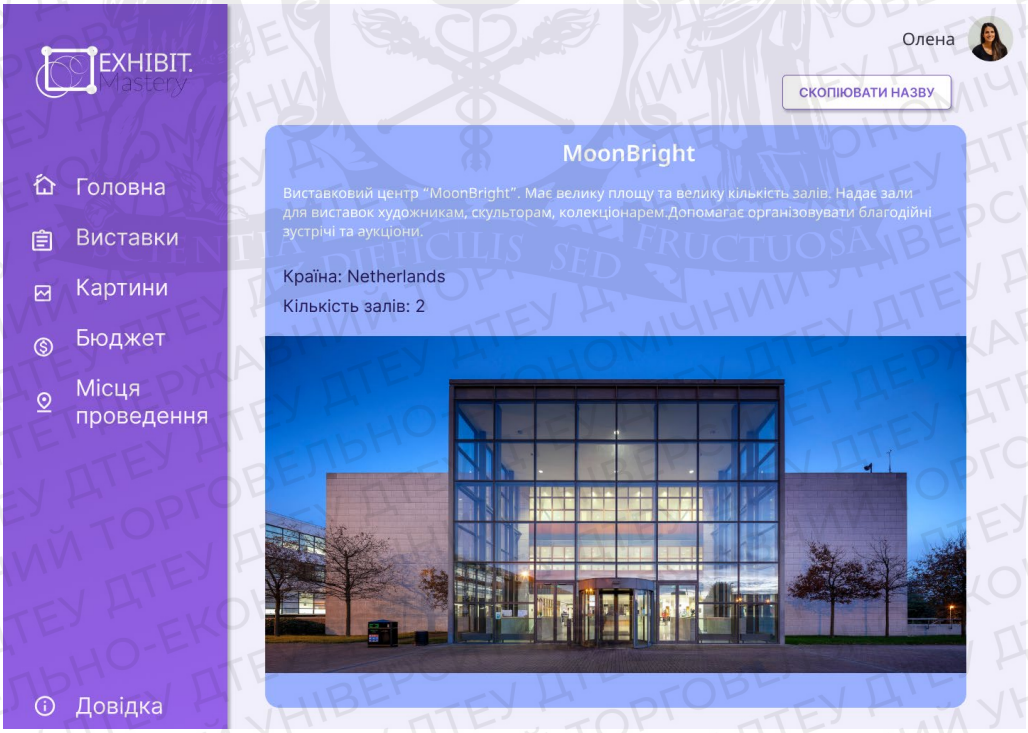

Рис.4 Приклад картки-місця з таблиці «Місця проведення»

