

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розроблення інформаційної системи картинної галереї

Виконав: студент групи 4147ст

 Єфремова С. О.
(підпис)

Керівник роботи:

(посада, науковий ступінь вчене звання)

 Беркунська Н. М.
(підпис)

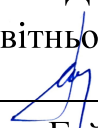
м. Миколаїв — 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 122 "Комп'ютерні науки"
Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми


к.т.н, доц. Гайда А.Ю.

" ____ " _____ 2025 р.

*ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ*
на здобуття ступеня вищої освіти "бакалавр"

Студенту Єфремовій Софії Олегівні

(прізвище, ім'я, по батькові)

1. Тема роботи: Інформаційна система картинної галереї

Керівник роботи Беркунська Наталя Михайлівна

Затверджені наказом ректора № 295-уч від " 27 " березня 2025 року.

2. Термін подання роботи: 20 червня 2025 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем,

4. Перелік питань, що належать до розробки (найменування розділів)

Розробка концепції інформаційної системи, розробка технічного проекту інформаційної системи, реалізація проекту інформаційної системи, охорона праці.

5. Перелік презентаційних матеріалів

Розробка інформаційної системи картинна галерея; Актуальність, ціль, завдання; Організаційна структура картинної галереї; Концепція системи; технічний проект; ER-діаграма; Логічна модель БД; Фізична модель БД; Використані технології; Розгортання системи; Реалізація інформаційної системи;с Результати впровадження.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Беркунська Н. М.	03.02.2025	22.02.2025
2	Беркунська Н. М.	22.02.2025	21.03.2025
3	Беркунська Н. М.	21.03.2025	25.04.2025
4	Беркунська Н. М.	25.04.2025	07.06.2025

7. Дата видачі завдання 03 лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	13.02.2025	
2	Постановка задачі	15.02.2025	
3	Розробка концепції	22.02.2025	
4	Розробка проекту ІС	21.03.2025	
5	Реалізація проекту	25.04.2025	
6	Розробка питань з охорони праці	07.06.2025	
7	Розробка графічних матеріалів	12.06.2025	
8	Подання роботи на перевірку на плагіат	10.06.2025	
9	Подання роботи рецензенту	15.06.2025	
10	Подання роботи на захист	20.06.2025	

Студент


(підпис)

Єфремова С. О.

(ПІБ)

Керівник роботи


(підпис)

Беркунська Н. М.

(ПІБ)

АННОТАЦІЯ

Дана кваліфікаційна робота присвячена дослідженню та розробці інформаційної системи «Картина галерея», яка слугує інструментом для автоматизації процесів управління, зберігання даних та презентації інформації про мистецькі об'єкти. У роботі представлено аналіз предметної області, вимоги до системи, концепція її функціонування та технічний проект. Застосування запропонованих рішень спрямоване на оптимізацію роботи художніх галерей, поліпшення доступу до мистецьких колекцій та створення зручного цифрового середовища для взаємодії користувачів.

Робота викладена на 81 сторінках машинописного тексту, містить 20 рисунків, 1 таблицю, 3 додатки, список з 20 використаних джерел.

Робота виконана українською мовою.

ANNOTATION

This qualification work is devoted to the research and development of the information system "Picture Gallery", which serves as a tool for automating the processes of management, data storage and presentation of information about art objects. The work presents an analysis of the subject area, requirements for the system, the concept of its functioning and a technical project. The application of the proposed solutions is aimed at optimizing the work of art galleries, improving access to art collections and creating a convenient digital environment for user interaction.

The work is presented on 81 pages of typewritten text, contains 20 figures, 1 table, 3 appendices, a list of 20 sources used.

The work is done in Ukrainian.

Ключові слова

БД — База даних, ІС — інформаційна система, картинна галерея, сутність, атрибути, Hibernate — інструмент, що спрощує роботу з базами даних, Spring Data JPA — інтерфейс для зв'язку з сутностями бази даних, MySQL — система керування реляційними базами даних.

ЗМІСТ

Ключові слова.....	6
Вступ.....	8
Розділ 1 Розробка концепції інформаційної системи картина галерея.....	9
1.1 Опис об'єкту дослідження і аналіз предметної області.....	9
1.2 Формування вимог до інформаційної системи.....	14
1.3 Розробка концепції інформаційної системи.....	17
Розділ 2 Розробка технічного проекту інформаційної системи картинна галерея.....	20
2.1 Загальносистемні рішення.....	20
2.2 Рішення по інформаційному забезпеченню.....	21
2.2.1 Розробка концептуальної моделі даних.....	23
2.2.2 Розробка логічної моделі даних.....	26
2.2.3 Розроблення фізичної моделі даних.....	29
2.3 Рішення з програмного забезпечення.....	35
2.4 Рішення з технічного забезпечення.....	40
Розділ 3 Реалізація проекту інформаційної системи картинної галереї.....	43
Розділ 4 Охорона праці.....	45
Висновки.....	48
Список використаних джерел.....	49
Додаток А.....	52
Додаток Б.....	59
Додаток В.....	79

ВСТУП

Робота присвячена автоматизації процесів роботи картинної галереї через створення інформаційної системи.

Сучасний світ переповнений інформацією, тому виникає потреба в її структуризації. Для зручної організації користуються різноманітними технологіями, в тому числі веб-застосунки.

Головною перевагою використання баз даних є швидкість внесення, коригування та використання необхідної користувачу інформації. БД гарантують цілісність і зв'язок інформації, тобто при зміні або видалені даних в одному рядку таблиці дані в інших таблицях також оновляться. Тому використання БД полегшує роботу з інформацією. Бази даних також дозволяють вирішити проблеми зберігання та систематизації інформації згідно індивідуальних вимог компаній, корпорацій або окремих юридичних осіб.

Актуальність написання цієї роботи полягає в тому, що в наш час дуже важливим є автоматизація та комп'ютеризація процесів для більш ефективної роботи та функціонування будь-якого підприємства або організації, в тому числі й картинних галерей. Сучасні музеї суттєво відрізняються від колишніх, адже вони оснащені новітньою технікою та мають доступ до інтернету.

РОЗДІЛ 1 РОЗРОБКА КОНЦЕПЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ КАРТИНА ГАЛЕРЕЯ

1.1 Опис об'єкту дослідження і аналіз предметної області

Картинна галерея являє собою культурну інституцію, основною функцією якої є збереження, наукове дослідження, експонування та популяризація творів образотворчого мистецтва. Діяльність галереї може включати як демонстрацію постійної колекції, так і організацію тимчасових виставок, спрямованих на ознайомлення аудиторії з мистецтвом різних історичних періодів, художніх напрямів та індивідуальних творчих практик митців. [1].

Картинні галереї здійснюють активну взаємодію з соціумом, застосовуючи різноманітні формати комунікації. Зокрема, вони ініціюють масштабні культурні події, такі як «ночі в музеях», міжнародні виставкові проекти та тематичні мистецькі форуми. У контексті співпраці з освітніми інституціями реалізуються спеціалізовані програми для учнівської та студентської молоді, що передбачають ознайомлення з музейними колекціями, інтерактивні лекційні курси та практичні творчі сесії. Крім того, галереї можуть надавати можливості для становлення молодих митців, забезпечуючи платформи для експонування їхніх творів та фахового обговорення [2].

У контексті сучасного суспільства картинні галереї виконують значущу роль, забезпечуючи доступ до культурної спадщини, сприяючи формуванню естетичних уподобань, стимулюючи креативний розвиток особистості та слугуючи платформами для комунікації між мистецтвознавцями, художниками та шанувальниками мистецтва. Окрім того, вони реалізують науково-дослідницьку та освітню функції, пропонуючи різноманітні екскурсійні програми, лекційні курси, майстер-класи та інтерактивні заходи для різних категорій відвідувачів [2].

Результати наукових досліджень підтверджують значну користь відвідування музеїв та картинних галерей. На противагу перегляду фотокопій в інтернет-середовищі, що обмежує сприйняття площинним зображенням з фіксованої перспективи, безпосереднє ознайомлення з оригінальними творами мистецтва активізує когнітивні процеси. Мозок, у ситуації фізичної присутності, стимулюється до отримання більш повної інформації. Відвідувач отримує можливість наблизитися до об'єкта дослідження, обрати оптимальну точку огляду з урахуванням освітлення, детально вивчити нюанси живописної техніки, зокрема мазки та фактуру [2].

Діяльність картинної галереї охоплює кілька ключових аспектів. По-перше, це комплектування фондів. Картини можуть надходити до галереї різними шляхами: через державні закупівлі, приватні пожертви, міжнародний обмін або придбання меценатами. Важливим завданням установи є створення належних умов для збереження експонатів, що включає підтримку оптимальної температури, вологості та освітлення, проведення реставраційних робіт та страхування колекцій. По-друге, це виставкова діяльність. Галерея формує постійні експозиції, організовує тимчасові виставки та тематичні заходи. Менеджери працюють над концепціями виставок, розробляють логіку розміщення експонатів, створюють інформаційні матеріали та займаються популяризацією подій [1].

Розглянемо організаційну структуру, яка приведена на рис. 1.1.



Рисунок 1.1 — Організаційна структура картинної галереї

Директор галереї відповідає за стратегічний розвиток, фінансове управління, співпрацю з державними та приватними партнерами. Також директор здійснює керівництво самою галереєю, приймає на роботу та звільняє заступників, керівників філій. Директор виступає як представник закладу та несе відповідальність за його роботу [3]. Під керівництвом директора знаходяться заступник директора, менеджер з маркетингу, менеджер інформаційних технологій, а також головний бухгалтер.

Менеджер з маркетингу займається розробкою маркетингової стратегії галереї, створює рекламні кампанії, організовує співпрацю зі спонсорами та партнерами. Він аналізує ринок, проводить дослідження відвідуваності та впроваджує заходи для підвищення популярності виставок та заходів [3].

Менеджер з інформаційних технологій відповідає за сервісну підтримку та офісні застосунки закладу.

Заступник директора виконує адміністративні та організаційні функції, допомагає генеральному директору в управлінні галереєю, координує діяльність підрозділів і виконання стратегічних завдань. Він відповідає за внутрішню політику установи, взаємодію з партнерами та вирішення поточних питань управління. Під його розпорядженням знаходяться адміністратор кадрів, менеджер з освітніх програм, реставратори та завідувач господарством [3].

Завідувач господарством відповідає за технічне та господарське обслуговування галереї, контроль за чистотою, безпекою та станом приміщень. Він керує роботою обслуговуючого персоналу (робітників, охорони, прибиральників), організовує закупівлю матеріалів, забезпечує ремонтні роботи та взаємодіє з підрядниками..

Менеджер з освітніх програм координує екскурсії, лекції та спеціальні заходи, спрямовані на залучення нової аудиторії. Він займається підбором експонатів, розробкою тематичних концепцій та організацією експозицій. Заступник директора також керує реставраційними роботами. Екскурсоводи та лектори забезпечують проведення ознайомчих програм, а технічний персонал, зокрема реставратори, відповідає за збереження експонатів.

Адміністратор кадрів займається веденням кадрової документації, прийомом і звільненням працівників, обліком робочого часу та організацією навчання персоналу. Він також забезпечує дотримання трудового законодавства та взаємодіє з керівництвом щодо управління штатом.

Задачею головного бухгалтера є ведення бухгалтерської звітності картинної галереї, контроль фінансових операцій та дотримання податкового законодавства. Він відповідає за облік доходів і витрат, підготовку фінансових звітів, управління бюджетом установи, а також співпрацю з податковими органами та аудиторами.

Заступник головного бухгалтера допомагає в управлінні фінансовими потоками галереї, контролює правильність ведення бухгалтерських операцій та забезпечує своєчасну підготовку звітності. Він координує роботу бухгалтерів, відповідає за

розрахунок заробітної плати працівникам, ведення банківських операцій та контроль за дотриманням фінансової дисципліни.

Економіст займається плануванням фінансової діяльності галереї, аналізує доходи та витрати, формує бюджети та займається розрахунками ефективності витрат. Він також прогнозує фінансові ризики, проводить аналіз рентабельності виставок і проєктів, а також розробляє економічні стратегії для покращення фінансової стабільності установи.

Касири відповідають за продаж квитків до галереї.

Провівши аналіз предметної області обробки інформації в галереї бачимо, що вона виконується вручну, хоча є відповідні засоби та техніка для автоматизації процесів. Бухгалтерські та фінансові задачі вирішує впроваджена система стороннього розробника, але вона не враховує специфічну роботу картинної галереї, а саме: організація виставок, екскурсій, реставрацій тощо.

Ці та інші функціональні задачі, які дозволяють виконувати управління картинною галереєю, як от управління картинами, виставками або реставраціями — відсутні. У сучасному світі, де цифрові технології стають основою для ефективного управління процесами, використання застарілих методів звітності та адміністративної діяльності в картинних галереях може призводити до значних проблем. Облік експонатів, їх переміщення, стан та інформація про виставки зберігаються у вигляді електронних таблиць та Google документів, що підвищує ризик втрати важливих даних через людський фактор, фізичне пошкодження або їх несанкціоновану заміну. Відсутність централізованої цифрової бази даних ускладнює пошук необхідної інформації, що може затримувати робочі процеси та знижувати ефективність роботи працівників [4].

Ще однією проблемою є управління виставками та експозиціями. А саме, складність у контролі та плануванні заходів, розміщення картин у виставкових залах. Наприклад, якщо картина переміщується з однієї зали в іншу або відправляється на реставрацію, відсутність електронної системи може спричинити плутанину та помилки в

розташуванні творів. Крім того, можуть виникнути проблеми з оновленням даних, що в свою чергу призведе до неточностей у документації.

Окрему проблему становить комунікація між співробітниками та обмін інформацією. В умовах неавтоматизованого документообігу передача завдань, узгодження виставкових планів та координація між відділами займають значно більше часу, що може сповільнювати виконання робочих процесів. Відсутність автоматизованих звітів та системи аналітики також обмежує можливості адміністрації щодо оцінки роботи галереї та планування її подальшого розвитку.

На основі проведеного аналізу необхідно провести розробку інформаційної системи для картинної галереї. Ця система покликана забезпечити автоматизацію процесів обліку, управління експозиціями, моніторингу стану картин, а також покращити внутрішню комунікацію між співробітниками. Використання технологій дозволить зменшити ризик втрати важливих даних, оптимізувати робочий процес та підвищити ефективність діяльності галереї в умовах цифрового середовища [5].

