МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний технічний університет
«Харківський політехнічний інститут»
Кафедра «Програмної інженерії та
інформаційних технологій управління»

Лабораторна робота №1

з курсу

«Економіка та організація виробництва систем»
за темою «Розробка вимог»

Перевірила: стар. вик.
Єршова С.І.
Виконав: студент КН-218б
Парахін З.Г.

Харків 2022

Мета роботи: Отримати практичні навички моделювання програмних вимог.

Хід роботи

1. Ознайомитися із методичними рекомендаціями.
2. Проаналізувати функції.
3. Розробити вимоги щодо програмного забезпечення. Визначити вимоги до інтерфейсу.
4. Задокументувати розроблені вимоги, використовуючи Visual Paradigm for UML (за допомогою Requirement diagram).
5. Оформити звіт про виконання лабораторної роботи.
6. Оформити звіт про виконання лабораторної роботи.

Предметна область проекту –оренда залів фотостудії.

Оренда залів фотостудії – це процес, що забезпечує завчасне резервування певної технічно-матеріальної бази підприємства на певний проміжок часу. Це досить швидкий і зручний спосіб резервації, що надає можливість заздалегідь закріпити за певним клієнтом його побажань у вигляді залів та технічного обладнання. Так як ця галузь користується великим попитом, а тому вона має швидке зростання, тому що надає своїм клієнтам сучасну технічну базу, кваліфіковану допомогу для реалізації професійної фото та відео зйомки.

У світі існує безліч компаній, що надають послуги оренди залів, тому ми аналізуємо українські компанії, а саме : «LightfieldProduction» , «R18 Professionalphotostudio» та «InLight». Вони всі використовують онлайн-сервіси для прийому та оформлення послуги.

Споживач, коли заходить на сайт, може детально ознайомитись з інформацією щодо доступного асортименту залів та їх характеристик тієї чи іншої компанії, з інформацією про компанію та правилами резервування залів, після цього він залишає форму замовлення та очікує коли з ним з’єднається менеджер для подальшого оформлення послуги. Це зв’язано безпосередньо з важкістю процесу контролю робочого часу залів та студії під час оформлення замовлень на оренду залів фотостудії.

На основі аналізу предметної області, визначено, що до Web-ресурсів, що реалізують інтерфейс доступу клієнтів до послуг оренди залів, виставляються

такі вимоги:

- наявність детальної інформації про технічне забезпечення залів, їх розміщення відносно сонця, наявність вікон, рівень освітленості, розміри та інше;
- наявність портфоліо виконаних робіт саме у цій студії для зацікавлення користувача;
- кількість дій користувача для здійснення оренди повинно бути мінімальним;
- наявність спеціалізованого календаря, що відображає актуальну інформацію про зайнятий час для кожного залу підприємства.

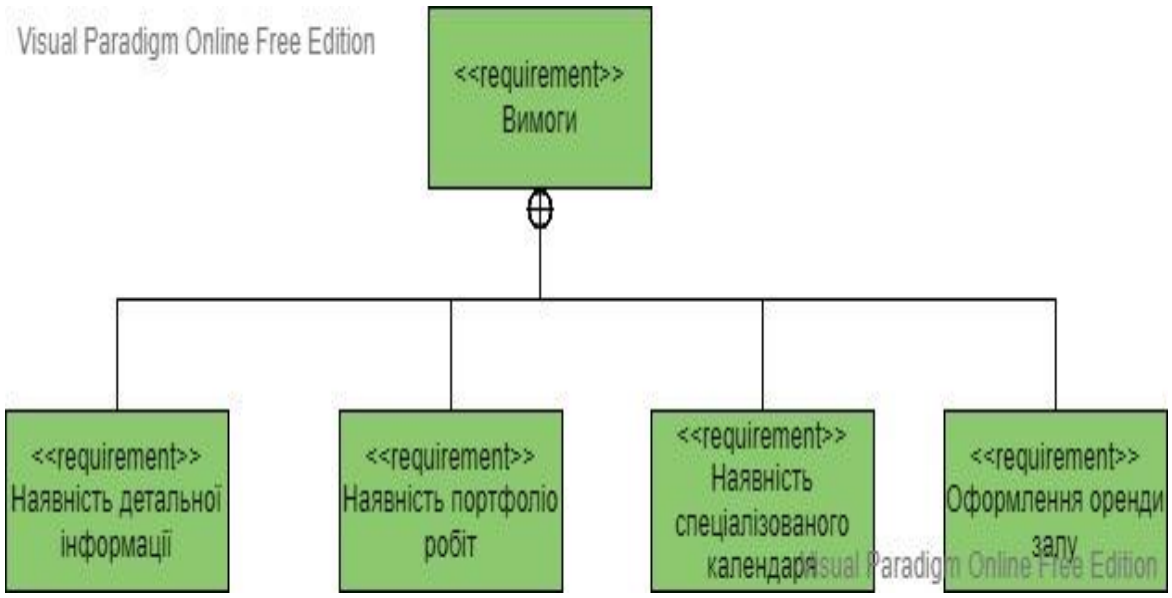


Рисунок 1 - Requirement diagram

Споживач, коли заходить на сайт, може детально ознайомитись з інформацією щодо доступного асортименту залів та їх характеристик, з інформацією про компанію та правилами резервування залів, після цього він реєструється, залишає форму замовлення та очікує коли з ним з'єднається менеджер для подальшого оформлення послуги. Це зв'язано безпосередньо з важкістю процесу контролю робочого часу залів та студії під час оформлення замовлень на оренду залів фотостудії. Основною метою створення інформаційної системи є автоматизація певних бізнес-процедур і бізнес-функцій, що виконуються при взаємодії користувача з системою.

З точки зору інформаційної системи, користувачі поділяються:

- незареєстрований користувач (гість);
- зареєстрований користувач;
- адміністратор системи.

Процедура ідентифікації користувача потрібна, для розділення прав та можливостей різних типів користувачів, а також подальшої роботи с клієнтами для спонукання створення нових замовлень шляхом створення програм лояльності, надання знижок та створення акцій на основі наданих особистих даних клієнтами.Клієнтська частина являє собою багатосторінковий сайт. Сайт повен бути зрозумілим для користувача. Вміст сайту має відповідати бізнес-вимогам, контент повинен бути унікальним. Навігація сайту повинна бути простою і зрозумілою, так як це позитивно впливає на конверсію.

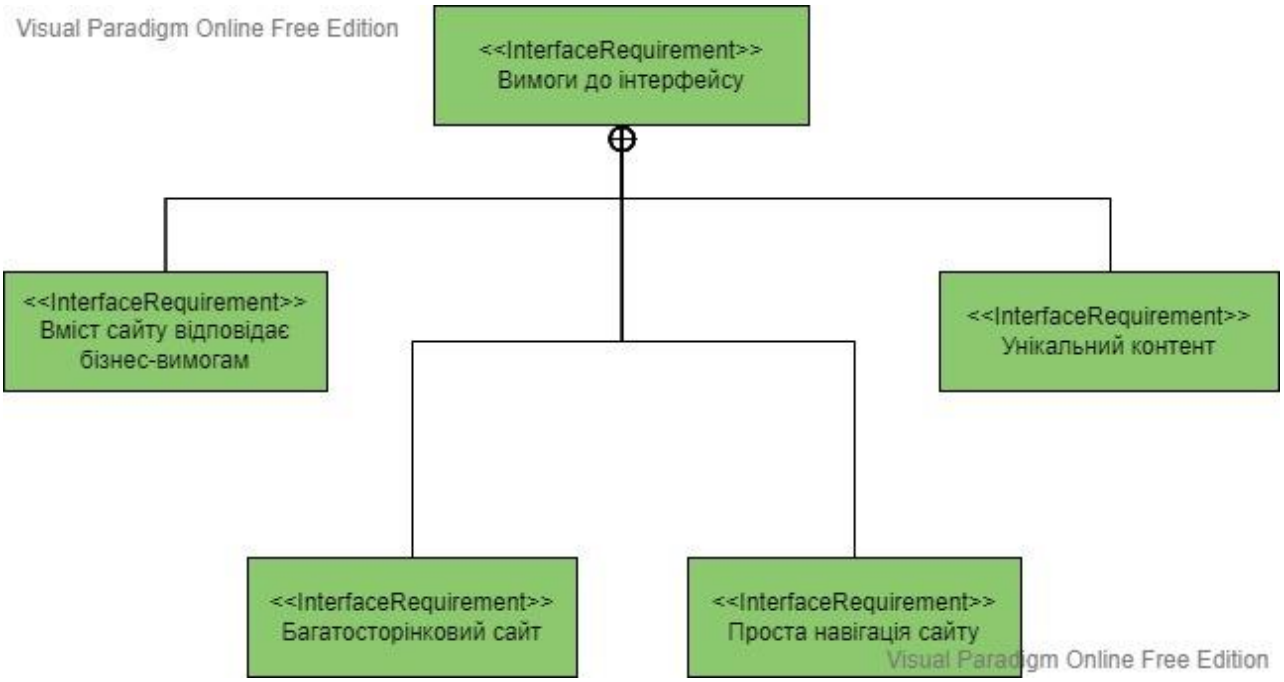


Рисунок 2 - Requirement diagram для інтерфейсу

Перед обличчям користувача зразу спливає кнопка замовлення залів. Ті хто прийшовши вперше перейдуть нижче та прочитають про компанію, а досвідчені юзери цього сайту перейдуть зразу ж на замовлення . У верхній частині сторінки відображено навігаційну панель, за допомогою якої користувач може :