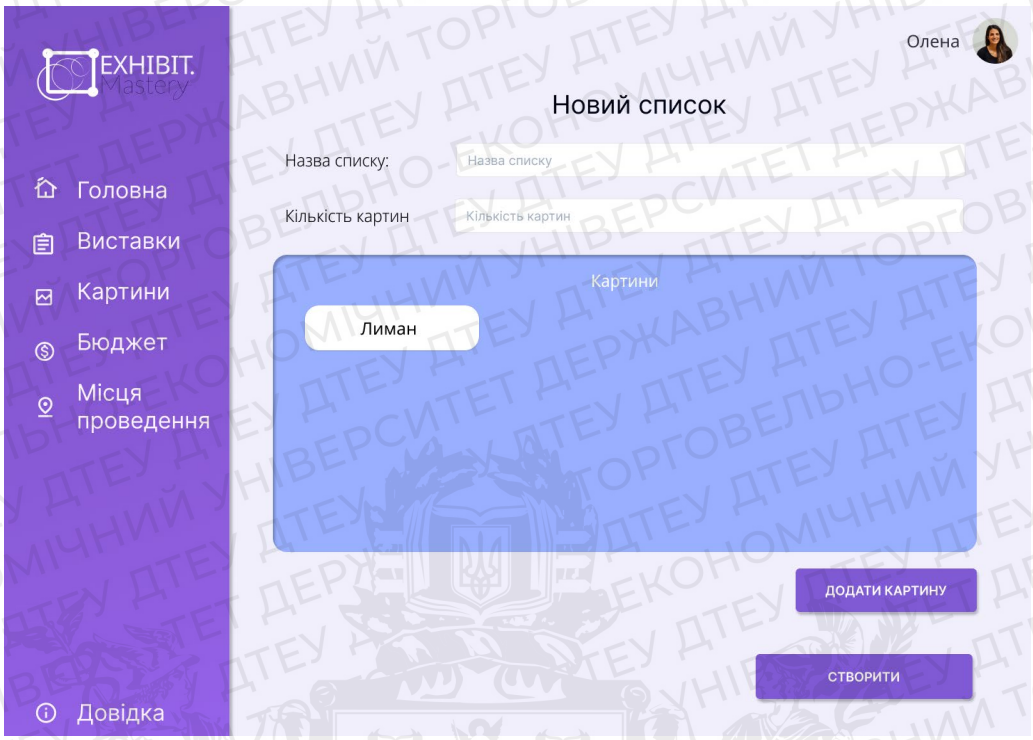

Рис.5 Форма для створення нового списку картин у панелі управління системою

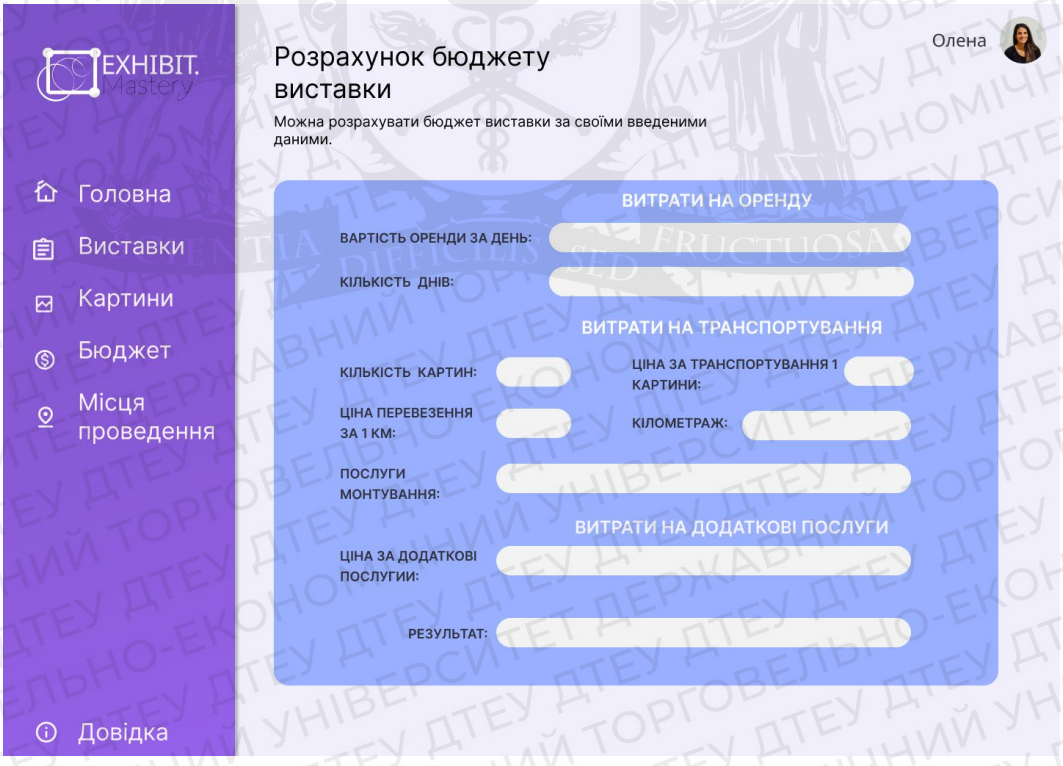

Рис.6 Сторінка «Бюджет» у панелі управління системою

Додаток Г

Код для розробки бази даних у MySQL

```
CREATE SCHEMA IF NOT EXISTS `exhibit_mastery` DEFAULT
CHARACTER SET utf8 ;
USE `exhibit_mastery` ;

CREATE TABLE IF NOT EXISTS `exhibit_mastery`.`userOrganizators` (
`user_Organizator_id` INT NOT NULL,
`name_Organizator` VARCHAR(45) NULL,
`lastname_Organizator` VARCHAR(45) NULL,
`nickname_Organizator` VARCHAR(45) NULL,
`activity_Organizator` VARCHAR(45) NULL,
`passwordOrganizator` VARCHAR(45) NULL,
`email_Organizator` VARCHAR(45) NULL,
PRIMARY KEY (`user_Organizator_id`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `exhibit_mastery`.`userOrganizations` (
`user_Organization_id` INT NOT NULL,
`name_company` VARCHAR(45) NULL,
`countryCompany` VARCHAR(45) NULL,
`activity_company` VARCHAR(45) NULL,
`password_Organization` VARCHAR(45) NULL,
`email_Organization` VARCHAR(45) NULL,
PRIMARY KEY (`user_Organization_id`))
ENGINE = InnoDB;


CREATE TABLE IF NOT EXISTS `exhibit_mastery`.`ListsOfArtworks` (
```

```
`ListsOfArtworks_id` INT NOT NULL,
`listNumber` INT NULL,
`nameOfList` VARCHAR(45) NULL,
`quantityOfArtworks` INT NULL,
PRIMARY KEY (`ListsOfArtworks_id`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `exhibit_mastery`.`placeForExhibition` (
`placeForExhibition_id` INT NOT NULL,
`nameOfPlaceExh` VARCHAR(45) NULL,
`counryOfPlaceExh` VARCHAR(45) NULL,
`addressOfPlaceExh` VARCHAR(45) NULL,
`quantityOfHalls` INT NULL,
PRIMARY KEY (`placeForExhibition_id`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `exhibit_mastery`.`Exhibitions` (
 `Exhibitions_id` INT NOT NULL,
 `nameExhib` VARCHAR(45) NULL,
 `descriptionExhib` VARCHAR(150) NULL,
 `typeExhib` VARCHAR(45) NULL,
 `dateExhib` DATE NULL,
 `userOrganizators_user_Organizator_id` INT NOT NULL,
 `userOrganizations_user_Organization_id` INT NOT NULL,
 `ListsOfArtworks_ListsOfArtworks_id` INT NOT NULL,
 `placeForExhibition_placeForExhibition_id` INT NOT NULL,
 PRIMARY KEY (`Exhibitions_id`, `userOrganizators_user_Organizator_id`,
`userOrganizations_user_Organization_id`,
`ListsOfArtworks_ListsOfArtworks_id`,
`placeForExhibition_placeForExhibition_id`),
```

```