1.2 Формування вимог до інформаційної системи

На основі прийнятого рішення щодо створення інформаційної системи сформовано вимоги. Основними критеріями якості є зручність інтерфейсу користувача, швидкодія, надійність та безпека даних.

Функціональні вимоги

Створення веб-сайту для представлення користувачам відповідної інформації про картинну галерею, а саме:

- Можливість управління та ведення каталогу картин: додавання, редагування, видалення записів про картини; збереження основних даних про картини (назва, рік створення, техніка виконання, художник, постачальник).

- Управління інформацією про художників і постачальників: ведення бази художників із можливістю редагування, ведення списку постачальників з даними для подальшого зв'язку з ними.
- Організація виставок: додавання, редагування і видалення виставок, також важливим завданням є призначення картин до конкретних виставок із можливістю вказати період експонування.
- Контроль стану експонатів: збереження інформації про технічний стан картин (потреба в реставрації) та ведення журналу реставрацій для картин, які роботи виконувались та коли.
- Передбачити в майбутньому можливість додавання інтерфейсу з діючим комплексом бухгалтерського обліку та звітності.

Сервісні засоби

Продуктивність системи визначається як здатність до швидкої обробки запитів, що є критично важливим для ефективної взаємодії з потенційно великими обсягами даних. Масштабованість забезпечує можливість подальшого розширення функціоналу системи без істотних модифікацій її базової архітектури. Зручність у підтримці досягається за рахунок проектування системи з урахуванням простоти внесення змін та оновлень.

Функціонування системи передбачає обов'язкове підключення до мережі Інтернет як для клієнтської (через веб-браузер), так і для серверної частин, забезпечуючи обмін даними та віддалений доступ користувачів.

Забезпечення безпеки даних є ключовим аспектом розробки, що включає використання протоколів шифрування (HTTPS) для передачі даних, контроль доступу до бази даних, регулярне резервне копіювання та розмежування прав користувачів для гарантування цілісності та конфіденційності інформації.

Вимоги по програмному забезпеченню

При розробці використано об'єктно-орієнтований підхід, що відповідає вимогам проекту та сприяє покращенню модульності, розширюваності та підтримки коду. Мовою програмування обрано — Java, тому що вона не тільки забезпечує високу рівень переносимості, а й має в собі широкий спектр функціональних можливостей [1].

Компілятор та середовище розробки: IntelliJ IDEA — має багато інструментів для розробки на Java та забезпечує зручне середовище для програмістів.

Для ефективного збереження, обробки та доступу до даних у системі використовується система управління базами даних MySQL. Вона дозволяє забезпечити цілісність, швидкий доступ та багатокористувацьку роботу з даними галереї [7].

Веб-застосунок функціонує як комп'ютерна інформаційна система, яка реалізує логіку взаємодії між користувачем та базою даних через веб-інтерфейс. Для повноцінної роботи системи необхідна наявність підключення до мережі Інтернет, оскільки всі запити до серверної частини здійснюються в онлайн-режимі.

Користувачі взаємодіють з системою за допомогою графічного веб-інтерфейсу, який забезпечує зручну навігацію та інтуїтивне управління даними. Система підтримується через Інтернет, що дозволяє отримувати доступ до даних з будь-якого пристрою (комп'ютера, ноутбука, планшета), маючи відповідні права доступу. Таким чином, адміністратори, реставратори, куратори та інші працівники галереї можуть у реальному часі працювати з актуальною інформацією.

Для реалізації інформаційної системи використані різноманітні інструменти.

Функціональні засоби:

- Мова програмування — Java (JDK21, Spring Framework) [11];
- мова гіпертекстової розмітки — HTML [12];
- мова каскадних таблиць стилів — CSS;
- система управління базами даних — MySQL (8.3).

Сервісні програмні засоби:

- Програмне забезпечення для роботи з текстовими файлами — Word;
- мережа, для онлайн доступу до сайту — Інтернет;
- програмне забезпечення для створення діаграм та схем — draw.io.

Технічні засоби проекту

2 принтери для ч/б друку Pantum P2500NW, 5 комп'ютерів, наприклад одноплатний комп'ютер Raspberry Pi 4, з доступом в Інтернет та наступними характеристиками:

- Windows 10-11/64-bit;
- RAM не менше 8096 Мб;
- Клавіатура;
- Миша;
- Монітор;
- Жорсткий диск (рекомендовано SSD) не менше 256 Гб.

Зважаючи на всі перелічені вимоги буде виконуватись розробка концепції інформаційної системи.

1.3 Розробка концепції інформаційної системи

На основі сформульованих вимог розроблено концепцію, яка представлена на рис. 1.2.

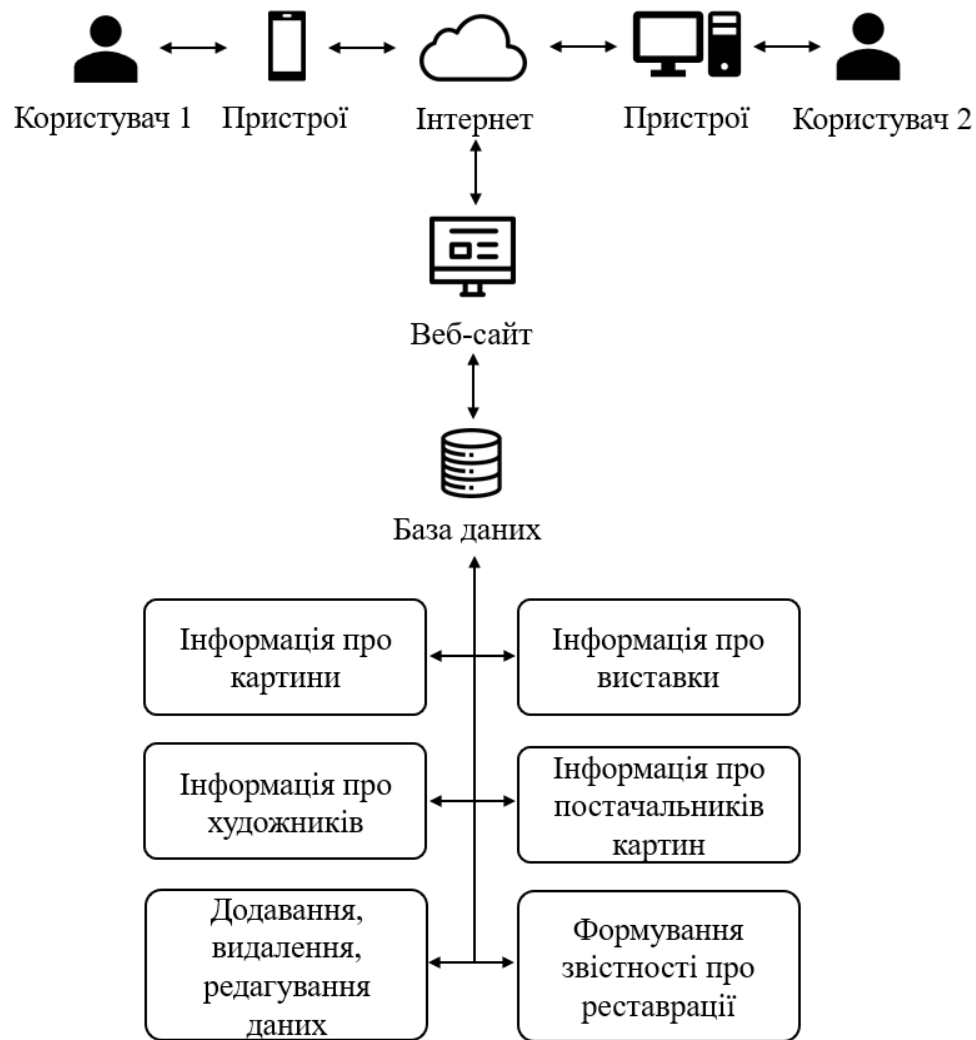


Рисунок 1.2 — Концептуальна модель інформаційної системи

Інформаційна система картинної галереї розроблена для зручного обліку, управління та обробки даних, що стосуються основних аспектів функціонування галереї. Основним елементом цієї системи є централізована база даних, у якій зберігається структурована інформація про всі ключові компоненти галереї: картини, художників, виставки, постачальників, а також звітність щодо реставрацій. Взаємодія з цією базою даних забезпечує швидкий доступ до інформації та її оперативне оновлення.

Кожен запис у базі даних має свою категорію. Наприклад, інформація про картини включає назву, автора, техніку виконання, дату створення тощо. Інформація про

художників охоплює біографічні дані, стиль, наявні роботи в колекції галереї. Інформація про виставки містить дані про час, місце проведення, представлені експонати, учасників та партнерів. Також система веде облік постачальників, які забезпечують матеріали, обладнання або інші ресурси, необхідні для діяльності галереї.

Функціонал системи передбачає додавання, редагування та видалення даних, що дозволяє гнучко та ефективно керувати інформацією. У разі зміни інформації дані можна швидко оновити. Система також дає можливість формувати звітність про проведені реставраційні роботи, включаючи інформацію про стан картини, обсяг виконаних робіт, відповідальних працівників тощо.

Висновок

На основі розробленої концепції інформаційної системи розроблюється технічний проект інформаційної системи, який досягне поставленої мети підвищення ефективності роботи картинної галереї.

РОЗДІЛ 2 РОЗРОБКА ТЕХНІЧНОГО ПРОЕКТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ КАРТИННА ГАЛЕРЕЯ

2.1 Загальносистемні рішення

Загальносистемні рішення інформаційної системи «Картинна галерея» — це комплекс універсальних, багаторазово використовуваних компонентів та підходів, які забезпечують функціонування інформаційної системи в цілому, незалежно від її конкретних предметних функцій (таких як облік експонатів, організація виставок, реставрацій тощо). Ці рішення є фундаментом для побудови системи і забезпечують їхню інтеграцію та ефективну взаємодію.

До основних категорій загальносистемних рішень для інформаційної системи картинної галереї можуть належати:

Технічна інфраструктура

- Апаратне забезпечення, що включає сервери для зберігання даних та організації роботи програмного забезпечення, комп'ютери для персоналу, мережеве обладнання для зв'язку між компонентами системи, обладнання для резервного копіювання та відновлення даних.

- Системне програмне забезпечення: операційні системи серверів та робочих станцій, системи управління базами даних (СУБД), засоби адміністрування та моніторингу системи, засоби забезпечення інформаційної безпеки (антивірусне програмне забезпечення, між мережеві екрани тощо).

- Мережева інфраструктура, тобто локальна мережа (LAN) для внутрішнього зв'язку, підключення до Інтернету для зовнішньої взаємодії.

Забезпечення інформаційної безпеки

- Шифрування даних для захисту конфіденційної інформації під час зберігання та передачі.

- Ведення записів про дії користувачів та системні події для відстеження та аналізу.

- Резервне копіювання — створення копій важливих даних для запобігання їх втраті та забезпечення можливості відновлення у разі збоїв.

Управління даними

- Розробка загальної структури бази даних, визначення сутностей, їхніх атрибутів та взаємозв'язків.

- Інструменти інтеграції даних, що включає в себе засоби для обміну даними між різними підсистемами та зовнішніми системами.

- Забезпечення якості даних. Процедури контролю цілісності, достовірності та актуальності інформації.

Загальні сервіси та інструменти

- Система документообігу для управління внутрішньою документацією галереї.

- Засоби звітності та аналітики — загальні інструменти для формування звітів на основі даних з різних підсистем.

Отже, загальносистемні рішення є критично важливими для забезпечення стабільності, надійності, безпеки та масштабованості інформаційної системи картинної галереї. Їхнє якісне проектування та впровадження закладає основу для ефективної роботи системи.

2.2 Рішення по інформаційному забезпеченню

У процесі розробки даної інформаційної системи було застосовано парадигму об'єктно-орієнтованого програмування, що є оптимальним рішенням для забезпечення високого рівня модульності, масштабованості та супроводжуваності кодової

бази відповідно до вимог проєкту. Вибір мови програмування Java зумовлений не лише її кросплатформністю, що забезпечує незалежність від конкретної апаратної архітектури, але й наявністю широкого спектра бібліотек і фреймворків, що сприяє прискоренню та оптимізації процесу розробки [10].

В якості інтегрованого середовища розробки (IDE) використовувався IntelliJ IDEA, який надає розширений набір інструментів для програмування на Java, включаючи інтелектуальне автодоповнення коду, рефакторинг, налагодження, що значно підвищує продуктивність розробки [10].

Для організації ефективного зберігання, обробки та доступу до структурованих даних застосовано реляційну систему управління базами даних MySQL. Дана СУБД характеризується надійністю, високою продуктивністю при обробці запитів, підтримкою транзакційності та механізмів забезпечення цілісності даних, що є критично важливим для забезпечення коректної та безперебійної роботи інформаційної системи картинної галереї в умовах багатокористувацького доступу [7].

Розроблений веб-застосунок функціонує як комп'ютерна інформаційна система, що реалізує бізнес-логіку взаємодії між кінцевим користувачем та базою даних через інтуїтивно зрозумілий веб-інтерфейс. В контексті розробки серверної частини застосовано фреймворк Spring Boot, який суттєво спрощує процес конфігурації та розгортання Java-застосунків, надаючи готові рішення для управління залежностями, маршрутизації запитів та інтеграції з іншими сервісами [9].

Для повноцінного функціонування системи необхідне стабільне підключення до мережі Інтернет, оскільки обмін даними між клієнтською та серверною частинами здійснюється в режимі реального часу за протоколами мережі.

У результаті реалізації обраних рішень було обрано ефективне інформаційне забезпечення системи, що включає стабільну взаємодію між користувачем та базою

даних через веб-інтерфейс. Використані технології сприяють надійній роботі системи в умовах багатокористувацького доступу та полегшують її подальший розвиток.

2.2.1 Розробка концептуальної моделі даних

Концептуальна модель визначається як абстрактне та структуроване представлення сутності даних, що охоплює ключові поняття, їхні атрибути та взаємозв'язки в межах досліджуваної предметної області. Процес розробки даної моделі ґрунтується на ретельному аналізі предметної області, що включає виявлення та систематизацію релевантної інформації про об'єкти, процеси та їхні характеристики. Отримана в результаті аналізу інформація слугує основою для формування цілісного відображення даних, яке є важливим етапом у процесі проектування інформаційних систем та баз даних [7].