Продивитися головну сторінку, рухаючись по міткам

Зробити замовлення

Висновок: шляхом виконання завдань лабораторної роботи були здобуті навички моделювання вимог, розроблені вимоги і створені відповідні діаграми та опис.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний технічний університет
«Харківський політехнічний інститут»
Кафедра «Програмної інженерії та
інформаційних технологій управління»

Лабораторна робота №2

з курсу

«Економіка та організація виробництва систем»
за темою «Розробка проекту. Розробка архітектури»

Перевірила: стар. вик.
Єршова С.І.
Виконав: студент КН-218б
Парахін З.Г.

Харків 2022

Мета роботи:

Отримати практичні навички архітектурного проектування програмного забезпечення.

Хід роботи

1. Визначити стратегію проектування для реалізації програмного забезпечення.
2. Розробити архітектуру програмного забезпечення.
3. Оформити звіт про виконання лабораторної роботи.

Реалізований проект зможе надати споживачам послуг компанії NewStandart використання онлайн форми для швидкого замовлення послуги з вказанням всіх доступних особливостей та подальшим відстеженням статусу замовлення.Для персоналу компанії проект зможе надати інструментарій для автоматизації взаємодії з клієнтом, його інформування та обробки даних. Основна ціль проекту – швидке вирішення проблеми організації роботи компанії.

Архітектура програмного забезпечення (англ. *software architecture*) — абстракція елементів системи на певній фазі її роботи. Проект будується на основі моделі «TO-BE» і включає в себе функціональну модель майбутньої системи згідно стандарту IDEF0.

Методологія IDEF0 розглядає нашу систему у вигляді структури функціональних блоків. Результатом цієї методології є модель, що складається з діаграм, фрагментів текстів і глосарію, що мають посилання один на одного.

Діаграма – це головний компонент моделі, усі функції система та інтерфейси поділяються на дуги і блоки. Також існує керуюча інформація, яка входить в блок зверху, вхідні дані - зліва, а вихідні – справа. Механізм – це якийсь об’єкт який здійснює операцію, який зображений дугами, які входять знизу.

Функціональна модель зображена на рисунку 1.

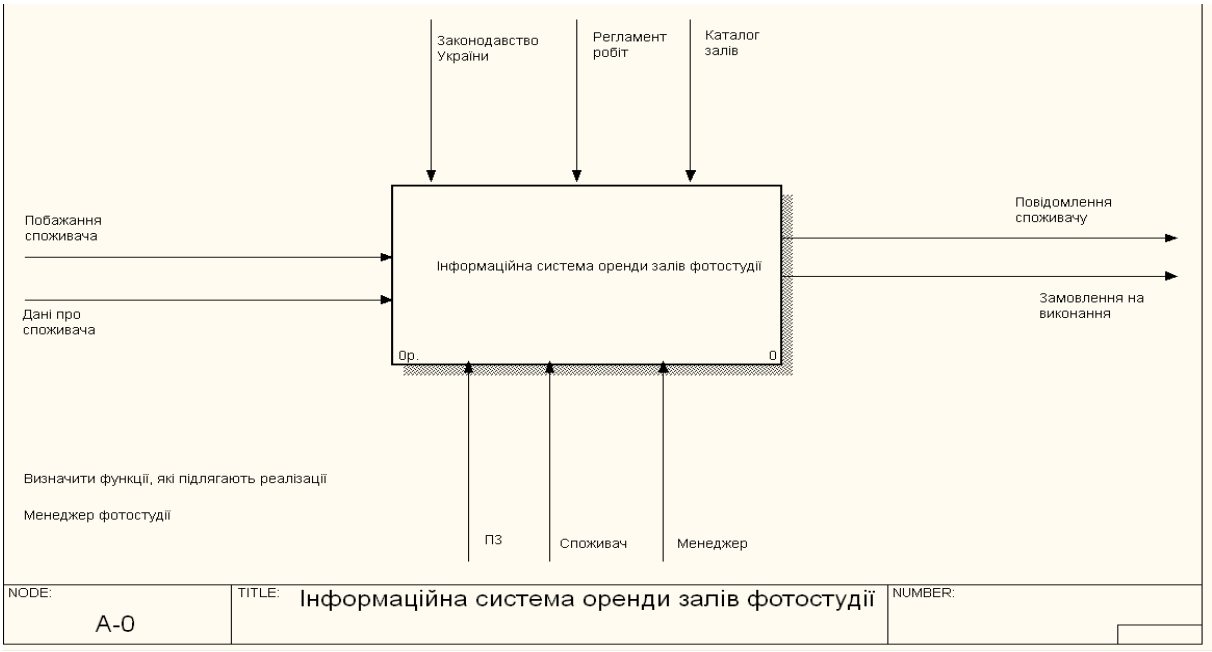


Рисунок 1 – Концептуальна діаграма (A0)

Вхідні дані : побажання споживача, дані про споживача

Вихідні дані : замовлення на виконання, повідомлення споживачу

Управління : каталог залів, законодавство України та регламент робіт

Механізми : споживач, менеджер, програмне забезпечення

Для більш детального зображення створюється декомпозиція у вигляді ієрархії діаграм. Кожна детальна діаграма є декомпозицією батьківського блоку.

Діаграма декомпозиції, яка зображена на рисунку 2 складається з 4 функціональних блоків:

Обробити замовлення

Сформувати замовлення

Зв’язатися з споживачем

Змінити статус замовлення

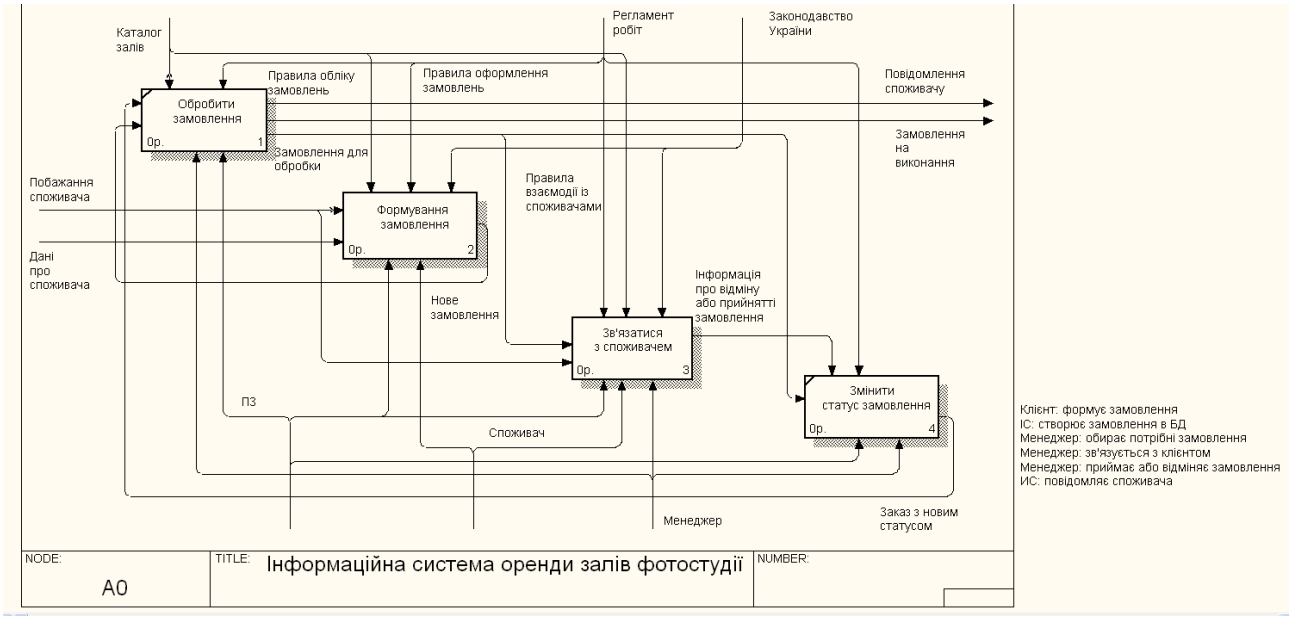


Рисунок 2 – Концептуальна діаграма (A0)