  INDEX `fk_Exhibitions_userOrganizators_idx`
(`userOrganizators_user_Organizator_id` ASC) VISIBLE,
  INDEX `fk_Exhibitions_userOrganizations1_idx`
(`userOrganizations_user_Organization_id` ASC) VISIBLE,
  INDEX `fk_Exhibitions_ListsOfArtworks1_idx`
(`ListsOfArtworks_ListsOfArtworks_id` ASC) VISIBLE,
  INDEX `fk_Exhibitions_placeForExhibition1_idx`
(`placeForExhibition_placeForExhibition_id` ASC) VISIBLE,
  CONSTRAINT `fk_Exhibitions_userOrganizators`
    FOREIGN KEY (`userOrganizators_user_Organizator_id`)
    REFERENCES `exhibit_mastery`.`userOrganizators` (`user_Organizator_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Exhibitions_userOrganizations1`
    FOREIGN KEY (`userOrganizations_user_Organization_id`)
    REFERENCES `exhibit_mastery`.`userOrganizations` (`user_Organization_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Exhibitions_ListsOfArtworks1`
    FOREIGN KEY (`ListsOfArtworks_ListsOfArtworks_id`)
    REFERENCES `exhibit_mastery`.`ListsOfArtworks` (`ListsOfArtworks_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Exhibitions_placeForExhibition1`
    FOREIGN KEY (`placeForExhibition_placeForExhibition_id`)
    REFERENCES `exhibit_mastery`.`placeForExhibition`
(`placeForExhibition_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
CREATE TABLE IF NOT EXISTS `exhibit_mastery`.`authorOfArtwork` (
  `authorOfArtwork_id` INT NOT NULL,
  `authorName` VARCHAR(45) NULL,
  `authorLastname` VARCHAR(45) NULL,
  `authorNickname` VARCHAR(45) NULL,
  `authorCountry` VARCHAR(45) NULL,
  `phoneNumber` INT(13) NULL,
  PRIMARY KEY (`authorOfArtwork_id`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `exhibit_mastery`.`Artwork` (
  `Artwork(Painting)_id` INT NOT NULL,
  `nameAW` VARCHAR(45) NULL,
  `numberAW` VARCHAR(45) NULL,
  `imgAW` VARCHAR(45) NULL,
  `descriptionAW` VARCHAR(200) NULL,
  `widthAW` INT NULL,
  `lengthAW` INT NULL,
  `weightAW` INT NULL,
  `authorOfArtwork_authorOfArtwork_id` INT NOT NULL,
  PRIMARY KEY (`Artwork(Painting)_id`,
`authorOfArtwork_authorOfArtwork_id`),
  INDEX `fk_Artwork_authorOfArtwork1_idx`
(`authorOfArtwork_authorOfArtwork_id` ASC) VISIBLE,
  CONSTRAINT `fk_Artwork_authorOfArtwork1`
    FOREIGN KEY (`authorOfArtwork_authorOfArtwork_id`)
    REFERENCES `exhibit_mastery`.`authorOfArtwork` (`authorOfArtwork_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