Концептуальна модель являє собою фундаментальний компонент у розробці баз даних, оскільки її призначення полягає у забезпеченні зручності роботи з інформацією та її логічній організації. Наявність структурованої моделі є необхідною умовою для ефективного виконання запитів користувачів, формування звітних документів, розробки інтерфейсних форм тощо. На даному етапі розробки першочерговим завданням постає визначення ключових сутностей предметної області та встановлення типу їхніх зв'язків [7].

Після проведення аналізу предметної області галереї було виділено ключові сутності, які формують інформаційне середовище майбутньої системи. До них належать: Картина, Жанр, Напрямок, Виставка, Зал, Художник, Реставрація, Реставратор і Постачальник. Для кожної із зазначених сутностей були визначені відповідні атрибути.

Атрибути сутностей:

- Картина: код картини, назва картини, рік написання, художник, напрямок, постачальник, жанр.

- Жанр: код жанру, назва.
- Напрямок: код напрямку, назва.
- Виставки: код виставки, тема, дата початку, дата закінчення, зал, ціна, картини.
- Зал: код залу, назва, поверх, кількість кімнат.
- Художник: код художника, ПБ, дата народження, країна.
- Реставрація: код реставрації, картина, дата початку, дата завершення, реставратор, вид роботи, ціна.
- Реставратор: код реставратора, назва агентства, ПБ, номер телефону, електрона пошта.
- Постачальник: код постачальника, назва постачальника, ПБ, номер телефону, електрона пошта.

На рис. 2.1 представлена ER-діаграма, що зображує концептуальний рівень проектування інформаційної системи бази даних.

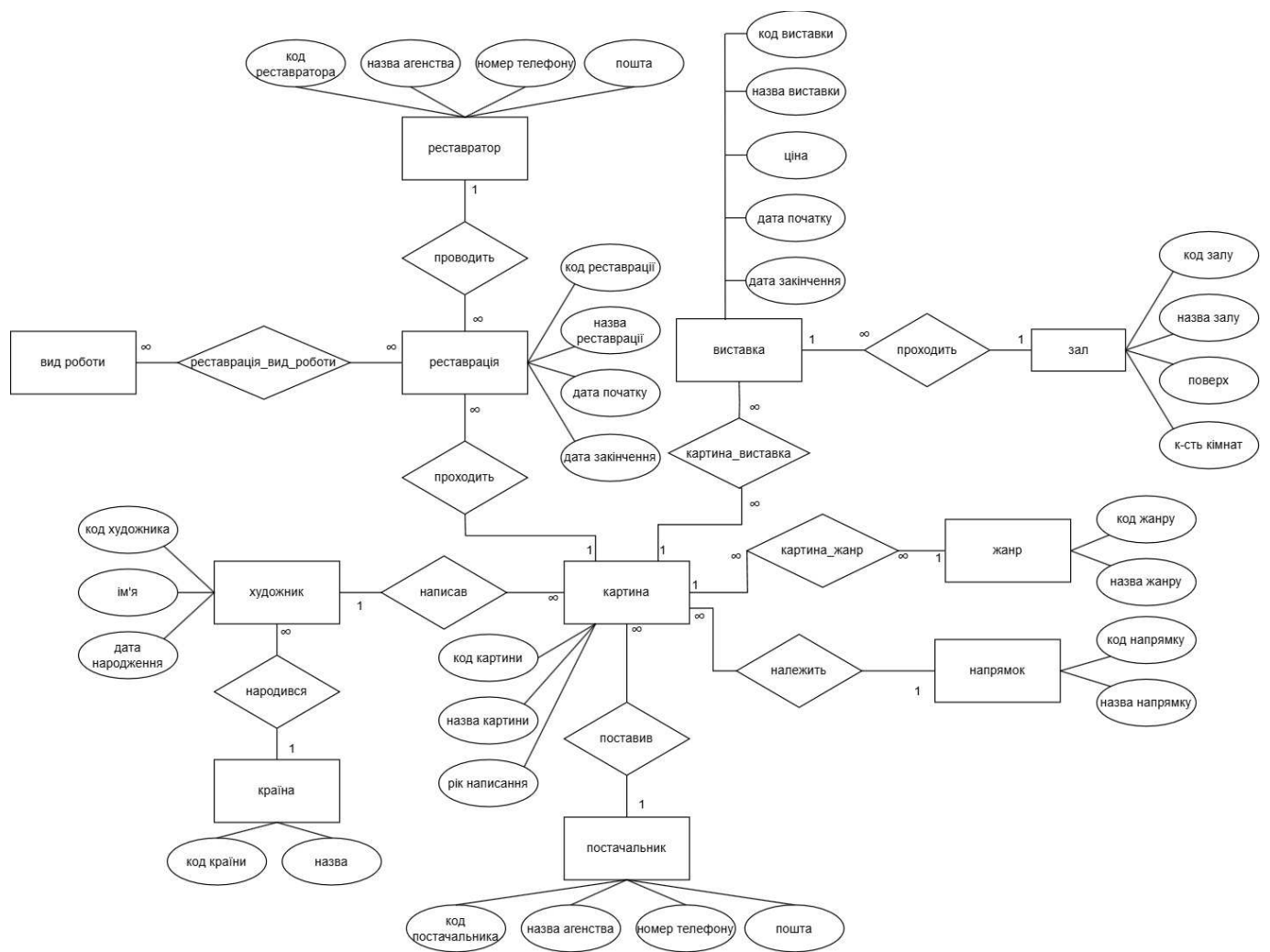


Рисунок 2.1 — ER-діаграма сутностей БД

Ключовим об'єктом системи є картина, яка є центральною сутністю, пов'язаною із такими об'єктами, як художник, жанр і напрямок мистецтва, постачальник, а також її участь у виставках та реставраційних роботах. Виставки проводяться у відповідних залах, які мають технічні характеристики (поверх, кількість кімнат), що також враховуються при організації експозицій.

Окрема частина концепції присвячена обліку реставраційних робіт. Передбачено збереження інформації про види робіт, реставраторів, агентства, до яких вони належать, а також часові рамки проведення робіт. Реставрації можуть бути

багатоетапними, тому реалізовано зв'язок між реставрацією та видами робіт у вигляді багатьох до багатьох.

Модель підтримує зв'язки «один до багатьох», тобто один художник може мати багато картин, та «багато до багатьох» — одна картина може мати декілька жанрів. Це забезпечує цілісність даних і дозволяє масштабувати систему за потреби.

Отже, розроблена концептуальна модель інформаційної системи у вигляді ER-діаграми. Вона відображає логіку зберігання та обробки інформації. Розроблена концепція є основою для створення фізичної структури бази даних та подальшої реалізації інформаційної системи.

2.2.2 Розробка логічної моделі даних

На етапі логічного проектування розробляється логічна модель даних, що є результатом трансформації раніше спроектованої концептуальної моделі. У реляційних базах даних зв'язки між таблицями визначають кореляції між множинами даних, що зберігаються у відповідних табличних структурах. Фундаментальні типи зв'язків, що використовуються в СУБД, включають наступні парадигми:

1. «Один до одного» (One-to-One).
2. «Один до багатьох» (One-to-Many).
3. «Багато до багатьох» (Many-to-Many) [7,8].

При розгляданні концептуальної моделі бази даних можемо виявити наступні зв'язки:

1. Художник — Картина: зв'язок «один до багатьох» (1 : M). Один художник може написати багато картин, а одна картина може бути написана лише одним художником.

1. Картина — Жанр: зв'язок «багато до багатьох» (N : M). Одна картина може мати багато жанрів, а один жанр може належати багатьом картинам, для реалізації створена ще одна таблиця: картина_жанр.

2. Картина — Виставка: зв'язок «багато до багатьох» (N : M). Одна картина може бути представлена на багатьох виставках, а одна виставка може включати багато картин, для реалізації створена ще одна таблиця: картина_виставка.

3. Картина — Реставрація: зв'язок «багато до багатьох» (1 : M). Одна картина може бути багато разів на реставрації, а на одній реставрації може бути одна картина.

4. Напрямок — Картина: зв'язок «один до багатьох» (1 : M). Одна картина може мати один напрямок, а один напрямок може належати багатьом картинам.

5. Зал — Виставка: зв'язок «один до багатьох» (1 : M). Одна виставка може проходити в багатьох залах, а один зал може належати одній виставці.

6. Постачальник — Картина: зв'язок «один до багатьох» (1 : M). Один постачальник може постачати багато картин, одна картина має лише одного постачальника.

7. Реставратор — Реставрація: зв'язок «один до багатьох» (1 : M). Один реставратор може проводити багато реставрацій, одна реставрація може проводитися тільки одним реставратором.

На рис. 2.8 реалізована ERD діаграма, що відображає всі сутності, атрибути, зв'язки та ключі.

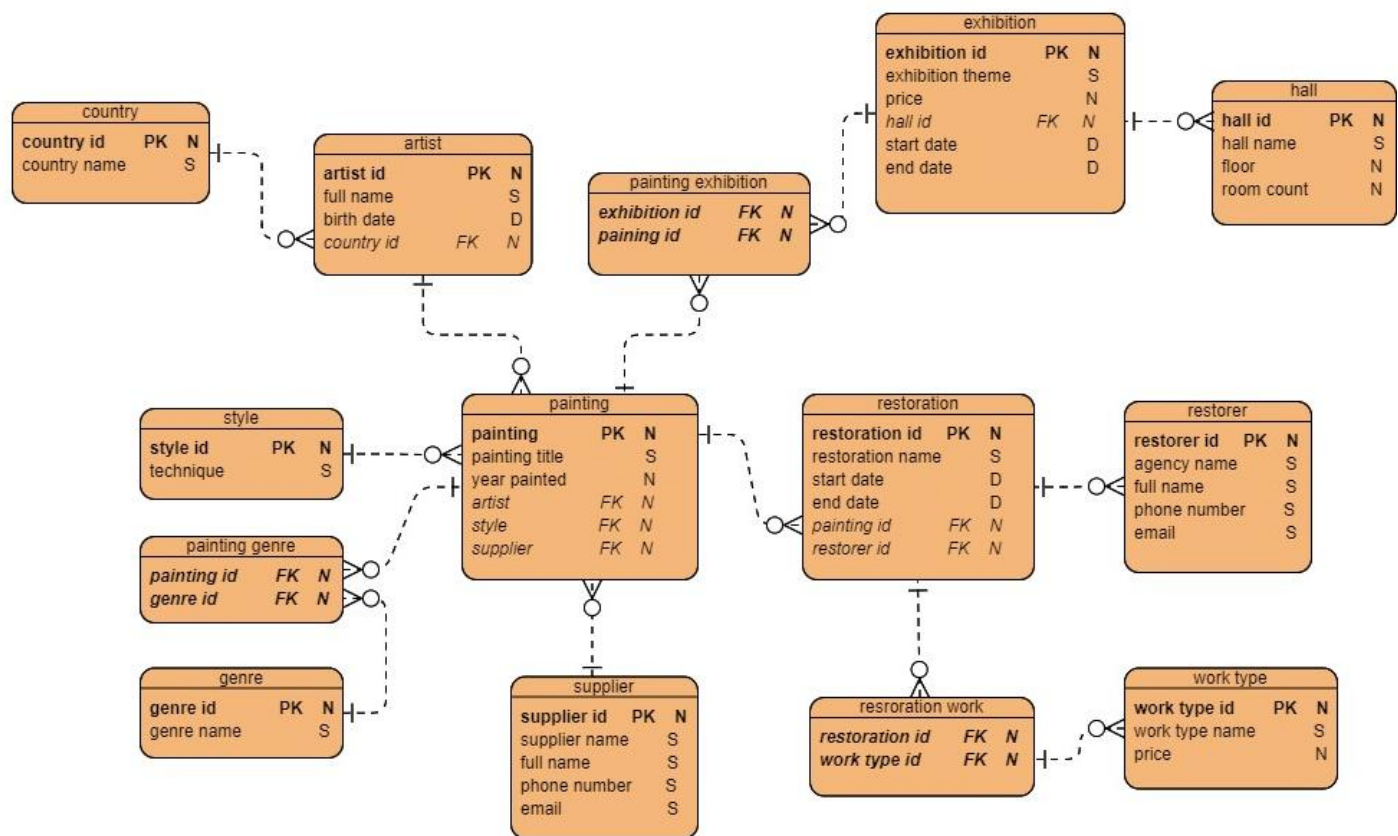


Рисунок 2.2 — ERD діаграма

Згідно за правилами переносу ключів маємо:

- До таблиці картина додаються атрибути: код художника, код напрямку, код постачальника.
- До таблиці реставрація додається атрибут код реставратора.
- До таблиці виставки додається атрибут код залу.
- Для реалізації зв'язку між таблицями картина та жанр створюється додаткова таблиця жанр_картина, що буде містити в собі коди відповідних таблиць.
- Для реалізації зв'язку між таблицями картина та виставка створюється додаткова таблиця картина_виставка, яка буде містити коди відповідних таблиць.

На основі розробленої логічної моделі даних, буде виконуватись фізичне проектування інформаційної системи.

2.2.3 Розроблення фізичної моделі даних

Під час опрацювання попередніх етапів розробки бази даних формувалися різноманітні послідовні та взаємозв'язані моделі, що давали уявлення про загальну будову проекту, візуалізували його опис і подали інформаційні структури на різних рівнях. Фізична модель — наступний етап проектування. Це модель даних, реалізована у вигляді опису бази даних.

Нормалізація відношень

Третя нормальна форма (3НФ) є одним із ключових принципів проектування реляційних баз даних, спрямованим на мінімізацію надмірності даних та усунення аномалій оновлення, вставки та видалення. Відношення знаходиться у 3НФ, якщо воно відповідає вимогам другої нормальної форми (2НФ) та не містить транзитивних функціональних залежностей неключових атрибутів. Транзитивна залежність виникає, коли неключовий атрибут залежить від іншого неключового атрибута, а не безпосередньо від первинного ключа [7, 8].

Для нормалізації бази даних та ліквідації небажаних функціональних залежностей між атрибутами та забезпечення мінімального дублювання даних, сутність художник розділяється на 2: художник та країна. Нова сутність допоможе уникнути помилок при заповненні БД. Атрибути сутності країна: код країни, назва країни.