Далі кожен блок може бути декомпозовано. Детальні діаграми рівнів A2-A3 представлені на рисунку 3-4

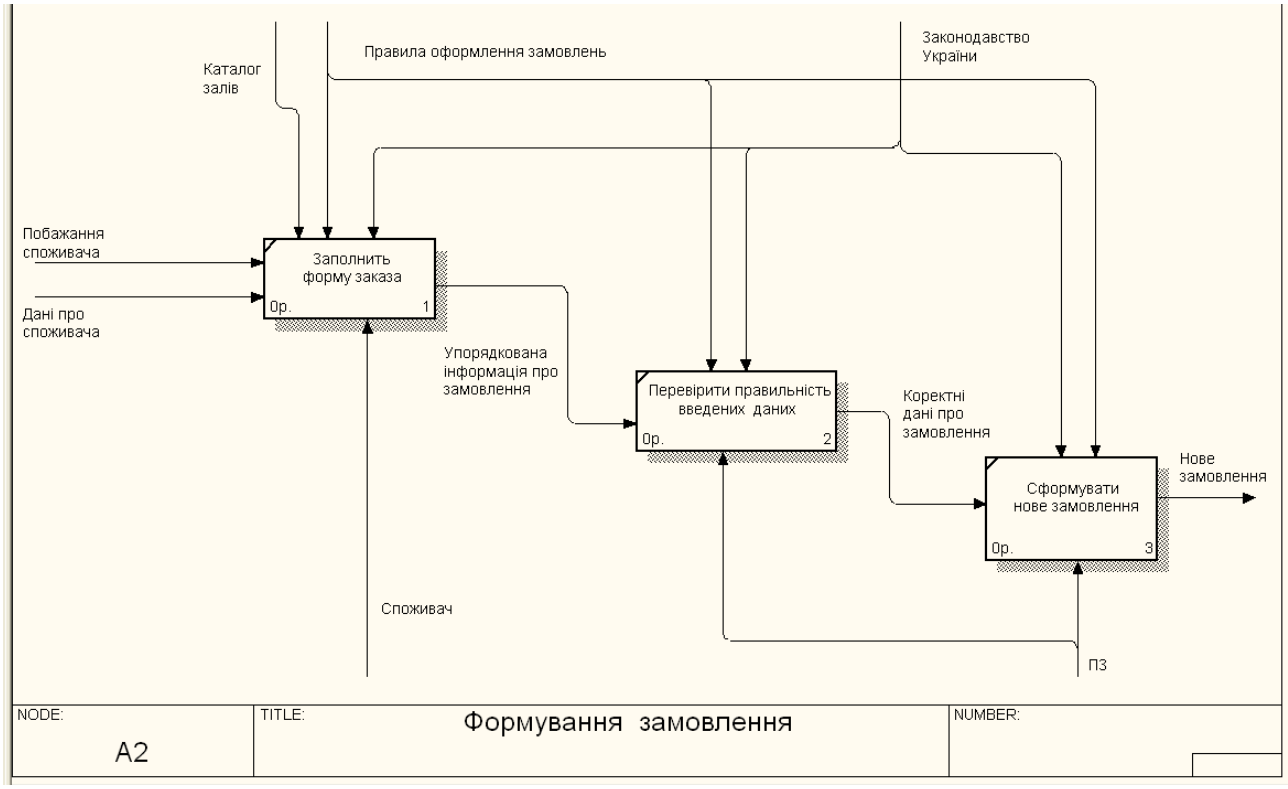


Рисунок 3 – Діаграма декомпозиції «Формування замовлення»(A2)

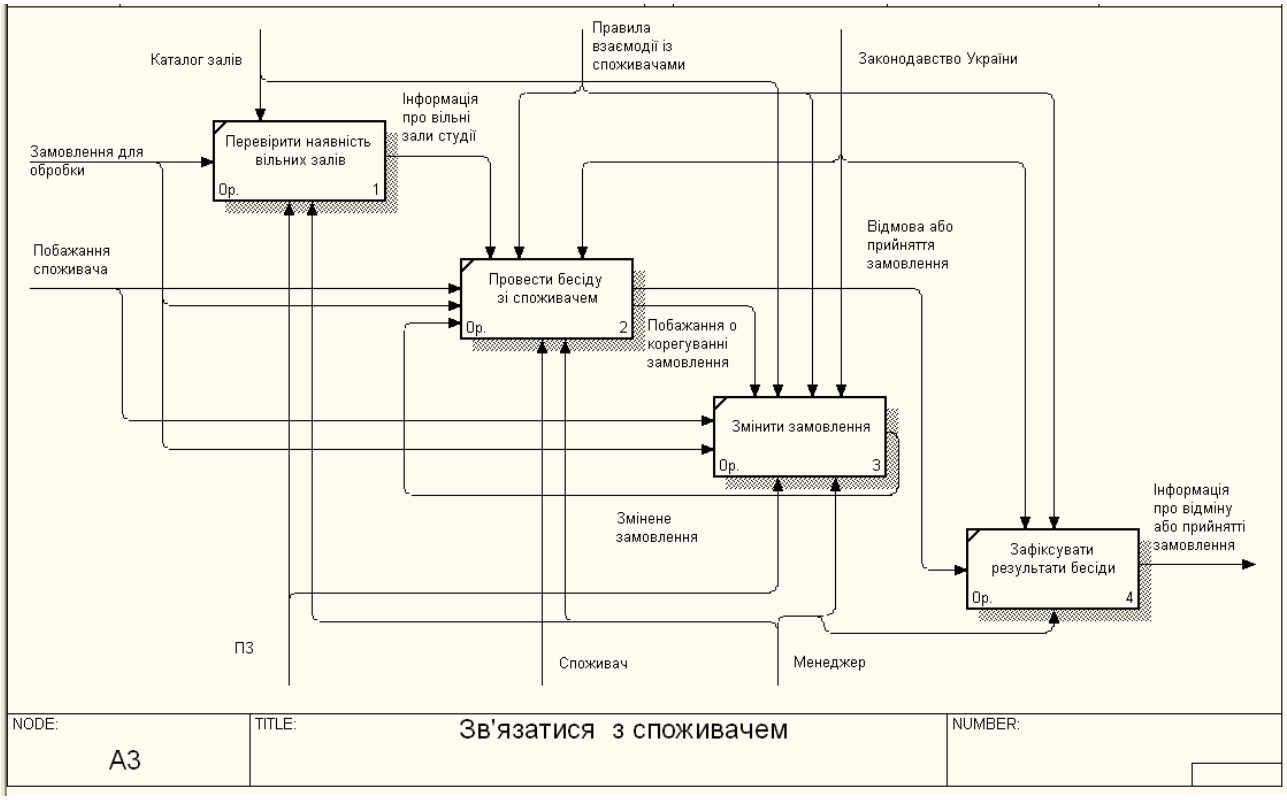


Рисунок 4 – Діаграма декомпозиції «зв’язатися з споживачем»(A3)

Після всіх декомпозицій ми отримаємо дерево вузлів, яке демонструє ієрархію декомпозиції :