```
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `exhibit_mastery`.`ListOfPlaces` (
  `ListOfPlaces_id` INT NOT NULL,
  `nameOfPlace` VARCHAR(45) NULL,
  `country` VARCHAR(45) NULL,
  PRIMARY KEY (`ListOfPlaces_id`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `exhibit_mastery`.`Hall` (
  `Hall_id` INT NOT NULL,
  `hallNumber` INT NULL,
  `sizeOfArea` INT NULL,
  `hallName` VARCHAR(45) NULL,
  `placeForExhibition_placeForExhibition_id` INT NOT NULL,
  PRIMARY KEY (`Hall_id`, `placeForExhibition_placeForExhibition_id`),
  INDEX `fk_Hall_placeForExhibition1_idx`
(`placeForExhibition_placeForExhibition_id` ASC) VISIBLE,
  CONSTRAINT `fk_Hall_placeForExhibition1`
    FOREIGN KEY (`placeForExhibition_placeForExhibition_id`)
    REFERENCES `exhibit_mastery`.`placeForExhibition`
(`placeForExhibition_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


CREATE TABLE IF NOT EXISTS
`exhibit_mastery`.`ListsOfArtworks_has_Artwork` (
  `ListsOfArtworks_ListsOfArtworks_id` INT NOT NULL,
```

```
  `Artwork_Artwork(Painting)_id` INT NOT NULL,
  `ListsOfArtworks_has_Artwork_id` INT NOT NULL,
  PRIMARY KEY (`ListsOfArtworks_ListsOfArtworks_id`,
`Artwork_Artwork(Painting)_id`, `ListsOfArtworks_has_Artwork_id`),
  INDEX `fk_ListsOfArtworks_has_Artwork_Artwork1_idx`
(`Artwork_Artwork(Painting)_id` ASC) VISIBLE,
  INDEX `fk_ListsOfArtworks_has_Artwork_ListsOfArtworks1_idx`
(`ListsOfArtworks_ListsOfArtworks_id` ASC) VISIBLE,
  CONSTRAINT `fk_ListsOfArtworks_has_Artwork_ListsOfArtworks1`
    FOREIGN KEY (`ListsOfArtworks_ListsOfArtworks_id`)
    REFERENCES `exhibit_mastery`.`ListsOfArtworks` (`ListsOfArtworks_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_ListsOfArtworks_has_Artwork_Artwork1`
    FOREIGN KEY (`Artwork_Artwork(Painting)_id`)
    REFERENCES `exhibit_mastery`.`Artwork` (`Artwork(Painting)_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

Додаток Д

Текст програми

Весь програмний код, який включає усі модулі, має дуже великий обсяг, тому наводити його повністю недоцільно. Для прикладу наведено код для реалізації панелі навігації, реєстрації, та код для реалізації чек-листа з файлів.