Сутність реставрація містить в собі атрибут вид роботи та пов'язаний з ним — ціна, для нормалізації даного відношення створюється нова сутність вид роботи, що буде містити в собі наступні атрибути: код виду роботи, назва, ціна. Для реалізації даного зв'язку (N : M) створюється таблиця реставрація_вид, що містить в собі ключі відповідних таблиць.

Після проведення нормалізації, таблиці бази даних знаходиться в 3НФ.

Опис сутностей та атрибутів наведений в табл. 2.1.

Таблиця 2.1 — Опис атрибутів сутностей

Сутність картина				
Атрибут	Назва в СУБД	Тип атрибута	Обмеження	Властивість
Код_картини	Painting_Id	INT	NOT NULL	AUTO_INCREMENT, PRIMARY KEY
Назва_картини	Painting_Title	VARCHAR	NOT NULL, 100	
Рік_написання	Year_Painted	DATE	NOT NULL	
Художник	Artist_Id	INT	NOT NULL	FOREIGN KEY
Код_напрямку	Style_Id	INT	NOT NULL	FOREIGN KEY
Код_постачальника	Supplier_Id	INT	NOT NULL	FOREIGN KEY
Сутність художник				
Код_художника	Artist_Id	INT	NOT NULL	AUTO_INCREMENT, PRIMARY KEY
ПІБ	Full_Name	VARCHAR	NOT NULL, 40	
Дата_народження	Birth_Date	DATE	NOT NULL	
Код_країни	Country_Id	INT	NOT NULL	FOREIGN KEY

Сутність країна				
Код_країни	Country_Id	INT	NOT NULL	AUTO_INCREMENT, PRIMARY KEY
Країна	Country_Name	VARCHAR	NOT NULL, 30	
Сутність реставрація				
Код_реставрації	Restoration_Id	INT	NOT NULL	AUTO_INCREMENT, PRIMARY KEY
Назва_реставрації	Restoration_Name	VARCHAR	NOT NULL, 50	
Дата_початку	Start_Date	DATE	NOT NULL	
Дата_закінчення	End_Date	DATE		
Код_картини	Painting_Id	INT	NOT NULL	FOREIGN KEY
Код_реставратора	Restorer_Id	INT	NOT NULL	FOREIGN KEY
Сутність реставрація_вид				
Код_виду_роботи	Work_Type_Id	INT	NOT NULL	FOREIGN KEY
Код_реставрації	Restoration_Id	INT	NOT NULL	FOREIGN KEY

Сутність виставка				
Код_виставки	Exhibition_Id	INT	NOT NULL	AUTO_INCREMENT, PRIMARY KEY
Тема_виставки	Exhibition_The me	VARCHAR	NOT NULL, 50	
Код_залу	Hall_Id	INT	NOT NULL	FOREIGN KEY
Ціна	Price	INT	NOT NULL	
Дата_початку	Start_Date	DATE	NOT NULL	
Дата_закін- чення	End_Date	DATE	NOT NULL	
Сутність картина_виставка				
Код_виставки	Exhibition_Id	INT	NOT NULL	FOREIGN KEY
Код_картини	Painting_Id	INT	NOT NULL	FOREIGN KEY
Сутність жанр				
Код_жанру	Genre_Id	INT	NOT NULL	AUTO_INCREMENT, PRIMARY KEY
Жанр	Genre_Name	VARCHAR	NOT NULL, 25	
Сутність жанр_картина				
Код_жанру	Genre_Id	INT	NOT NULL	FOREIGN KEY
Код_картини	Painting_Id	INT	NOT NULL	FOREIGN KEY
Сутність напрямок				
Код_напрямку	Style_Id	INT	NOT NULL	AUTO_INCREMENT, PRIMARY KEY
Напрямок	Technique	VARCHAR	NOT NULL, 25	

Сутність постачальник				
Код_постачальника	Supplier_Id	INT	NOT NULL	AUTO_INCREMENT, PRIMARY KEY
Назва_постачальника	Supplier_Name	VARCHAR	NOT NULL, 50	
ПІБ	Full_Name	VARCHAR	NOT NULL, 50	
Номер_телефону	Phone_Number	VARCHAR	NOT NULL, 50	
Електрона_пошта	Email	VARCHAR	NOT NULL, 50	
Сутність реставратор				
Код_реставратора	Restorer_Id	INT	NOT NULL	AUTO_INCREMENT, PRIMARY KEY
Назва_агенції	Agency_Name	VARCHAR	NOT NULL, 50	
ПІБ	Full_Name	VARCHAR	NOT NULL, 50	
Номер_телефону	Phone_Number	VARCHAR	NOT NULL, 50	
Електрона_пошта	Email	VARCHAR	NOT NULL, 50	
Сутність реставрація_вид				
Код_реставрації	Restoration_Id	INT	NOT NULL	FOREIGN KEY
Код_виду_роботи	Work_Type_Id	INT	NOT NULL	FOREIGN KEY

Сутність вид_роботи				
Код_виду_ро- боти	Work_Type_Id	INT	NOT NULL	AUTO_INCREMENT, PRIMARY KEY
Назва_ виду_роботи	Work_Type_Na me	VARCHAR	NOT NULL, 40	
Ціна	Price	INT	NOT NULL	

Структура бази даних реалізована в обраній СУБД та представлена на рис. 2.9.

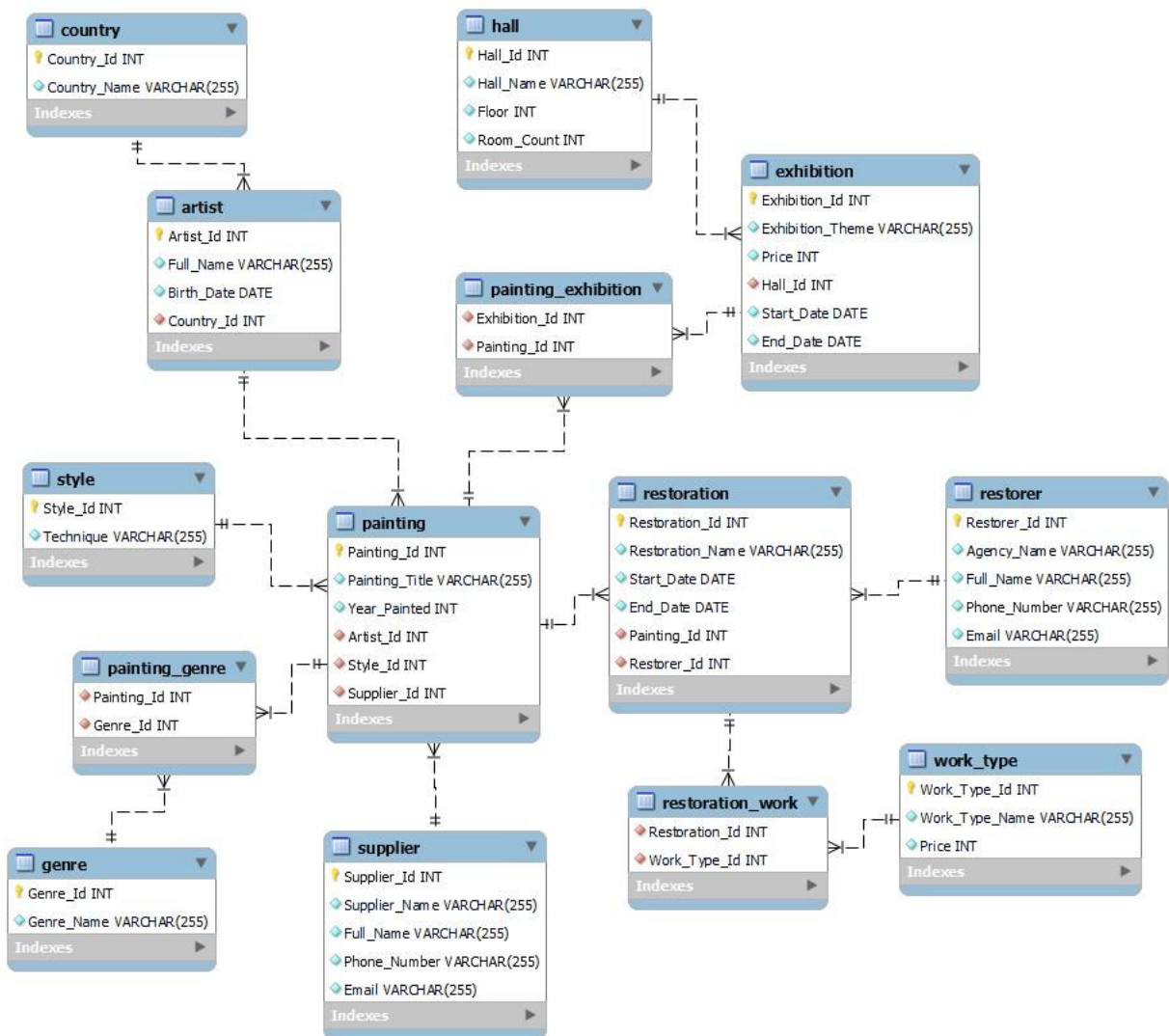


Рисунок 2.3 — Фізична модель БД

У результаті розроблено фізичну модель даних, яка відповідає вимогам третьої нормальної форми та забезпечує цілісність і не надмірність інформації.

2.3 Рішення з програмного забезпечення

Для забезпечення стійкої та ефективної взаємодії з системою керування базами даних MySQL у розробленій інформаційній системі застосовано технологію Java Persistence API (JPA) у поєднанні з фреймворком Spring Data JPA, що базується на ORM-бібліотеці Hibernate. JPA визначає стандартизований інтерфейс для управління даними в Java-застосунках, забезпечуючи абстракцію від особливостей конкретної СУБД [12].

Spring Data JPA, у свою чергу, значно спрощує процес доступу до даних шляхом автоматичної генерації значної частини коду, необхідного для виконання типових операцій CRUD (створення, читання, оновлення, видалення). Використання репозиторіїв Spring Data JPA дозволяє зосередитися на бізнес-логіці, мінімізуючи обсяг шаблонного коду для взаємодії з базою даних [12].

Hibernate, як провідна реалізація JPA, забезпечує відображення об'єктної моделі застосунку на реляційну модель бази даних. Він автоматично генерує SQL-запити, керує транзакціями та забезпечує механізми кешування для оптимізації продуктивності. Таким чином, комбіноване використання JPA, Spring Data JPA та Hibernate сприяє розробці надійного, масштабованого та зручного у підтримці рівня доступу до даних у інформаційній системі картинної галереї [9].

В рамках використання даних технологій для кожної таблиці було створено відповідні класи сутностей бази даних. Коди класів сутностей наведені у додатку Б.

Клас сутності Painting використовується для представлення інформації про картини в базі даних. Він містить поля для зберігання основних характеристик картини,

таких як ідентифікатор (id), назва (paintingTitle), рік створення (yearPainted), а також посилання на автора (artist), стиль (style) та постачальника (supplier).

За допомогою анотацій JPA (@Entity — сутність, @Table — вказує на конкретну таблицю в базі даних, @Id — первинний ключ, @GeneratedValue — значення, що буде генеруватися автоматично, @Column — пов'язує поле з відповідним стовпцем таблиці, @ManyToOne — тип зв'язку, @JoinColumn — зовнішній ключ, @ManyToMany) клас відображається на таблицю "painting" у базі даних та визначає зв'язки з іншими таблицями: "artist", "style", "supplier" (зв'язок «багато до одного») та "genre" (зв'язок «багато до багатьох» через проміжну таблицю "painting_genre").

Використання анотацій Lombok (@Getter, @Setter) автоматично генерує методи для доступу до полів класу (геттери та сеттери), спрощуючи таким чином написання коду. Поле genres представляє набір жанрів, до яких належить картина, використовуючи Set для уникнення дублікатів.

Загалом, цей клас є моделлю даних для картин, що визначає їхні атрибути та взаємозв'язки з іншими сутностями в системі керування картинною галереєю. Інші класи наведені у додатку Б.

Також при розробці було використано фреймворк Spring Boot, що базується на Java та Spring Framework, і призначений для спрощення процесу розробки, конфігурації та розгортання автономних production-ready Spring-застосунків. Однією з ключових переваг Spring Boot є його здатність автоматично налаштовувати більшість необхідних компонентів на основі доданих залежностей, що значно зменшує обсяг ручної конфігурації.

Крім того, Spring Boot забезпечує легку інтеграцію з різноманітними технологіями та бібліотеками. Зокрема, для створення динамічних веб-інтерфейсів Spring Boot спрощує підключення шаблонізаторів, таких як Thymeleaf. Додавання відповідної залежності автоматично налаштовує Thymeleaf як механізм представлення, дозволяючи

використовувати його синтаксис для створення HTML-шаблонів, що динамічно заповнюються даними з бекенду застосунку [9].

На прикладі наведеного в додатках лістингу веб-сторінки «картини» розглянемо використання шаблону Thymeleaf для динамічного формування таблиці зі списком картин. Атрибут `th:each` здійснює ітерацію по колекції `paintings`, для кожної картини створюючи новий рядок `<tr>`.

У кожному рядку атрибут `th:text` відповідає за відображення відповідних властивостей картини, таких як `id`, `paintingTitle`, `yearPainted` та інформації про пов'язані сутності (`artist.fullName`, `style.technique`, `supplier.fullName`).

Крім того, Thymeleaf використовується для генерації динамічних URL-адрес за допомогою `th:href` (наприклад, для перегляду жанрів картини) та для передачі атрибутів даних до HTML-елементів через `th:attr` (для модальних вікон редагування та видалення), що дозволяє клієнтському JavaScript отримувати необхідну інформацію про кожну картину.

Таким чином, Thymeleaf виступає як сполучна ланка між даними, отриманими з бекенду, та HTML-відображенням, забезпечуючи динамічне наповнення таблиці інформацією про кожну картину та інтерактивність елементів інтерфейсу.

Зв'язок між класами в проєкті відбувається через певні шари та компоненти, такі як сервіси, контролери, репозиторії та представлення (веб-сторінки).

Репозиторії використовуються для взаємодії з базою даних. Кожен репозиторій відповідає за одну таблицю в базі даних і містить методи для здійснення операцій з цією таблицею (створення, читання, оновлення, видалення). Вони використовують сервіси для отримання/збереження даних у базу даних. Також в репозиторіях можна виконувати запити. Наприклад репозиторій `GenreRepository` (додаток Б) містить два методи `findByPaintingsArtistId(Integer id)` та `findGenresNotInPainting (@Param("paintingId") int paintingId)`. Перший метод виконує запит до бази даних для

отримання списку жанрів, пов'язаних з картинами, які належать митцю з вказаним ідентифікатором.