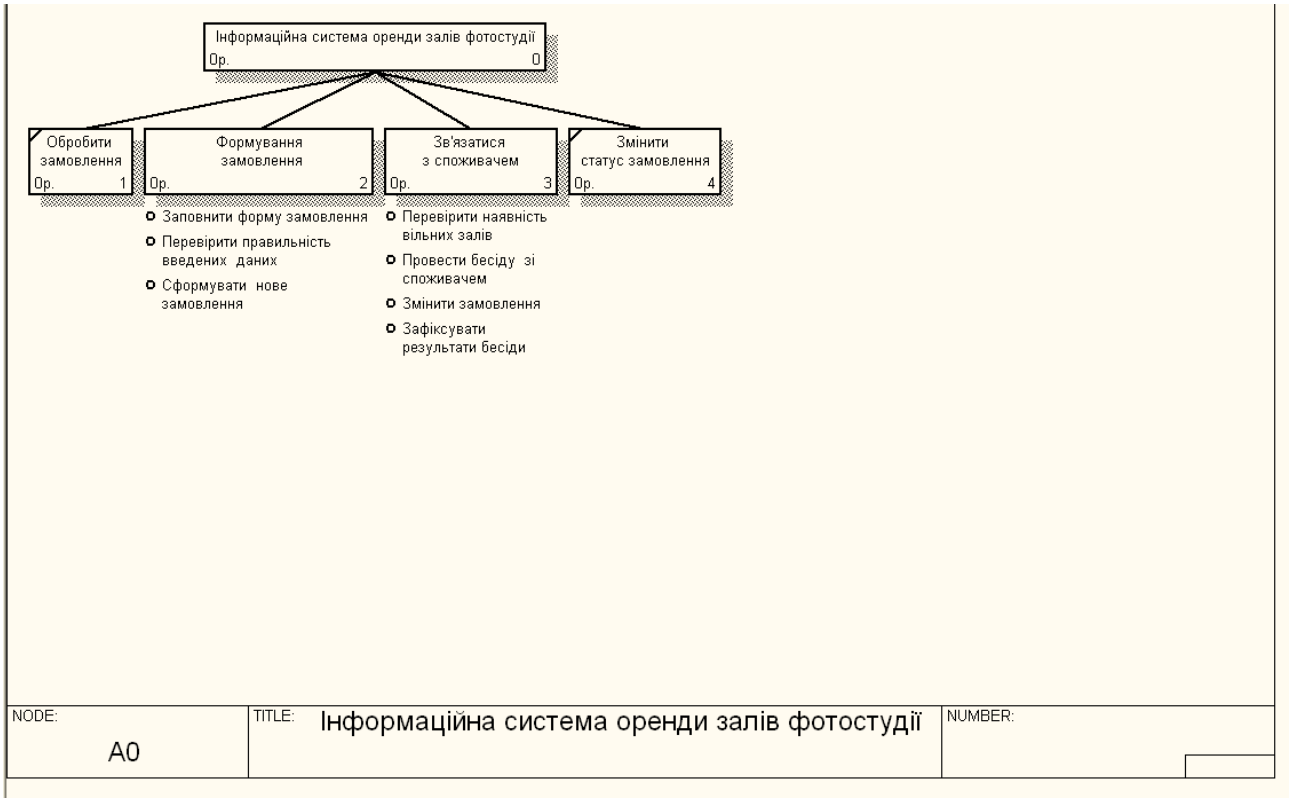


Рисунок 5 – Діаграма дерева вузлів

UML(UnifiedModelingLanguage) – уніфікована мова об’єктно-орієнтованого моделювання, використовується у парадигмі об’єктно-орієнтованого програмування. Є невід’ємною частиною уніфікованого процесу розробки програмного забезпечення. Головною метою є представлення системи у такому вигляді, в якому її можна було легко перевести в програмний код.

1.1.1 UseCaseDiagram(Діаграма прецедентів)

Діаграма прецедентів – це діаграма, на якій зображено відношення між акторами та прецедентами в системі.

Діаграма прецедентів є графом, що складається з множини акторів(об’єкти які якось взаємодіють з нашою системою), прецедентів(дій над системою) обмежених границею системи(прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів та відношень узагальнення між акторами

В нашій діаграмі кожен актор має певну роль, розрізняють таких акторів:

Споживач
Менеджер

На рисунку 6 зображено діаграму прецедентів для інформаційної системи оренди залів фотостудії. На цій діаграмі ми бачимо усі функції доступні акторам та взаємодія їх між собою

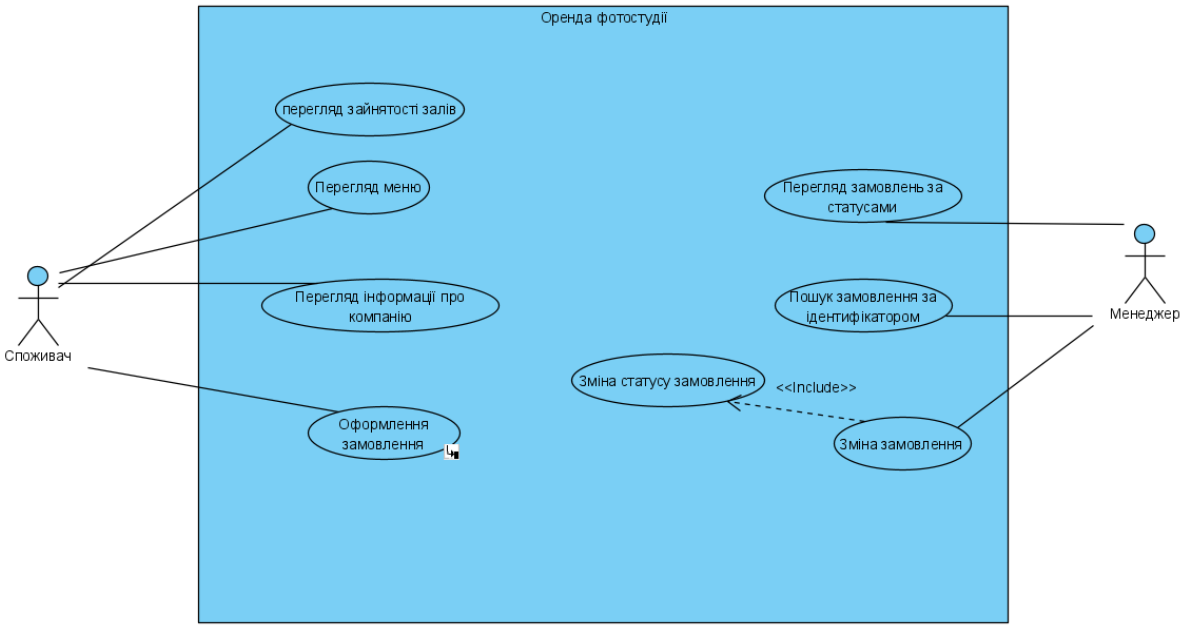


Рисунок 6 – Діаграма прецедентів

1.1.2 Sequencediagram(діаграма послідовності дій)

Діаграма послідовності – відображає взаємодію об’єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об’єкти та послідовність відправлених повідомлень.

Діаграма послідовності є наступним етапом після створення діаграми прецедентів, це робиться для уточнення та більш детального опису логіки сценаріїв використання.

За допомогою діаграми послідовності дій можна відобразити життєвий цикл кожного обраного прецеденту.

Діаграма послідовності оформлення замовлення представлено на рисунку 7 – 8.