Код для створення панелі навігації у дешборді з системою файл Sidebar.tsx:

```
import React from 'react';
import Link from 'next/link';
import Image from 'next/image';
import Icon from '@mdi/react';
import { MdSettings } from 'react-icons/md';

export const Sidebar = () => {
 return (
  <div className="flex">
   <div className="fixed bg-vioex-medium  w-200 h-screen p-4 shadow-lg
flex flex-col justify-between">
    <main className="w-full">
     <Image
      src={'./assets/white-logo.svg'}
      alt={'White logo'}
      width={240}
      height={115}
     ></Image>
     <div className="">
      <div className="flex flex-row justify-center p-2 m-2 items-center pl-
2 text-white">
       <Image
        className="flex mr-4"
        src={'./assets/icon-home.svg'}
        alt={'Home icon'}
        width={30}
        height={30}
       ></Image>
       <Link href={'/'}>Головна</Link>
      </div>
      <div className="flex flex-row justify-center items-center pl-2 text-
white">
       <Image
        className="flex mr-4"
        src={'./assets/icon-exhibitions.svg'}
```

```
            alt={'Exhibitions icon'}
            width={25}
            height={25}
          ></Image>
          <Link href={'/exhibitions'}>Виставки</Link>
        </div>
        <div className="flex flex-row justify-center pt-4 pr-1 items-center
pl-2 text-white">
          <Image
            className="flex mr-4"
            src={'./assets/icon-lists.svg'}
            alt={'Pictures icon'}
            width={25}
            height={25}
          ></Image>
          <Link href={'/pictures'}>Картини</Link>
        </div>
        <div className="flex flex-row justify-center pt-4 pr-3 items-center
pl-2 text-white">
          <Image
            className="flex mr-4"
            src={'./assets/icon-money.svg'}
            alt={'Budget icon'}
            width={30}
            height={30}
          ></Image>
          <Link href={'/budget'}>Бюджет</Link>
        </div>
        <div className="flex flex-row justify-center pt-2 pl-7 items-center
pl-2 text-white">
          <Image
            className="flex mr-4"
            src={'./assets/icon-places.svg'}
            alt={'Places icon'}
            width={30}
            height={30}
          ></Image>
          <Link href={'/places'}>
            Місця <br></br>проведення
          </Link>
        </div>
        <div className='flex flex-row justify-center pt-20 pr-3 items-center
pl-2 text-white'>
          <Image
```

```
                    className="flex mr-4"
                    src={'./assets/icon-info.svg'}
                    alt={'Home icon'}
                    width={30}
                    height={30}
                   ></Image>
                   <Link href={'/information'}>Довідка</Link>
                  </div>
                 </div>
                </main>
               </div>
              </div>
             );
            };
export default Sidebar;
```

Файл AddTask.tsx

```
import { Button } from '@chakra-ui/react';
import { AiOutlinePlus } from 'react-icons/ai';
import { FaTrash } from 'react-icons/fa';
import Modal from './Modal';
import { FormEventHandler, useState } from 'react';
import { useRouter } from 'next/navigation';
import { addTodo } from '../../../api';
import { v4 as uuidv4 } from 'uuid';

const AddTask = () => {
 const router = useRouter;
 const [modalOpen, setModalOpen] = useState<boolean>(false);
 const [newTaskValue, setNewTaskValue] = useState<string>('');

 const handleSubmitNewTodo: FormEventHandler<HTMLFormElement> =
async (e) => {
  e.preventDefault();
  console.log(newTaskValue);
  await addTodo({
   id: uuidv4(),
   text: newTaskValue,
  });
  setNewTaskValue('');
 };
 return (
  <div className="flex mb-4 justify-center">
   <Button
    onClick={() => setModalOpen(true)}
```

```
        className="text-blupan-texti m-2 bg-blupan-panel rounded-[60px]
shadow-pan-1 "
      >
        <AiOutlinePlus className="" size={18} />
      </Button>
      <Modal modalOpen={modalOpen} setModalOpen={setModalOpen}>
        <form onSubmit={handleSubmitNewTodo}>
          <h3 className="font-bold text-lg">Create task</h3>
          <div className="font-bold">
          <input
            value={newTaskValue}
            onChange={(e) => setNewTaskValue(e.target.value)}
            type="text"
            placeholder="placeholders"
            className="input input boardered w-full w-full"
          />
          <button type="submit" className="btn">
            Створити
          </button>
          </div>
        </form>
      </Modal>

      <Button className="text-blupan-texti m-2 align-left bg-blupan-panel
rounded-[60px] shadow-pan-1">
        <FaTrash />
      </Button>
    </div>
  );
};

export default AddTask;
```

З повним кодом веб-застосунку, який включає усі файли проекту, можна ознайомитися на веб-сервісі GitHub:https://github.com/akkimiho/Exhibit.Mastery-app.git