Метод `findGenresNotInPainting` `findGenresNotInPainting(@Param("paintingId") int paintingId)` також виконує запит до бази даних, але для отримання жанрів, які не пов'язані з конкретною картиною. В параметрах метода передається ідентифікатор конкретної картини, щоб виключити з результатів жанри, які вже пов'язані з нею.

Сервіси містять логіку застосунку. Вони координують роботу репозиторіїв та виконують операції з даними. Сервіси можуть взаємодіяти з більш низьким рівнем (репозиторії) для отримання даних, обробки їх та передачі вищому рівню (контролерам) для подальшої обробки. Розглянемо `PaintingService` (додаток Б), він відповідає за логіку обробки даних, пов'язаних з картинами в системі. У ньому присутні наступні методи:

Метод `findAll()` отримує всі картини з бази даних, використовуючи стандартний метод `findAll()` з `paintingRepository`, і повертає список усіх картин.

Метод `find(int id)`: приймає ідентифікатор картини і знаходить відповідну картину у базі даних, використовуючи метод `findById(id)` з `paintingRepository`. Якщо картину не знайдено, повертається порожній об'єкт `Painting`.

Метод `addPainting(String title, int year, Artist artist, Style style, List<Integer> genres, Supplier supplier)` додає нову картину до бази даних. Спочатку він створює новий об'єкт `Painting` з отриманими даними. В свою чергу він викликає `genreService`, щоб отримати лист (список) зі всіх жанрів. Збереження до бази даних відбувається через метод `save(painting)` з `paintingRepository`. Спочатку відбувається збереження картини, а потім до уже існуючого запису додаються жанри.

Метод `findAllPaintingById(List<Integer> ids)` — приймає список ідентифікаторів картин і повертає лише ті картини, ідентифікатори яких знаходяться у переданому списку.

Метод `findGenresNotInPainting(int id)`: отримує ідентифікатор картини та повертає список жанрів, які не пов'язані з цією картиною. Для цього він викликає метод `findGenresNotInPainting(id)` з `genreRepository`, який виконує запит до бази даних і повертає жанри, які не включені до картини з вказаним ідентифікатором.

Контролери відповідають за обробку HTTP-запитів та взаємодію з клієнтом. Вони викликають відповідні методи сервісів для отримання необхідних даних та передачі їх на сторінку (або відповідь на запит). Контролери також відповідають за обробку вхідних даних від користувача, перевірку їх коректності та передачу сервісам для подальшої обробки.

Розглянемо контролер `PaintingController` (додаток Б) який відповідає за обробку HTTP-запитів, пов'язаних з картинами в системі. Містить наступні методи:

Метод `paintings(Model model)` — обробляє GET-запит до URL `"/painting"`, який призначений для відображення списку всіх картин. Викликає метод `paintingService` — `findAll()`, щоб отримати список усіх картин, та додає цей список до моделі.

Метод `paintingGenre(@RequestParam int id, Model model)` — обробляє GET-запит до URL `"/painting_genre"`, та відображає всі жанри конкретної картини. Він отримує ідентифікатор картини через `id` та викликає метод сервісу `find(id)`, щоб отримати картину. Також він викликає метод сервісу `findGenresNotInPainting(id)`. Обидва отримані об'єкти (картина та список жанрів) додаються до моделі для відображення на сторінці.

Метод `addPainting()` — обробляє GET-запит до URL `"/add_painting"`, який призначений для відображення сторінки додавання нової картини (форми).

За додання самих картин відповідає метод `addPainting(@RequestParam String title, @RequestParam int year, @RequestParam Artist artist, @RequestParam Style style, @RequestParam List<Integer> genre, @RequestParam Supplier supplier)`. Він обробляє GET-запит до URL `"/addPainting"`. Після отримання всіх параметрів викликається

метод `addPainting()` `paintingService`, щоб додати нову картину до бази даних. Після успішного додавання картини користувач перенаправляється назад на сторінку `"/painting"`.

Метод `deletePainting(@PathVariable("id") int id)` — відповідає за видалення картин. Він обробляє GET-запит до URL `"/painting/delete_painting/{id}"`, який призначений для видалення картини з бази даних за її ідентифікатором.

Сторінки або веб-інтерфейс відображають інформацію для користувача та дозволяють користувачам взаємодіяти з застосунком. Вони отримують дані від контролерів та відображають їх на веб-сторінках або в інтерфейсі користувача. Представлення можуть надсилати дані до контролерів для обробки введених даних або для виконання певних дій. Приклад веб-сторінки представлений у додатку Б.

Звіт по жанрам, художникам та картинам. Для створення звіту використовується репозиторій `GenreRepository`. Він визначає метод `findAllGenresWithArtistsAndPaintings()`, який містить JPQL-запит для отримання даних про жанри з інформацією про художників і картин. Виконує цей запит до бази даних та повертає результат у сервіс. Сервіс в свою чергу отримує запит від контролера та звертається до репозиторію для пошуку відповідних даних.

На основі всіх попередніх етапів розробки були прийняті такі рішення з програмного забезпечення, що дозволить реалізувати всі вимоги до застосунку.

2.4 Рішення з технічного забезпечення

Для розгортання системи достатньо мати одноплатний комп'ютер, наприклад Raspberry Pi 4. Вибір даного апаратного рішення обумовлений його оптимальним співвідношенням продуктивності, енергоефективності та вартості, що робить його

придатним для функціонування інформаційної системи в автономному режимі або в складі невеликої локальної мережі. Raspberry Pi 4 Model B є значним удосконаленням порівняно з попередніми поколіннями, пропонуючи користувачеві продуктивність, порівнянну з настільними комп'ютерами початкового рівня на архітектурі X86 [13].

Плата оснащена високопродуктивним 64-розрядним чотири ядерним процесором, що забезпечує достатню обчислювальну потужність для обробки запитів та керування базою даних. Покращений мультимедійний модуль та збільшена швидкість і об'єм оперативної пам'яті (від 1 до 4 ГБ залежно від моделі) сприяють плавному функціонуванню веб-сервера та обробці графічного контенту. Підтримка двох дисплеїв з роздільною здатністю до 4K через два порти micro-HDMI розширює можливості використання системи, наприклад, для демонстраційних цілей. Наявність Gigabit Ethernet та дводіапазонного Wi-Fi 5.0 забезпечує гнучкість підключення до мережі, а порти USB 3.0 дозволяють підключати необхідну периферію [13].

Вимоги до клієнтської частини

Для клієнтського комп'ютера потрібна наявність будь-якого сучасного веб-браузера та доступ до мережі Інтернет. Оскільки розроблена система являє собою веб-застосунок, вся взаємодія користувача з інформаційною системою відбувається через стандартний веб-браузер. Це забезпечує кросплатформність та не вимагає встановлення додаткового програмного забезпечення на стороні клієнта. Наявність інтернет-з'єднання є необхідною умовою для обміну даними з сервером, на якому розгорнуто основну частину інформаційної системи та базу даних. Такий підхід спрощує процес доступу до системи для кінцевих користувачів, незалежно від їхнього операційного середовища.

Висновок

Отже, для розгортання системи було обрано апаратну платформу Raspberry Pi 4, що забезпечує достатню обчислювальну потужність, енергоефективність та низьку

вартість для автономного або локального розгортання системи. В свою чергу для клієнта потрібно доступ до будь-якого сучасного браузеру, що гарантує кросплатформність.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОЕКТУ ІНФОРМАЦІЙНОЇ СИСТЕМИ КАРТИННОЇ ГАЛЕРЕЇ

Для впровадження системи розроблено інструкцію користувача, яка наведена в додатку А.

Аналіз отриманих результатів

Під час розробки бази даних та веб-застосунку було дотримано відповідних вимог, створено коди програмних модулів з використанням технологій Spring (додаток Б) і реалізовано наступні функції.

- Перегляд художників: адміністратори можуть переглядати інформацію про художників та їх твори, список жанрів та картин кожного художника, створювати звіт за жанрами.

- Перегляд картин: адміністратори можуть переглядати колекцію картин та дані про них: рік написання, художник, стиль, постачальник, список всіх жанрів кожної картини, та жанрів, які не належать до картини, додавати та видаляти картини зі списку.

- Перегляд виставок: адміністратори можуть дізнаватися виставки, їх теми, місця проведення дати початку та кінця, зали, де вони проходять, переглядати список всіх картин, що представлені на виставках, додавати\видаляти виставки.

- Реставрація картин: адміністратори можуть переглядати, додавати та видаляти записи про реставрацію картин (дані про реставратора, дата початку\закінчення реставрації, види робіт, що виконувались над картиною), створювати звіт по всім реставраціям та завдяки йому переглядати загальну суму кожної реставрації.

Розгортання системи буде відбуватись на одноплатному комп'ютері Raspberry Pi 4.

Головна сторінка web-застосунку має посилання на наступні сторінки: Картини, Виставки, Реставрації, Художники. Вигляд сторінки представлений на рис. 3.1

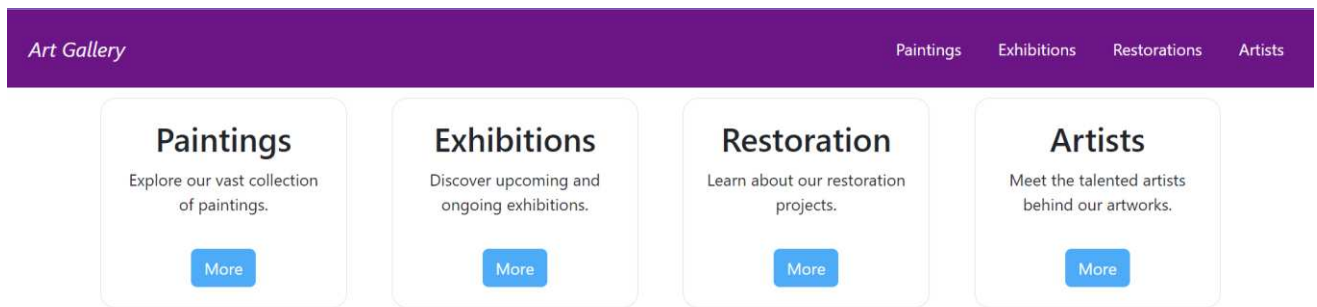


Рисунок 3. 1 — головна сторінка застосунку

В майбутньому застосунок можна вдосконалити та реалізувати нові можливості та функціонал. Додавання та видалення жанрів для кожної картини. Додати зображення для картин та інакше організувати їх вивід. Реалізувати пошук картин за різними критеріями. Організувати різні види доступу, як наприклад: адміністратор з можливістю видалення будь-яких даних, додання та редагування, менеджер або робітник галереї з можливостями додання та редагування, та відвідувачі галереї для яких доступний тільки перегляд.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ

1. Загальні положення

У сфері інформаційних технологій охорона праці набуває особливого значення у зв'язку з тривалою статичною роботою, напругою зору, впливом електромагнітного випромінювання, небезпекою ураження електричним струмом тощо. Організація безпечного робочого процесу з використанням комп'ютерної техніки є необхідною умовою створення комфортного середовища для користувача.

2. Вимоги до організації робочого місця

2.1. Розміщення обладнання

Робоче місце користувача комп'ютера має відповідати встановленим гігієнічним та ергономічним вимогам. Розташування столів повинно забезпечувати мінімальну відстань між боковими поверхнями моніторів — не менше 1,2 м, а відстань до стіни або іншої перешкоди перед екраном — не менше 1 м [14].

2.2. Освітлення

Недостатнє або неправильне освітлення призводить до підвищеного навантаження на зір і може викликати головний біль або швидку втому. Робоче місце повинно мати комбіноване освітлення (природне та штучне), при цьому яскравість екрану повинна відповідати умовам навколишнього середовища [15]. Оптимальний рівень освітленості для офісних приміщень — 300-500 лк.

2.3. Мікроклімат

Комфортний мікроклімат є важливим чинником для підтримання працездатності. Температура повітря в приміщенні повинна бути в межах 20-24°C, вологість — 40-60%. Повітря повинно бути свіжим і не містити шкідливих речовин. Вентиляція — природна або примусова — має забезпечувати 60 м³/год на одну людину [16].

3. Ергономіка та робоча поза

Згідно з ДСанПіН 3.3.2-007-98 [17], ергономіка робочого місця є запорукою зниження ризику професійних захворювань, таких як остеохондроз, тунельний синдром, короткозорість.

3.1. Стіл та стілець

Стіл повинен мати висоту 700-750 мм, а стілець — регулюватися по висоті та мати спинку. Наявність підставки під ноги рекомендується. Поверхня столу має бути матовою, без відблисків.

3.2. Положення тіла

- Екран розміщують на відстані 50-70 см від очей.
- Центр екрана повинен бути на 10-20 см нижче рівня очей.
- Передпліччя мають лежати на столі або підлокітниках стільця.

4. Режим праці та відпочинку

4.1. Робочий час

Для дорослих працівників час роботи за комп'ютером не повинен перевищувати 6 годин на день. При 8-годинному робочому дні решта часу має бути присвячена іншим видам діяльності [17].

4.2. Перерви

Рекомендується кожні 60 хвилин роботи робити 10-15-хвилинні перерви. Під час перерв доцільно виконувати гімнастику для очей (рухи вгору-вниз, вліво-вправо, фокусування на віддаленому предметі) та фізичні вправи (нахили, обертання головою, розтягування рук і спини).

5. Електробезпека

5.1. Основні вимоги

Комп'ютери та інша офісна техніка підключаються через справні розетки з заземленням. Заборонено користуватися подовжувачами без захисту, а також самостійно ремонтувати обладнання без відповідної підготовки [18].

5.2. Захист від перенапруги

Необхідно використовувати мережеві фільтри, стабілізатори напруги та джерела безперебійного живлення (UPS), що захищають обладнання від коливань напруги в електромережі.

6. Пожежна безпека

Комп'ютерна техніка є джерелом потенційної пожежної небезпеки. Основні правила пожежної безпеки:

- Не залишати обладнання без нагляду у ввімкненому стані;
- не закривати вентиляційні отвори пристроїв;
- заборонено використовувати поблизу техніки горючі матеріали.

В приміщенні має бути вогнегасник (порошковий або вуглекислотний) з маркуванням [19].