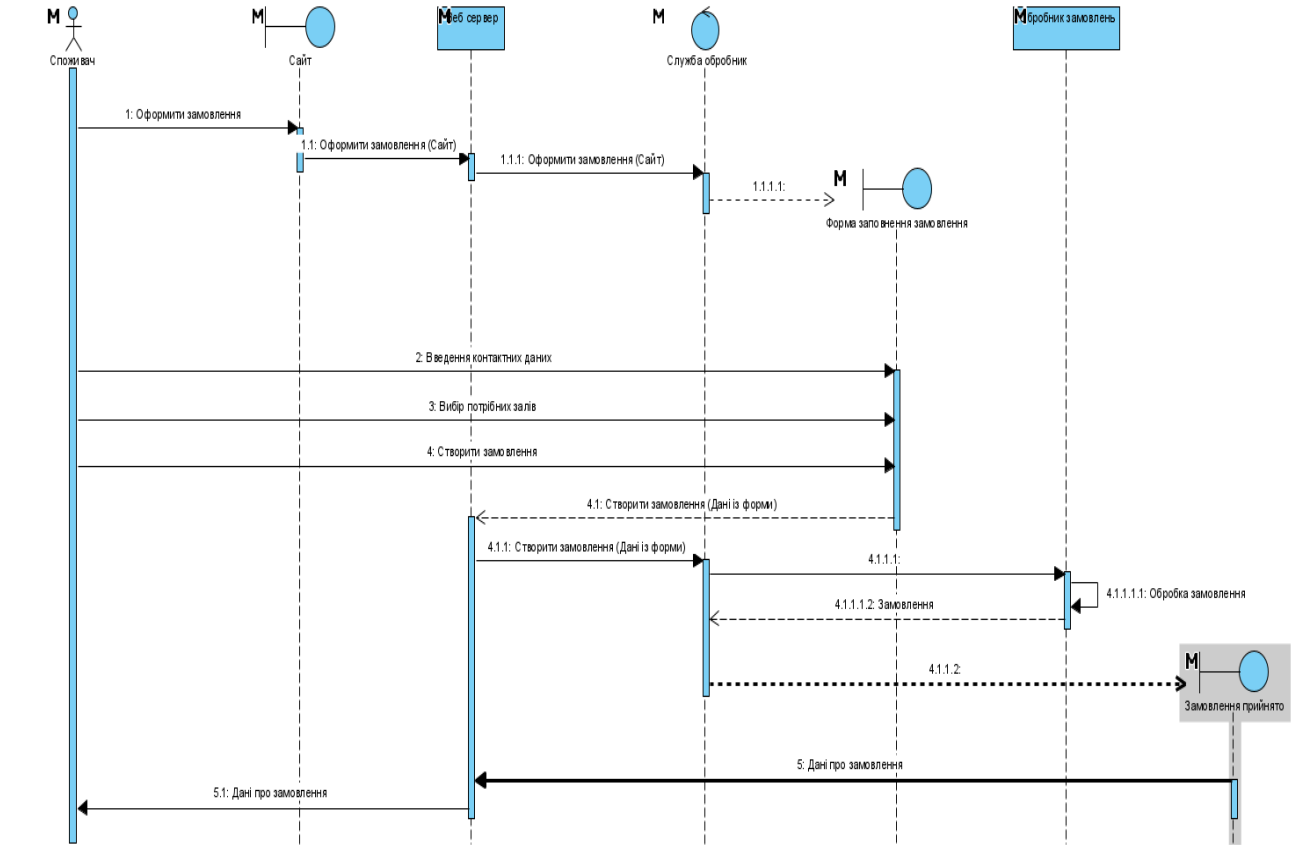


Рисунок 7 – Діаграма послідовності для оформлення замовлення

На цій діаграмі нашим актором є клієнт нашого сайту.

Об’єкти діаграми :

Споживач
Сайт
Веб сервер
Служба обробник
Форма оформлення замовлення
Контролер замовлень

Після цього за допомогою синхронізації з CommunicationDiagram ми отримаємо кооперативну діаграму :

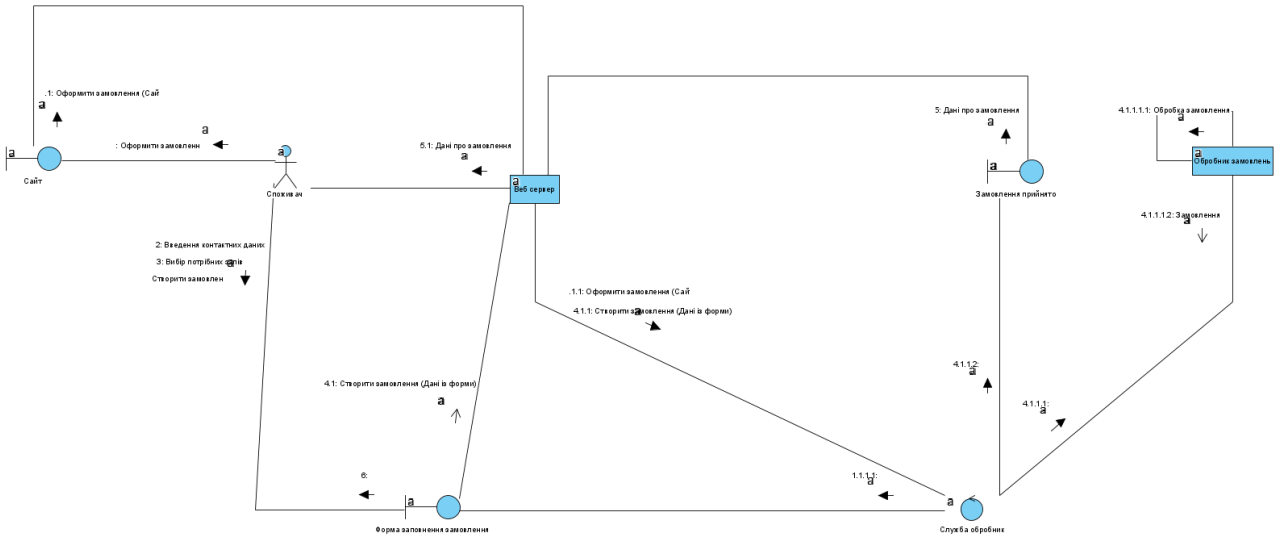


Рисунок 8 – Кооперативна діаграма для оформлення замовлення

1.1.3 ClassDiagram (Діаграма класів)

Діаграма класів – це статичне представлення структури моделі. Ця діаграма відображає статичні елементи, такі як:

Класи
Типи даних
Їх зміст
Відношення між ними

В діаграмі класів існують такі різні види зв’язків :

Асоціація(показують що об’єкти сутності пов’язані між собою)
Агрегація(відображає структурне відношення між рівноправними сутностями)
Композиція (агрегація за значенням)

Діаграма класів представлена на рисунку 9

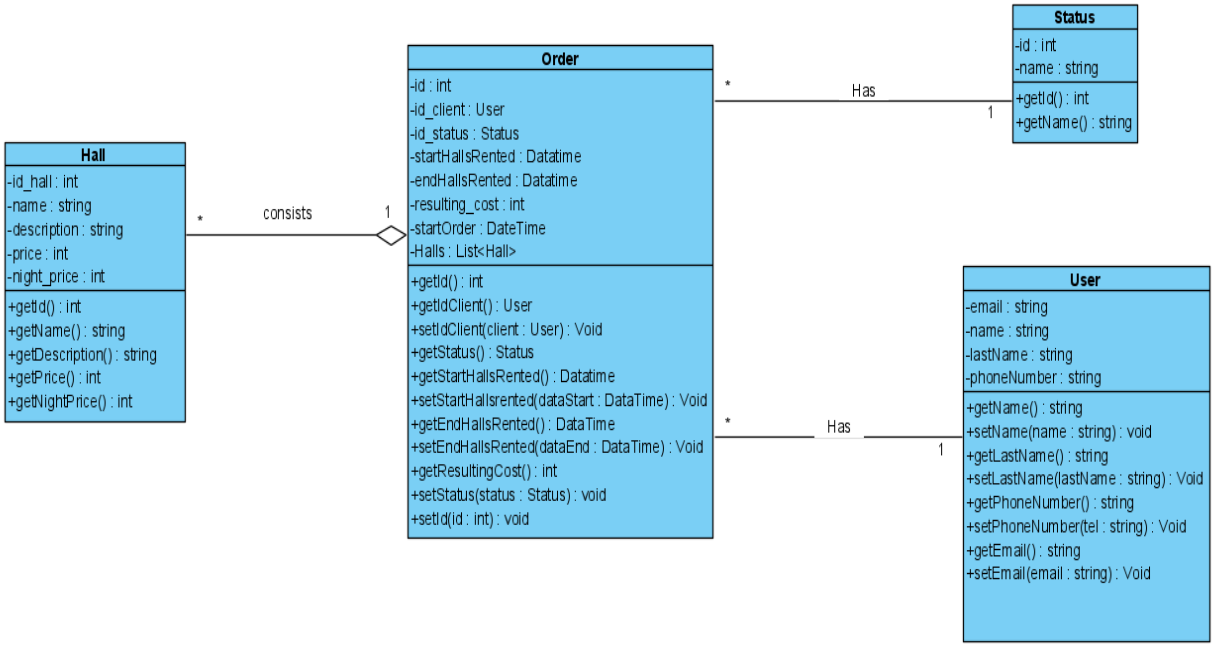


Рисунок 9 – Діаграма класів основних сутностей

Діаграма класів основних сутностей представлена у вигляді основних класів та зв’язків між ними

Діаграма класів для оформлення замовлення представлена на рисунку 10

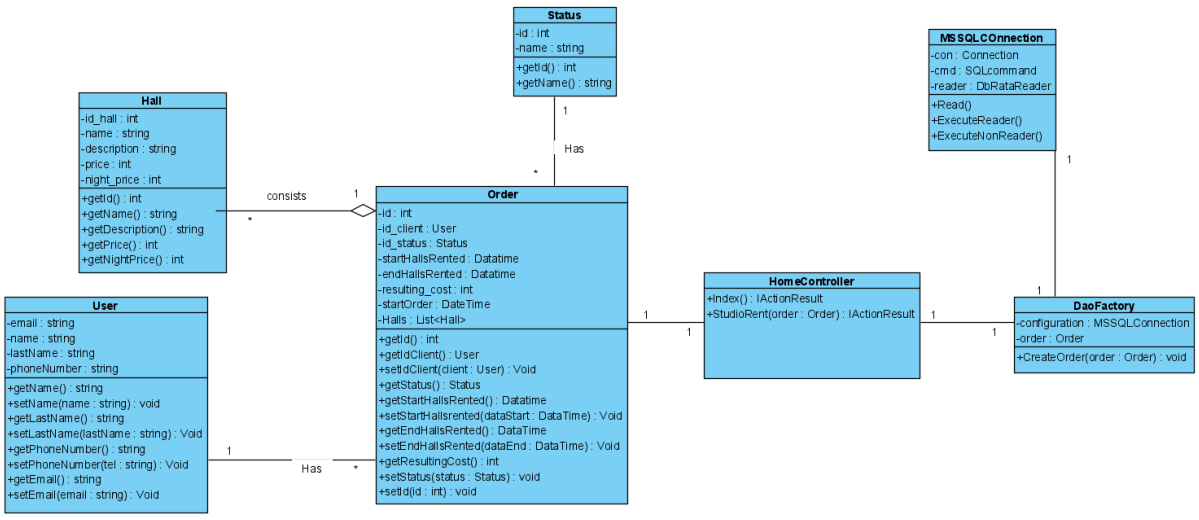
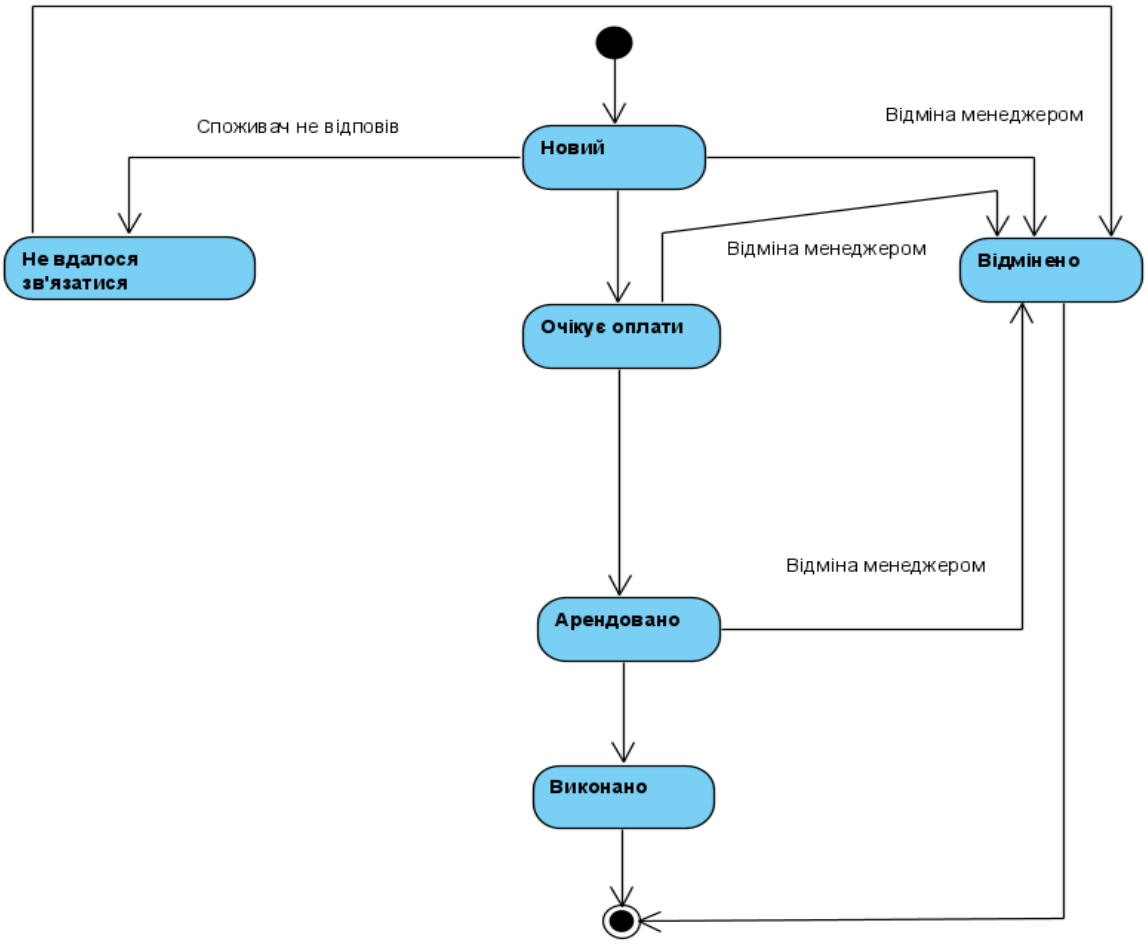


Рисунок 10 – Діаграма класів для оформлення замовлення

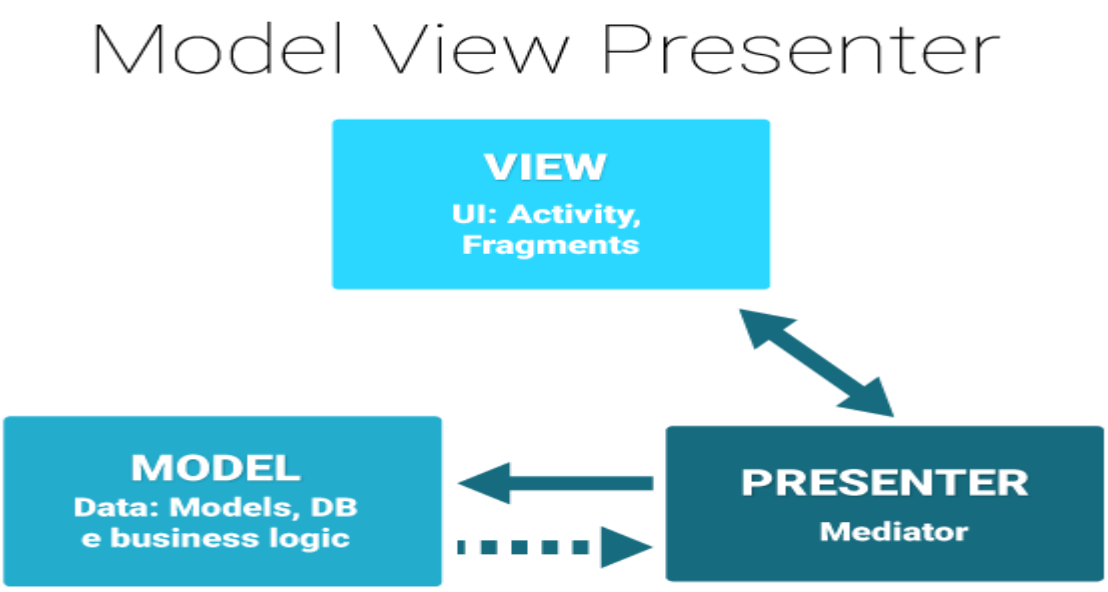
1.1.4 StateMachineDiagram (діаграма станів)

Діаграма станів є графом спеціального виду, що представляє певний автомат. Вершинами графа є можливі стани автомата, зображувані відповідними графічними символами, а дуги позначають його переходи зі стану в стан. Діаграму представлена на рисунку 11



Рисунку 11 Діаграма станів замовлення

У проекті використовуватись буде архітектура веб-сервісу з кросплатформовою підтримкою з пат терном MVP.



MVP – продукт, що володіє мінімальними, але достатніми функціями для задоволення перших споживачів. Основне завдання - отримання зворотного зв'язку для формування гіпотез подальшого розвитку продукту.

Архітектура веб - сервісу – сукупність найважливіших рішень щодо організації програмного середовища. Це базова організація системи, втілена в її компонентах, їх стосунки між собою і з оточенням, а також принципи, що визначають проектування і розвиток системи.

Слаботипізована мова програмування – відсутність строгої вимоги щодо призначення типу даних. Як наслідок: свобода дій, але запуск та робота програми не гарантує її повної коректності та надійності.