7. Шкідливі фактори при роботі з ПК

Основні фактори, що можуть негативно впливати на здоров'я користувача:

Електромагнітне випромінювання — рівень випромінювання сучасних моніторів відповідає міжнародним стандартам, але все ж рекомендується обмежити тривале перебування біля пристроїв. Статичне навантаження — може викликати біль у спині, шиї, руках [20].

8. Дії в аварійних ситуаціях

При виникненні аварії необхідно:

- негайно знеструмити обладнання;
- повідомити керівництво;
- у разі пожежі — викликати службу порятунку за номером 101;
- у разі травм — надати першу допомогу та викликати швидку (103).

Дотримання вимог охорони праці при роботі з комп'ютерною технікою — обов'язкова умова для забезпечення безпеки працівників. Впровадження ергономічних та санітарних норм дозволяє зменшити вплив шкідливих факторів та підвищити ефективність роботи.

ВИСНОВКИ

У роботі над проектом з базою даних «Картинна галерея» було розроблено та реалізовано комплексний веб-застосунок, який включає в себе різноманітні функціональні можливості для керування картинами, виставками, реставраціями та іншими аспектами роботи галереї.

На етапі розробки був проведений детальний аналіз вимог, визначено основні сутності та їх взаємозв'язки. Була розроблена модель даних і створена база даних MySQL, включаючи таблиці для представлення інформації про картини, митців, виставки, жанри, стилі та інше. Розроблена база даних «Картинна галерея» містить чотирнадцять таблиць, звіти по реставраціям та виставкам, два запити на вибірку жанрів по картинам та художникам, форми на додання картин, виставок, реставрацій. Для зручної взаємодії користувачів та роботи з системою розроблено відповідні HTML-сторінки, що візуалізують весь функціонал описаний вище.

Для забезпечення функціональності веб-застосунку були розроблені відповідні сервіси, контролери та репозиторії мовою Java. Сервіси виконують логічні операції над даними, контролери обробляють HTTP-запити та керують взаємодією між користувачем та системою, а репозиторії взаємодіють з базою даних для зберігання та отримання даних.

Отже, завдяки використанню розробленого веб-застосунку, користувачі можуть легко взаємодіяти з базою даних картинної галереї, а також керувати різноманітними аспектами її роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. UML [Електронний ресурс]: Вікіпедія. Вільна енциклопедія. — Режим звернення: <https://ru.wikipedia.org/wiki/UML> — дата звернення: 09.03.2025.
2. UML [Електронний ресурс]: KyivGallery. Вплив сучасного мистецтва на нашу свідомість. — Режим звернення: <https://kyiv.gallery/> — дата звернення: 09.03.2025.
3. Старостенко Г. Положення про національний музей «Київська картинна галерея» 2017 — 9с. Режим доступу: [https://kyivaudit.gov.ua/vr/ka/company.nsf/0/4D54BF0AB0C8B09DC225822F00503441/\\$file.pdf](https://kyivaudit.gov.ua/vr/ka/company.nsf/0/4D54BF0AB0C8B09DC225822F00503441/$file.pdf) — дата звернення: 09.03.2025.
4. Ременяк Л.В. Проектування інформаційних систем: конспект лекцій. Одеса, Одеський державний екологічний університет, 2016. 152 — дата звернення: 24.04.2025.
5. Вендров А. М. Проектування програмного забезпечення економічних інформаційних систем. М.: Фінанси та статистика, 2006. — 544 с — дата звернення: 24.04.2025.
6. Марченко А.В. Проектування інформаційних систем 2016 90с. Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20160217112601/content-20160217112601.pdf — дата звернення: 25.04.2025.
7. Доценко С. І. Організація та системи керування базами даних: Навч. посібник. — Харків: УкрДУЗТ, 2023. — 117 с. Режим доступу: <http://lib.kart.edu.ua/bitstream.pdf> — дата звернення: 25.04.2025.
8. Сидоренко В.В., Константинова Л.В., Смірнов С.А. Організація баз даних: Навчальний посібник. — Кропивницький: ЦНТУ, 2018. — 274 с. режим доступу: <https://dspace.kntu.kr.ua/server/api/core/bitstreams/a050035e-e868-4140-b2ba-e449d7394255/content> — дата звернення: 25.04.2025.
9. UML [Електронний ресурс]: Знайомство з Maven, Spring, MySQL, Hibernate та перший CRUD додаток. — Режим звернення:

<https://javarush.com/ua/groups/posts/uk.2251.znayomstvo-z-maven-spring-mysql-ibernet-ta-pershiy-crud-dodatok-chastina-3> — дата звернення: 04.05.2025.

10. Джошуа Блох. Ефективна Java; [В оригіналі] 2016. — 440 с дата звернення: 04.05.2025.

11. Гослинг, Джей; Джой, Білл; Брача, Гіл; Баклі, Гай-Стіл. *The Java Language Specification*. Addison-Wesley Professional, США, 2014. — 876 с. режим доступу: <https://docs.oracle.com/javase/specs/jls/se24/jls24.pdf> — дата звернення: 11.05.2025.

12. Дакетт, Джон. *HTML and CSS: Design and Build Websites*. Wiley, США, 2011. — 514 с. режим доступу: <https://wtf.tw/ref/duckett.pdf> — дата звернення: 11.05.2025.

13. UML [Електронний ресурс]: Інтернет-магазин мережевого обладнання <https://networkdiscount.com.ua/ua/p1351615191-mikrokompyuter-raspberry-model.html> — дата звернення: 28.05.2025.

14. UML [Електронний ресурс]: Закон України «Про охорону праці» № 2694-XI. — <https://zakon.rada.gov.ua/laws/show/2694-12> — дата звернення: 28.05.2025.

15. UML [Електронний ресурс]: Державні санітарні правила та норми ДСанПіН 5.5.6.009-98 — <https://zakon.rada.gov.ua/laws/show/z0753-98> — дата звернення: 28.05.2025.

16. UML [Електронний ресурс]: Інструкція з охорони праці при роботі з персональним комп'ютером — <https://osvita-docs.com/node/321> — дата звернення: 28.05.2025.

17. UML [Електронний ресурс]: Вимоги до мікроклімату в робочих приміщеннях — <https://zakon.rada.gov.ua/laws/show/z0630-06> — дата звернення: 28.05.2025.

18. UML [Електронний ресурс]: ДСанПіН 3.3.2-007-98 «Гігієнічні вимоги до відеодисплейних терміналів» — <https://zakon.rada.gov.ua/laws/show/z0753-98> — дата звернення: 28.05.2025.

19. UML [Електронний ресурс]: Інструкція з електробезпеки в офісі — <https://www.niiosvita.gov.ua/wp-content/uploads/2023/01/instrukciya-elektro.pdf> — дата звернення: 28.05.2025.

20. UML [Електронний ресурс]: Правила пожежної безпеки в Україні. Наказ МВС № 1417 від 30.12.2014 р. – <https://zakon.rada.gov.ua/laws/show/z0136-15> — дата звернення: 28.05.2025.

ДОДАТКИ

Додаток А

Інструкція користувача.

При вході в систему можна побачити екран, наведений на рис. 5.1

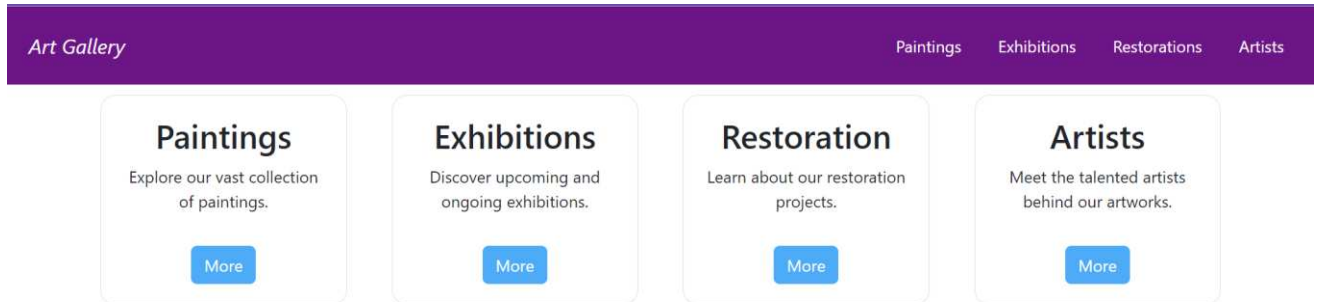


Рисунок 5. 1 — головна сторінка застосунку

При натисканні на Paintings можна потрапити на сторінку Картини (рис 5.2), де представлена інформація про всі картини у галереї, також створена форма для додавання нових картин (рис 5.3) та сторінка, на якій можна переглянути жанри кожної картини та жанри до яких картина не належить (рис 5.4). Також на сторінці реалізована можливість видалення картин.

Art Gallery							
Paintings Exhibitions Restorations Artists							
Add Painting							
id	paining title	year painted	artist	style	genre	supplier	delete
1	The Starry Night	1889	Vincent van Gogh	Oil Painting	genre	John Smith	delete
2	Mona Lisa	1503	Leonardo da Vinci	Oil Painting	genre	Emma Johnson	delete
3	Guernica	1937	Pablo Picasso	Sculpture	genre	Michael Brown	delete
4	The Night Watch	1642	Rembrandt	Oil Painting	genre	Sarah Davis	delete
5	The Creation of Adam	1512	Michelangelo	Sculpture	genre	Christopher Wilson	delete
6	The Persistence of Memory	1931	Vincent van Gogh	Fresco	genre	John Smith	delete
7	The Last Supper	1495	Leonardo da Vinci	Sculpture	genre	Emma Johnson	delete

Рисунок 5. 2 — сторінка Картини

Add New Painting

Title

Year Painted

Artist

Select artist

Style

Select style

Genre

Landscape
Portrait
Still Life

Supplier

Select supplier

Submit

Рисунок 5. 3 — форма для додавання нових картин

Art Gallery

PaintingsExhibitionsRestorationsArtists

The Starry Night

Current Genres

Available Genres

Genre

Landscape

Portrait

Genre

Still Life

Historical

Abstract

Figurative Composition

Scenic

Рисунок 5.4 — жанри картини

Щоб потрапити на сторінку Виставки (рис. 5.5), потрібно натиснути Exhibitions, тут представлена інформація про всі виставки, що проходять у галереї, створена форма для додавання нових виставок (рис 5.6) та сторінка, на якій відображаються всі картини на виставці (рис 5.7). Також на сторінці реалізована можливість видалення виставок при натисканні кнопки delete.

Art Gallery

PaintingsExhibitionsRestorationsArtists

Add Exhibition

id	theme	price	hall	start date	end date	paintings	delete
1	Impressionism Masterpieces	100	Main Hall	2024-05-01	2024-06-30	paintings	delete
2	Renaissance Art Showcase	150	Gallery A	2024-07-15	2024-08-31	paintings	delete
3	Modern Art Revolution	200	Gallery B	2024-09-15	2024-10-31	paintings	delete
4	Baroque Extravaganza	100	Main Hall	2024-11-15	2025-01-15	paintings	delete
5	Contemporary Creations	300	Gallery A	2025-02-01	2025-03-31	paintings	delete
6	Abstract Art Exhibition	150	Gallery B	2025-04-15	2025-06-30	paintings	delete

Рисунок 5.5 — сторінка Виставки

Add New Exhibition

Theme

Price

Hall

Start Date

End Date

Painting

The Starry Night
Mona Lisa
Guernica
The Night Watch

Рисунок 5.6 — форма для додавання нових виставок

Art Gallery			
		Paintings	Exhibitions
		Restorations	Artists
Impressionism Masterpieces			
id	painting	artist	style
2	Mona Lisa	Leonardo da Vinci	Oil Painting
1	The Starry Night	Vincent van Gogh	Oil Painting
3	Guernica	Pablo Picasso	Sculpture

Рисунок 5.7 — список картин на виставці

На сторінці Реставрації (рис. 5.8) представлена інформація про всі реставрації. Реалізовано створення звіту по кожній реставрації: список всіх реставрацій, список робіт, що проводились з картинами та загальна вартість кожної реставрації (рис. 5.9). Створена форма для додавання нових виставок (рис 5.11) та сторінка, на якій

відображаються всі види робіт, що виконувались на обраній реставрації (рис 5.10). Також на сторінці реалізована можливість видалення реставрацій при натисканні кнопки delete.

Art Gallery

PaintingsExhibitionsRestorationsArtists

View Restoration Report

Add Restoration

id	restoration name	start date	end date	painting	restorer	work type	delete
1	The Starry Night Restoration	2024-01-15	2024-03-30	The Starry Night	Jennifer Lee	work type	delete
2	Mona Lisa Cleaning	2024-02-20	2024-04-15	Mona Lisa	Daniel Martinez	work type	delete
3	Guernica Repair	2024-03-10	2024-06-01	Guernica	Emily Brown	work type	delete
4	The Night Watch Restoration	2024-04-05	2024-07-20	The Night Watch	Jennifer Lee	work type	delete
5	The Persistence of Memory Restoration	2024-05-20	2024-08-10	The Creation of Adam	Daniel Martinez	work type	delete
6	The Last Supper Preservation	2024-06-15	2024-09-30	The Persistence of Memory	Emily Brown	work type	delete

Рисунок 5.8 — вигляд сторінки Реставрації

Art Gallery

PaintingsExhibitionsRestorationsArtists

Restoration Report

Restoration Name	Start Date	End Date	Work Types
The Starry Night Restoration	2024-01-15	2024-03-30	Cleaning - 1500 ₴ Frame repair - 500 ₴ Color restoration - 2000 ₴
Mona Lisa Cleaning	2024-02-20	2024-04-15	Frame repair - 500 ₴ Applying varnish - 1500 ₴ Color restoration - 2000 ₴
Guernica Repair	2024-03-10	2024-06-01	Cleaning - 1500 ₴
The Night Watch Restoration	2024-04-05	2024-07-20	Frame repair - 500 ₴

Рисунок 5.9 — звіт по кожній реставрації

Art Gallery			Paintings	Exhibitions	Restorations	Artists
The Starry Night Restoration						
id	work type	price				
1	Cleaning	1500				
3	Color restoration	2000				
2	Frame repair	500				

Рисунок 5. 10 — роботи, що виконувались на кожній реставрації

Add New Restoration

Restoration Name

Start Date

End Date

Painting

Select painting

Restorer

Select restorer

Work Types

Cleaning
Repair
Retouching

Submit

Рисунок 5.11 — форма для додання нових реставрацій

На сторінці Художники (рис. 5.12) представлена інформація про всіх художників, чиї картини присутні в галереї. Створена сторінка, на якій відображаються всі

картини обраного художника (рис. 5.13) та список жанрів, в котрих писав кожен художник (рис 5.14).