Кросплатформова підтримка – можливість роботи з веб – сервісом використовуючи будь-яку роздільну здатність пристрою (смартфон, планшет, ноутбук, комп’ютер) не заважаючи перегляду змісту веб – сторінок.

MPA – багатосторінковий веб застосунок.

Висновок: була розроблена архітектура проекту, розроблені необхідні абстракції для її реалізації на базі вимог до інформаційної системи.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний технічний університет
«Харківський політехнічний інститут»
Кафедра «Програмної інженерії та
інформаційних технологій управління»

Лабораторна робота №3

з курсу

«Економіка та організація виробництва систем»
за темою «Розробка проекту. Розробка бази даних.»

Перевірила: стар. вик.
Єршова С.І.
Виконав: студент КН-218б
Парахін З.Г.

Харків 2022

Мета роботи:

Отримати практичні навички розробки бази даних.

Хід роботи

1. Розробити базу даних.
2. Оформити звіт про виконання лабораторної роботи.

1.2 Логічне й фізичне моделювання даних

Логічна модель даних – це модель, яка відображає загальне бачення системи та не прив’язана до будь-якої СУБД. Призначена для візуалізації об’єктів даних та взаємозв’язків між ними.

Фізична модель даних – це модель, яка є реалізацією логічної моделі даних, що виражена у термінах мови опису конкретної СУБД.

У ході проектування було створено фізичну модель високонавантаженої бази даних на платформі MySQLy вигляді EER-діаграми згідно з нотацією IDEF1X для таблиць типу InnoDB та MyIsam (за допомогою програмного пакету Workbench).

Наведені типи таблиці платформи MySQL, а саме InnoDB та MyIsam, відрізняються тим, що:

- Таблиці типу InnoDBпідтримують посилальну цілісність та зв’язки за рахунок зовнішніх ключів, підтримують транзакції, мають швидше виконання одночасних запитів до різних даних однієї таблиці та змішаних запитів SELECT, UPDATE, DELETE, INSERT, а також є можливість відновлення даних таблиці після збою;
- Таблиці типу MyISAMмають більш швидке виконання однотипних запитів INSERT, SELECT, швидке виконання запитів типу COUNT(*), підтримку повнотекстового пошуку та індексування полів у запитах. Основний

недолік – підтримку посилальної цілісності варто забезпечити власноруч, створивши тригери для усіх видів операцій.

На рис. 2.1 зображена EER-модель бази даних з типом таблиць InnoDB.

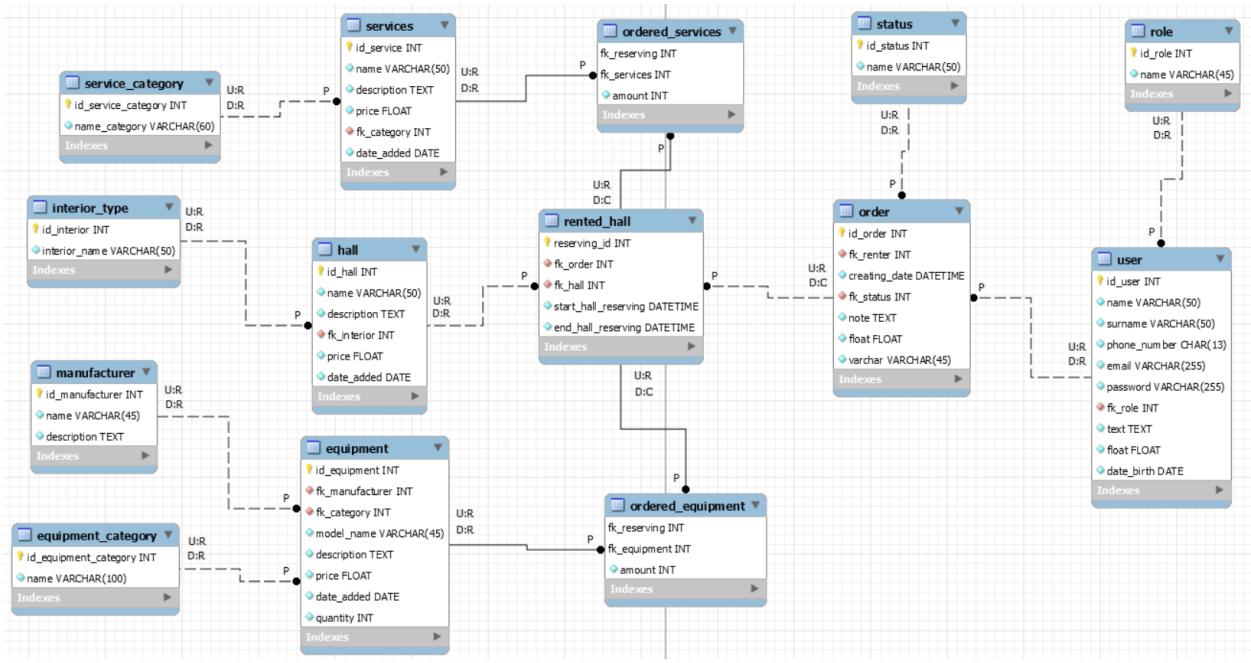


Рисунок 2.1 – Фізична модель даних (InnoDB)

Таблиця 3.1 – Сутності та їх призначення

Сутність	Призначення
Role	Зберігає інформацію про ролі користувачів
User	Зберігає дані користувача
Order	Зберігає дані про замовлення
Status	Зберігає інформацію про статус замовлення
Hall	Зберігає інформацію про зали студії
Equipment_category	Зберігає інформацію про категорію обладнання
Equipment	Зберігає дані про обладнання
Service	Зберігає інформацію про доступні послуги

На наведеній вище фізичній моделі даних відображено первинні ключі таблиць (PK)та зовнішні ключі (FK), типи даних полів, атрибути «NULL»та «NOTNULL», визначені зв’язки між таблицями та типи посилальної цілісності за зовнішнім ключем у вигляді скорочень (U:R, D:R, D:C).

Реалізована фізична модель бази даних відповідає вимогам третьої нормальної форми, кожна основна таблиця зв’язана хоча б з однією підставковою таблицею та кожна таблиця містить не менше п’яти атрибутів з наступними обов’язковими типами даних: «INT», «VARCHAR», «TEXT», «FLOAT», «DATE».

До основних таблиць належать: «User», «Order», «Hall», «Services», «Equipment».

Для реалізації повнотекстового пошуку таблицям та текстовим полям у них було встановлене кодування «utf8»та кодування для схеми порівняння «utf8_unicode_ci».

Фізична модель даних з використанням таблиць типу MyISAMза своєю структурою, таблицями, атрибутами таблиць повністю відповідаю фізичній моделі даних з використанням таблиць типу InnoDB, що зображена на рис. 2.1, за виключенням зв'язків між таблицями. Реалізація зв'язків виконується за допомогою тригерів, через те, що таблиці типу MyISAMне підтримують забезпечення посилальної цілісності за зовнішнім ключем.

На рис. 2.2 зображена EER-модель бази даних з типом таблиць MyISAM.