Art Gallery					Paintings	Exhibitions	Restorations	Artists
id	full name	birth date	country	painting	genre			
1	Vincent van Gogh	1853-03-30	Netherlands	paintings	genres			
2	Leonardo da Vinci	1452-04-15	Italy	paintings	genres			
3	Pablo Picasso	1881-10-25	Spain	paintings	genres			
4	Rembrandt	1606-07-15	Netherlands	paintings	genres			
5	Michelangelo	1475-03-06	Italy	paintings	genres			
6	Claude Monet	1840-11-14	Netherlands	paintings	genres			
7	Henri Rousseau	1844-05-21	Netherlands	paintings	genres			

Рисунок 5.12 — вигляд сторінки Художника

Art Gallery		Paintings	Exhibitions	Restorations	Artists
Vincent van Gogh					
id	painting	year painted	style		
6	The Persistence of Memory	1931	Fresco		
1	The Starry Night	1889	Oil Painting		

Рисунок 5.13 — список картин обраного художника

Art Gallery		Paintings	Exhibitions	Restorations	Artists
Vincent van Gogh					
id	genre				
1	Landscape				
2	Portrait				
6	Figurative Composition				

Рисунок 5. 14 — список жанрів в яких писав обраний художник

Додаток Б

Сутність художник

```
package com.example.art.data;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.time.LocalDate;
import java.util.LinkedHashSet;
import java.util.Set;

@Getter
@Setter
@Entity
@Table(name = "artist")
public class Artist {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Artist_Id", nullable = false)
    private Integer id;

    @Column(name = "Full_Name", nullable = false)
    private String fullName;

    @Column(name = "Birth_Date", nullable = false)
    private LocalDate birthDate;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "Country_Id", nullable = false)
    private Country country;

    @OneToMany(mappedBy = "artist")
    private Set<Painting> paintings = new LinkedHashSet<>();
}
```

Сутність країна

```
package com.example.art.data;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.ArrayList;
import java.util.LinkedHashSet;
```

```

import java.util.List;
import java.util.Set;

@Getter
@Setter
@Entity
@Table(name = "country")
public class Country {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Country_Id", nullable = false)
    private Integer id;

    @Column(name = "Country_Name", nullable = false)
    private String countryName;

    @OneToMany(mappedBy = "country", orphanRemoval = true)
    private List<Artist> artists = new ArrayList<>();
}

```

Сутність виставка

```

package com.example.art.data;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.time.LocalDate;
import java.util.LinkedHashSet;
import java.util.Set;

@Getter
@Setter
@Entity
@Table(name = "exhibition")
public class Exhibition {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Exhibition_Id", nullable = false)
    private Integer id;

    @Column(name = "Exhibition_Theme", nullable = false)
    private String exhibitionTheme;

    @Column(name = "Price", nullable = false)
    private Integer price;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)

```



```

@JoinColumn(name = "Hall_Id", nullable = false)
private Hall hall;

@Column(name = "Start_Date", nullable = false)
private LocalDate startDate;

@Column(name = "End_Date", nullable = false)
private LocalDate endDate;

@ManyToMany
@JoinTable(name = "painting_exhibition",
    joinColumns = @JoinColumn(name = "Exhibition_Id"),
    inverseJoinColumns = @JoinColumn(name = "Painting_Id"))
private Set<Painting> paintings = new LinkedHashSet<>();
}

```

Сутність виставка

```

package com.example.art.data;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.time.LocalDate;
import java.util.LinkedHashSet;
import java.util.Set;

@Getter
@Setter
@Entity
@Table(name = "exhibition")
public class Exhibition {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Exhibition_Id", nullable = false)
    private Integer id;

    @Column(name = "Exhibition_Theme", nullable = false)
    private String exhibitionTheme;

    @Column(name = "Price", nullable = false)
    private Integer price;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "Hall_Id", nullable = false)
    private Hall hall;

    @Column(name = "Start_Date", nullable = false)

```

```

private LocalDate startDate;

@Column(name = "End_Date", nullable = false)
private LocalDate endDate;

@ManyToMany
@JoinTable(name = "painting_exhibition",
           joinColumns = @JoinColumn(name = "Exhibition_Id"),
           inverseJoinColumns = @JoinColumn(name = "Painting_Id"))
private Set<Painting> paintings = new LinkedHashSet<>();
}

```

Сутність жанр

```

package com.example.art.data;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;

@Getter
@Setter
@Entity
@Table(name = "genre")
public class Genre {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Genre_Id", nullable = false)
    private Integer id;

    @Column(name = "Genre_Name", nullable = false)
    private String genreName;

    @ManyToMany(mappedBy = "genres")
    private Set<Painting> paintings = new LinkedHashSet<>();

    @Override
    public String toString() {
        return genreName;
    }
}

```

Сутність зал

```

package com.example.art.data;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
@Table(name = "hall")
public class Hall {
    @Id
    @Column(name = "Hall_Id", nullable = false)
    private Integer id;

    @Column(name = "Hall_Name", nullable = false)
    private String hallName;

    @Column(name = "Floor", nullable = false)
    private Integer floor;

    @Column(name = "Room_Count", nullable = false)
    private Integer roomCount;
}

```

Сутність картина

```

package com.example.art.data;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;

@Getter
@Setter
@Entity
@Table(name = "painting")
public class Painting {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Painting_Id", nullable = false)

```

```

private Integer id;

@Column(name = "Painting_Title", nullable = false)
private String paintingTitle;

@Column(name = "Year_Painted", nullable = false)
private Integer yearPainted;

@ManyToOne(fetch = FetchType.LAZY, optional = false)
@JoinColumn(name = "Artist_Id", nullable = false)
private Artist artist;

@ManyToOne(fetch = FetchType.LAZY, optional = false)
@JoinColumn(name = "Style_Id", nullable = false)
private Style style;

@ManyToOne(fetch = FetchType.LAZY, optional = false)
@JoinColumn(name = "Supplier_Id", nullable = false)
private Supplier supplier;

@ManyToMany
@JoinTable(name = "painting_genre",
    joinColumns = @JoinColumn(name = "painting_Id"),
    inverseJoinColumns = @JoinColumn(name = "genre_Id"))
private Set<Genre> genres = new LinkedHashSet<>();
}

```

Сутність картина виставка

```

package com.example.art.data;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
@Table(name = "painting_exhibition")
public class
PaintingExhibition {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "Exhibition_Id", nullable = false)
    private Exhibition exhibition;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)

```

```

@JoinColumn(name = "Painting_Id", nullable = false)
private Painting painting;

```

```

}

```

Сутність картина жанр

```

package com.example.art.data;

```

```

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

```

```

@Getter
@Setter
@Entity
@Table(name = "painting_genre")
public class PaintingGenre {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "Painting_Id", nullable = false)
    private Painting painting;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "Genre_Id", nullable = false)
    private Genre genre;
}

```

Сутність реставрація

```

package com.example.art.data;

```

```

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

```

```

import java.time.LocalDate;
import java.util.LinkedHashSet;
import java.util.Set;

```

```

@Getter
@Setter
@Entity
@Table(name = "restoration")
public class Restoration {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Restoration_Id", nullable = false)

```

```

private Integer id;

@Column(name = "Restoration_Name", nullable = false)
private String restorationName;

@Column(name = "Start_Date", nullable = false)
private LocalDate startDate;

@Column(name = "End_Date", nullable = false)
private LocalDate endDate;

@ManyToOne(fetch = FetchType.LAZY, optional = false)
@JoinColumn(name = "Painting_Id", nullable = false)
private Painting painting;

@ManyToOne(fetch = FetchType.LAZY, optional = false)
@JoinColumn(name = "Restorer_Id", nullable = false)
private Restorer restorer;

@ManyToMany
@JoinTable(name = "restoration_work",
           joinColumns = @JoinColumn(name = "Restoration_Id"),
           inverseJoinColumns = @JoinColumn(name = "Work_Type_Id"))
private Set<WorkType> workTypes = new LinkedHashSet<>();
}

```

Сутність реставрація робота

```

package com.example.art.data;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
@Table(name = "restoration_work")
public class RestorationWork {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "Restoration_Id", nullable = false)
    private Restoration restoration;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "Work_Type_Id", nullable = false)
    private WorkType workType;
}

```

```
}
```

Сутність реставратор

```
package com.example.art.data;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import lombok.Getter;
import lombok.Setter;
@Getter
@Setter
@Entity
@Table(name = "restorer")
public class Restorer {
    @Id
    @Column(name = "Restorer_Id", nullable = false)
    private Integer id;

    @Column(name = "Agency_Name", nullable = false)
    private String agencyName;

    @Column(name = "Full_Name", nullable = false)
    private String fullName;

    @Column(name = "Phone_Number", nullable = false)
    private String phoneNumber;

    @Column(name = "Email", nullable = false)
    private String email;
}
```

Сутність стиль

```
package com.example.art.data;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.LinkedHashSet;
import java.util.Set;

@Getter
@Setter
@Entity
@Table(name = "style")
public class Style {
```

```

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Column(name = "Style_Id", nullable = false)
private Integer id;

@Column(name = "Technique", nullable = false)
private String technique;

@OneToMany(mappedBy = "style")
private Set<Painting> paintings = new LinkedHashSet<>();
}

```

Сутність постачальник

```

package com.example.art.data;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.Setter;

import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.List;
import java.util.Set;

@Getter
@Setter
@Entity
@Table(name = "supplier")
public class Supplier {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "Supplier_Id", nullable = false)
    private Integer id;

    @Column(name = "Supplier_Name", nullable = false)
    private String supplierName;

    @Column(name = "Full_Name", nullable = false)
    private String fullName;

    @Column(name = "Phone_Number", nullable = false)
    private String phoneNumber;

    @Column(name = "Email", nullable = false)
    private String email;

    @OneToMany(mappedBy = "supplier")
    private List<Painting> paintings = new ArrayList<>();
}

```



```
}
```

Сутність вид роботи

```
package com.example.art.data;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@Entity
@Table(name = "work_type")
public class WorkType {
    @Id
    @Column(name = "Work_Type_Id", nullable = false)
    private Integer id;

    @Column(name = "Work_Type_Name", nullable = false)
    private String workTypeName;

    @Column(name = "Price", nullable = false)
    private Integer price;
}
```

Клас репозиторій GenreRepository.java

```
package com.example.art.repository;

import com.example.art.data.Genre;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.util.Collection;
import java.util.List;

public interface GenreRepository extends JpaRepository<Genre, Integer> {

    @Query("select g from Genre g inner join g.paintings paintings where paintings.artist.id = ?1")
    List<Genre> findByPaintingsArtistId(Integer id);

    @Query("SELECT g FROM Genre g WHERE g.id NOT IN (SELECT pg.genre.id FROM PaintingGenre pg WHERE pg.painting.id = :paintingId)")
}
```

```

        List<Genre> findGenresNotInPainting(@Param("paintingId") int paintingId);

        @Query("SELECT DISTINCT g FROM Genre g " +
                "JOIN FETCH g.paintings p " +
                "JOIN FETCH p.artist a")
        List<Genre> findAllGenresWithArtistsAndPaintings();
    }

```

Класс PaintingService.java

```

package com.example.art.service;

import com.example.art.data.*;
import com.example.art.repository.GenreRepository;
import com.example.art.repository.PaintingRepository;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class PaintingService {
    private final GenreRepository genreRepository;
    private PaintingRepository paintingRepository;
    private GenreService genreService;
    public List<Painting> findAll() {
        return paintingRepository.findAll();
    }
    public Painting find(int id) {
        Optional<Painting> byId = paintingRepository.findById(id);
        return byId.orElse(new Painting());
    }

    public void addPainting(String title, int year, Artist artist,
        Style style, List<Integer> genres, Supplier supplier) {
        Painting painting = new Painting();
        List<Genre> allById = genreService.findAllGenreById(genres);
        painting.setPaintingTitle(title);
        painting.setYearPainted(year);
        painting.setArtist(artist);
        painting.setStyle(style);
        painting.setSupplier(supplier);
        paintingRepository.save(painting);
        painting.getGenres().addAll(allById);
        paintingRepository.save(painting);
    }
}

```

```

        public List<Painting> findAllPaintingById(List<Integer> ids) {
            return paintingRepository.findAll().stream().filter(painting ->
ids.contains(painting.getId())).toList();
        }

        public List<Genre> findGenresNotInPainting(int id) {
            return genreRepository.findGenresNotInPainting(id);
        }
    }

```

Класс PaintingController.java

```

package com.example.art.controller;

import com.example.art.data.*;
import com.example.art.repository.GenreRepository;
import com.example.art.repository.PaintingRepository;
import com.example.art.service.PaintingService;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestParam;

import java.util.List;

@Controller
@AllArgsConstructor
public class PaintingController {
    private PaintingService paintingService;

    @GetMapping("/painting")
    public String paintings(Model model) {
        model.addAttribute("paintings", paintingService.findAll());
        return "painting";
    }

    @GetMapping("/painting_genre")
    public String paintingGenre(@RequestParam int id, Model model) {
        Painting painting = paintingService.find(id);
        List<Genre> genres = paintingService.findGenresNotInPaint-
ing(id);
        model.addAttribute("painting", painting);
        model.addAttribute("genres", genres);
        return "painting_genre";
    }

    @GetMapping("/add_painting")

```

```

public String addPainting() {
    return "add_painting";
}
@GetMapping("/addPainting")
public String addPainting(@RequestParam String title,
                          @RequestParam int year,
                          @RequestParam Artist artist,
                          @RequestParam Style style,
                          @RequestParam List<Integer> genre,
                          @RequestParam Supplier supplier) {
    paintingService.addPainting(title, year, artist, style, genre, sup-
plier);
    return "redirect:/painting";
}
@GetMapping("/painting/delete_painting/{id}")
public String deletePainting(@PathVariable("id") int id) {
    paintingService.deleteById(id);
    return "redirect:/painting";
}
}

```

Файл artist.html

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>artist</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integ-
rity="sha384-QWTKZyjpPE-
jISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMHjY6hW+ALEwIH" cros-
sorigin="anonymous">
    <link rel="stylesheet" href="/styles.css">
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-custom">
    <div class="container-fluid">
        <a class="navbar-brand" href="/"><i class="bi bi-house menu-
item">Art Gallery</i></a>
        <button class="navbar-toggler" type="button" data-bs-tog-
gle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav"
aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ms-auto">
                <li class="nav-item">
                    <a class="nav-link" href="painting"><div

```

```

class="menu-item">Paintings</div></a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="exhibition"><div
class="menu-item">Exhibitions</div></a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="restoration"><div
class="menu-item">Restorations</div></a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="artist"><div class="menu-
item">Artists</div></a>
    </li>
</ul>
</div>
</div>
</nav>
<table class="table table-striped">
    <thead class="table-dark">
        <tr>
            <th>id</th>
            <th>full name</th>
            <th>birth date</th>
            <th>country</th>
            <th>painting</th>
            <th>genre</th>
        </tr>
    </thead>
    <tbody>
        <tr th:each="a:${artists}">
            <td th:text="${a.id}"></td>
            <td th:text="${a.fullName}"></td>
            <td th:text="${a.birthDate}"></td>
            <td th:text="${a.country.countryName}"></td>
            <td><a class="btn btn-outline-primary" th:href="@{art-
ist_painting(id=${a.id})}">paintings</a></td>
            <td><a class="btn btn-outline-primary" th:href="@{art-
ist_genre(id=${a.id})}">genres</a></td>
        </tr>
    </tbody>
</table>
</body>
</html>

```

Класс GenreService.java

```

package com.example.art.service;
import com.example.art.data.Genre;
import com.example.art.repository.GenreRepository;

```

```

import lombok.AllArgsConstructor;
import org.springframework.stereotype.Service;
import java.util.List;
@AllArgsConstructor
@Service
public class GenreService {
    private GenreRepository genreRepository;
    public List<Genre> findAllGenres() {
        return genreRepository.findAll();
    }

    public List<Genre> findAllGenreById(List<Integer> ids) {
        return genreRepository.findAll().stream().filter(genre ->
ids.contains(genre.getId())).toList();
    }
    public List<Genre> getAllGenresWithArtistsAndPaintings() {
        return genreRepository.findAllGenresWithArtistsAndPaintings();
    }
}

```

Класс GenreController.java

```

package com.example.art.controller;
import com.example.art.data.Genre;
import com.example.art.repository.GenreRepository;
import org.springframework.ui.Model;
import com.example.art.service.GenreService;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import java.util.List;
@AllArgsConstructor
@Controller
public class GenreController {
    private GenreService;
    private GenreRepository genreRepository;
    @GetMapping("/genre")
    public String genre(Model model) {
        model.addAttribute("genre", genreService.findAllGenres());
        return "genre";
    }
    @GetMapping("/genreReport")
    public String getGenreReport(Model model) {
        List<Genre> genres = genreService.getAllGenresWithArtistsAnd-
Paintings();
        model.addAttribute("genres", genres);
        return "report_genre";
    }
}

```

Класс ArtistController.java

```
package com.example.art.controller;

import com.example.art.data.Artist;
import com.example.art.data.Genre;
import com.example.art.service.ArtistService;
import lombok.AllArgsConstructor;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

import java.util.List;

@Controller
@AllArgsConstructor
public class ArtistController {
    private ArtistService artistService;
    @GetMapping("/artist")
    public String artist(Model model) {
        model.addAttribute("artists", artistService.findAll());
        return "artist";
    }
    @GetMapping("/artist_painting")
    public String artist_painting(@RequestParam int id, Model model) {
        Artist a = artistService.find(id);
        model.addAttribute("artist", a);
        return "artist_painting";
    }
    @GetMapping("/artist_genre")
    public String artistGenre(@RequestParam int id, Model model) {
        List<Genre> genres = artistService.findGenresByArtistId(id);
        model.addAttribute("genres", genres);
        model.addAttribute("artist", artistService.find(id));
        return "artist_genre";
    }
}
```

Класс ExhibitionController.java

```
package com.example.art.controller;

import com.example.art.data.Exhibition;
import com.example.art.data.Hall;
import com.example.art.repository.ExhibitionRepository;
import com.example.art.service.ExhibitionService;
import com.example.art.service.HallService;
import com.example.art.service.PaintingService;
import lombok.AllArgsConstructor;
import org.springframework.format.annotation.DateTimeFormat;
```

```

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.time.LocalDate;
import java.util.List;

@Controller
public class ExhibitionController {
    private final ExhibitionRepository exhibitionRepository;
    private ExhibitionService exhibitionService;
    private HallService hallService;
    private PaintingService paintingService;
    @GetMapping("/exhibition")
    public String exhibition(Model model) {
        model.addAttribute("exhibitions", exhibitionService.findAll());
        return "exhibition";
    }

    @GetMapping("/exhibitions_painting")
    public String exhibition_painting(@RequestParam int id, Model
model) {
        Exhibition e = exhibitionService.find(id);
        model.addAttribute("exhibition", e);
        return "exhibitions_painting";
    }

    @GetMapping("/exhibition/delete_exhibition/{id}")
    public String deleteRestoration(@PathVariable("id") int id){
        exhibitionRepository.deleteById(id);
        return "redirect:/exhibition";
    }

    @GetMapping("/add_exhibition")
    public String addExhibition(Model model) {
        model.addAttribute("halls", hallService.getAllHalls());
        model.addAttribute("paintings", paintingService.findAll());
        return "add_exhibition";
    }

    @PostMapping("/addExhibition")
    public String addExhibition(@RequestParam String theme,
                                @RequestParam int price,
                                @RequestParam Hall hall,
                                @RequestParam @DateTimeFormat(iso =
DateTimeFormat.ISO.DATE) LocalDate startDate,
                                @RequestParam @DateTimeFormat(iso =

```



```

DateFormat.ISO.DATE) LocalDate endDate,
                                @RequestParam List<Integer> painting) {
    exhibitionService.addExhibition(theme, price, hall, startDate,
endDate, painting);
    return "redirect:/exhibition";
}
}

```

Класс RestorationController.java

```

package com.example.art.controller;

import com.example.art.data.Painting;
import com.example.art.data.Restoration;
import com.example.art.data.Restorer;
import com.example.art.service.PaintingService;
import com.example.art.service.RestorationService;
import com.example.art.service.RestorerService;
import com.example.art.service.WorkTypeService;
import lombok.AllArgsConstructor;
import org.springframework.format.annotation.DateFormat;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

import java.time.LocalDate;
import java.util.List;

@Controller
public class RestorationController {
    private RestorationService restorationService;
    private PaintingService paintingService;
    private WorkTypeService workTypeService;
    private RestorerService restorerService;

    @GetMapping("/restoration")
    public String restoration(Model model) {
        model.addAttribute("restorations",
restorationService.findAll());
        return "restoration";
    }
    @GetMapping("/restoration_work")
    public String restoration_work(@RequestParam int id, Model model) {
        Restoration r = restorationService.findById(id);
        model.addAttribute("restoration", r);
        return "restoration_work";
    }
}

```

```

    }
    @GetMapping("/restoration-report")
    public String restorationReport(Model model) {
        List<Restoration> restorations = restorationService.findAll();
        double totalCostAll =
restorationService.calculateTotalCostAll(restorations);
        model.addAttribute("restorations", restorations);
        model.addAttribute("totalCostAll", totalCostAll);
        return "report_restoration";
    }

    @GetMapping("/restoration/delete_restoration/{id}")
    public String deleteRestoration(@PathVariable("id") int id){
        restorationService.deleteById(id);
        return "redirect:/restoration";
    }

    @GetMapping("/add_restoration")
    public String showAddRestorationForm(Model model) {
        model.addAttribute("paintings", paintingService.findAll());
        model.addAttribute("restorers", restorerService.findAll());
        model.addAttribute("works", workTypeService.findAll());

        return "add_restoration";
    }

    @PostMapping("/addRestoration")
    public String addRestoration(@RequestParam String restorationName,
                                @RequestParam @DateTimeFormat(iso =
DateTimeFormat.ISO.DATE) LocalDate startDate,
                                @RequestParam @DateTimeFormat(iso =
DateTimeFormat.ISO.DATE) LocalDate endDate,
                                @RequestParam Painting painting,
                                @RequestParam Restorer restorer,
                                @RequestParam List<Integer> workTypes)
    {
        restorationService.addRestoration(restorationName, startDate,
endDate, painting, restorer, workTypes);
        return "redirect:/restoration";
    }
}

```

Додаток В

SQL код для створення бази даних

```
CREATE DATABASE IF NOT EXISTS `art_gallery`;
USE `art_gallery`;
CREATE TABLE IF NOT EXISTS `country` (
  `Country_Id` INT NOT NULL AUTO_INCREMENT,
  `Country_Name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`Country_Id`)
);
CREATE TABLE IF NOT EXISTS `artist` (
  `Artist_Id` INT NOT NULL AUTO_INCREMENT,
  `Full_Name` VARCHAR(255) NOT NULL,
  `Birth_Date` DATE NOT NULL,
  `Country_Id` INT NOT NULL,
  PRIMARY KEY (`Artist_Id`),
  FOREIGN KEY (`Country_Id`) REFERENCES `country` (`Country_Id`)
);
CREATE TABLE IF NOT EXISTS `genre` (
  `Genre_Id` INT NOT NULL AUTO_INCREMENT,
  `Genre_Name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`Genre_Id`)
);
CREATE TABLE IF NOT EXISTS `style` (
  `Style_Id` INT NOT NULL AUTO_INCREMENT,
  `Technique` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`Style_Id`)
);
CREATE TABLE IF NOT EXISTS `supplier` (
  `Supplier_Id` INT NOT NULL AUTO_INCREMENT,
  `Supplier_Name` VARCHAR(255) NOT NULL,
  `Full_Name` VARCHAR(255) NOT NULL,
  `Phone_Number` VARCHAR(255) NOT NULL,
  `Email` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`Supplier_Id`)
);
CREATE TABLE IF NOT EXISTS `restorer` (
  `Restorer_Id` INT NOT NULL AUTO_INCREMENT,
  `Agency_Name` VARCHAR(255) NOT NULL,
  `Full_Name` VARCHAR(255) NOT NULL,
  `Phone_Number` VARCHAR(255) NOT NULL,
```

```

    `Email` VARCHAR(255) NOT NULL,
    PRIMARY KEY (`Restorer_Id`)
);
CREATE TABLE IF NOT EXISTS `painting` (
    `Painting_Id` INT NOT NULL AUTO_INCREMENT,
    `Painting_Title` VARCHAR(255) NOT NULL,
    `Year_Painted` INT NOT NULL,
    `Artist_Id` INT NOT NULL,
    `Style_Id` INT NOT NULL,
    `Supplier_Id` INT NOT NULL,
    PRIMARY KEY (`Painting_Id`),
    FOREIGN KEY (`Artist_Id`) REFERENCES `artist`(`Artist_Id`),
    FOREIGN KEY (`Style_Id`) REFERENCES `style`(`Style_Id`),
    FOREIGN KEY (`Supplier_Id`) REFERENCES `supplier`(`Supplier_Id`)
);
CREATE TABLE IF NOT EXISTS `hall` (
    `Hall_Id` INT NOT NULL AUTO_INCREMENT,
    `Hall_Name` VARCHAR(255) NOT NULL,
    `Floor` INT NOT NULL,
    `Room_Count` INT NOT NULL,
    PRIMARY KEY (`Hall_Id`)
);
CREATE TABLE IF NOT EXISTS `exhibition` (
    `Exhibition_Id` INT NOT NULL AUTO_INCREMENT,
    `Exhibition_Theme` VARCHAR(255) NOT NULL,
    `Hall_Id` INT NOT NULL,
    `Start_Date` DATE NOT NULL,
    `End_Date` DATE NOT NULL,
    PRIMARY KEY (`Exhibition_Id`),
    FOREIGN KEY (`Hall_Id`) REFERENCES `hall` (`Hall_Id`)
);
CREATE TABLE IF NOT EXISTS `work_type` (
    `Work_Type_Id` INT NOT NULL AUTO_INCREMENT,
    `Work_Type_Name` VARCHAR(255) NOT NULL,
    `Price` INT NOT NULL,
    PRIMARY KEY (`Work_Type_Id`)
);
CREATE TABLE IF NOT EXISTS `restoration` (
    `Restoration_Id` INT NOT NULL AUTO_INCREMENT,
    `Restoration_Name` VARCHAR(255) NOT NULL,
    `Start_Date` DATE NOT NULL,
    `End_Date` DATE NOT NULL,

```

```

    `Painting_Id` INT NOT NULL,
    `Restorer_Id` INT NOT NULL,
    PRIMARY KEY (`Restoration_Id`),
    FOREIGN KEY (`Painting_Id`) REFERENCES `painting` (`Painting_Id`),
    FOREIGN KEY (`Restorer_Id`) REFERENCES `restorer` (`Restorer_Id`)
);
CREATE TABLE IF NOT EXISTS `restoration_work` (
    `Restoration_Id` INT NOT NULL,
    `Work_Type_Id` INT NOT NULL,
    -- PRIMARY KEY (`Restoration_Id`, `Work_Type_Id`),
    FOREIGN KEY (`Restoration_Id`) REFERENCES `restoration` (`Restoration_Id`),
    FOREIGN KEY (`Work_Type_Id`) REFERENCES `work_type` (`Work_Type_Id`)
);
CREATE TABLE IF NOT EXISTS `painting_genre` (
    `Painting_Id` INT NOT NULL,
    `Genre_Id` INT NOT NULL,
    -- PRIMARY KEY (`Painting_Id`, `Genre_Id`),
    FOREIGN KEY (`Painting_Id`) REFERENCES `painting` (`Painting_Id`),
    FOREIGN KEY (`Genre_Id`) REFERENCES `genre` (`Genre_Id`)
);
CREATE TABLE IF NOT EXISTS `painting_exhibition` (
    `Exhibition_Id` INT NOT NULL,
    `Painting_Id` INT NOT NULL,
    -- PRIMARY KEY (`Exhibition_Id`, `Painting_Id`),
    FOREIGN KEY (`Exhibition_Id`) REFERENCES `exhibition` (`Exhibition_Id`),
    FOREIGN KEY (`Painting_Id`) REFERENCES `painting` (`Painting_Id`)
);

```