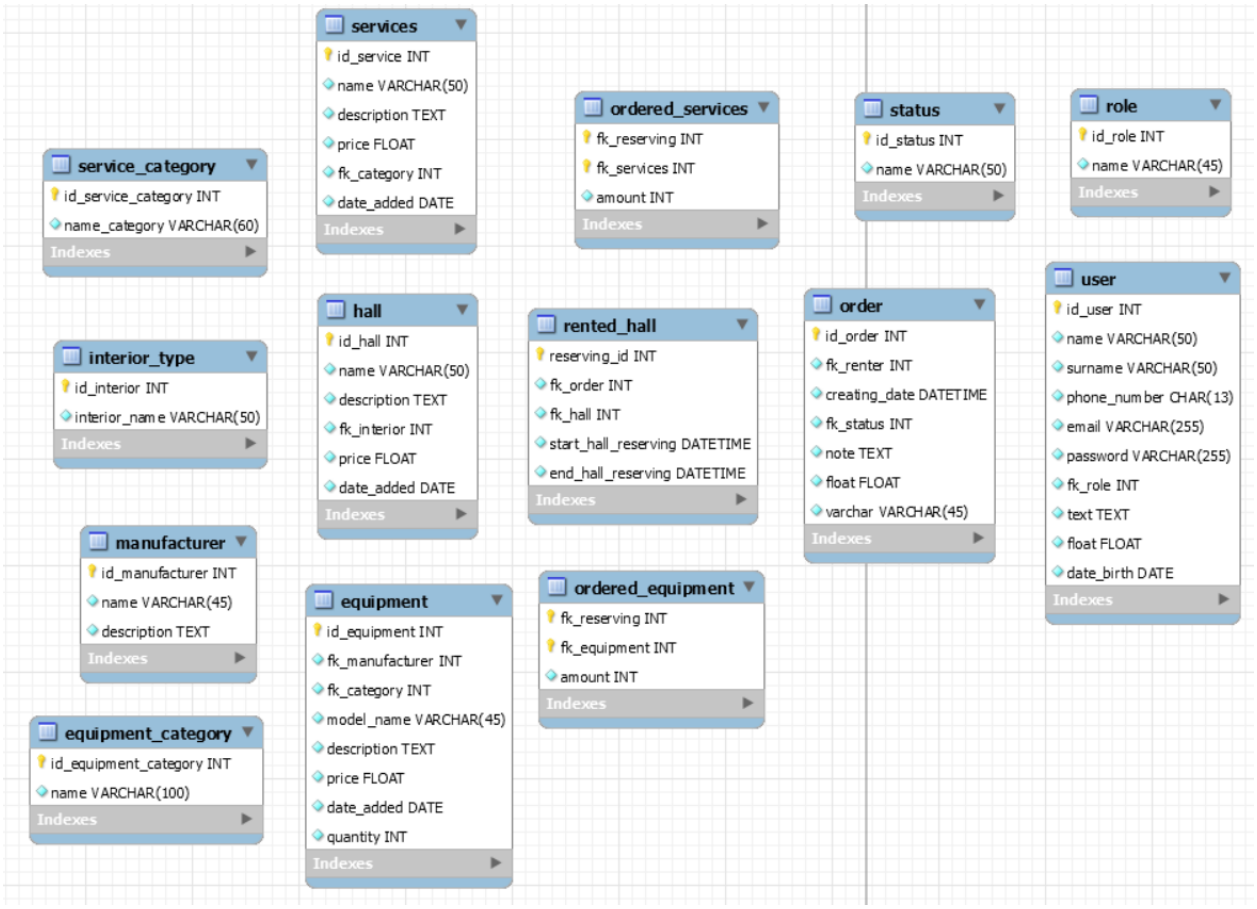


Рисунок 2.2 – Фізична модель даних (MyISAM)

На наведеній вище фізичній моделі даних відображено первинні ключі таблиць (PK)та зовнішні ключі (FK), типи даних полів, атрибути «NULL»та

«NOTNULL», для кожної таблиці визначені тригери, що реалізують зв’язки між таблицями та містять опис дій для кожного типу посилальної цілісності «ONINSERT», «ONDELETE», «ONUPDATE.

Реалізована фізична модель бази даних відповідає вимогам третьої нормальної форми, кожна таблиця містить не менше п’яти атрибутів з наступними обов’язковими типами даних: «INT», «VARCHAR», «TEXT», «FLOAT», «DATE».

До основних таблиць належать: «User», «Order», «Hall», «Services», «Equipment».

Для реалізації повнотекстового пошуку таблицям та текстовим полям у них було встановлене кодування «utf8»та кодування для схеми порівняння «utf8_unicode_ci».

Опис сутностей БД, зображених на рис. 2.1 та рис. 2.2:

1) Role – визначає роль користувача у системі. Складається з таких полів:id_role – унікальний ідентифікатор ролі користувача, name – назва ролі.

2) User – описує користувача інформаційної системи, що зареєструвався.Складається з таких полів:id_user – унікальний ідентифікатор користувача, name – ім’я користувача, surname – прізвище користувача, phone_number – контактний номер телефону, для зв’язку менеджера після створення нового замовлення, email – електронна пошта користувача та одночасно унікальне ім’я для входу у систему, password – пароль користувача, для підтвердження входу у систему, fk_role – містить ідентифікатор ролі, text – містить текстові дані, float – містить числа з плаваючою комою, date_birth – дата народження користувача.

3) Status – описує можливі статуси замовлення.Складається з таких полів: id_status – містить унікальний ідентифікатор статусу у системі, name – назва статусу.

4) Order – описує замовлення, що було зроблено певним користувачем. Складається з таких полів:id_order – унікальний ідентифікатор замовлення у системі, fk_renter – ідентифікатор клієнта, що зробив замовлення, creating_date – дата та час, коли було занесено резервацію до системи, fk_status

– ідентифікатор статусу замовлення, note – додаткові текстові помітки до замовлення, float – числові дані з плаваючою комою, varchar – текстові дані.

5) Equipment_category – описує категорії обладнання, що може бути обране користувачем при замовленні залу. Складається з таких полів: id_equipment_category – унікальний ідентифікатор категорії обладнання, name – назва категорії.

6) Manufacturer – описує виробників обладнання, що наявне у підприємства. Складається з таких полів: id_manufacturer – унікальний ідентифікатор виробника, name – назва виробника, description – опис виробника.

7) Equipment – описує обладнання, що користувач може обрати при оренді залу. Складається з таких полів: id_equipment – унікальний ідентифікатор обладнання, fk_manufacturer – ідентифікатор компанії виробника обладнання, fk_category – ідентифікатор категорії, до якої належить обладнання, model_name – назва обладнання, description – опис обладнання, price – ціна обладнання, date_added – дата додання цього обладнання до каталогу, quantity – кількість відповідного обладнання на складі.

8) Interior_type – описує типи інтер’єрів, що використовуються у залах фотостудії. Складається з таких полів: id_interior – унікальний ідентифікатор інтер’єру, interior_name – назва інтер’єру.

9) Hall – описує зали, що надаються для оренди користувачами. Складається з таких полів: id_hall – унікальний ідентифікатор залу, name – назва залу, description – опис залу, fk_interior – ідентифікатор інтер’єру, за яким оформлений зал, price – ціна залу за годину, date_added – дата додання залу до каталогу.

10) Service_category – описує категорії додаткових послуг, що надаються при резервації залів. Складається з таких полів: id_service_category – унікальний ідентифікатор категорії залу, name_category – назва категорії.

11) Services – описує додаткові послуги, що надаються з певним залом, який замовив користувач. Складається з таких полів: id_service – унікальний ідентифікатор послуги, name – назва послуги, description – опис послуги, price

– ціна послуги за годину, fk_category – ідентифікатор категорії, до якої належить послуга, date_added – дата додання послуги до каталогу.

12) Rented_hall – описує кошик замовлення користувача та визначає замовлені зали, що були замовлені. Складається з таких полів:reserving_id – унікальний ідентифікатор запису, fk_order – замовлення, до якого належить замовлений зал, fk_hall – ідентифікатор залу, що був замовлений, start_hall_reserving – дата та час початку оренди залу, end_hall_reserving – дата та час кінцю оренди залу.

13) Ordered_services – описує замовлені додаткові послуги до замовленого залу. Складається з таких полів:fk_reserving – ідентифікатор замовленого залу із кошика, fk_services – ідентифікатор послуги, amount – кількість послуг одного виду, що були замовлені (наприклад кількість додаткових учасників).

14) Ordered_equipment – описує замовлене обладнання до замовленого залу. Складається з таких полів:fk_reserving – ідентифікатор замовленого залу із кошика, fk_equipment – ідентифікатор обладнання, amount – кількість обладнання певного виду.

Висновки: набуто практичних навичок щодо створення бизи даних для інформаційної системи і засвоєно необхідні знання з теми роботи.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний технічний університет
«Харківський політехнічний інститут»
Кафедра «Програмної інженерії та
інформаційних технологій управління»

Лабораторна робота №4

з курсу

«Економіка та організація виробництва систем»
за темою «Розробка проекту. Розробка інтерфейсу.»

Перевірила: стар. вик.
Єршова С.І.
Виконав: студент КН-218б
Парахін З.Г.

Харків 2022

Мета роботи:

Отримати практичні навички проектування інтерфейсу.

Хід роботи

1. Розробити проект інтерфейсу.
2. Оформити звіт про виконання лабораторної роботи.

Відповідно до вимог представлених у роботі №1. Клієнтська частина являє собою багатосторінковий сайт. Сайт повен бути зрозумілим для користувача. Вміст сайту має відповідати бізнес-вимогам, контент повинен бути унікальним. Навігація сайту повинна бути простою і зрозумілою, так як це позитивно впливає на конверсію.

На рисунку 1 представлено інтерфейс веб-сайту :

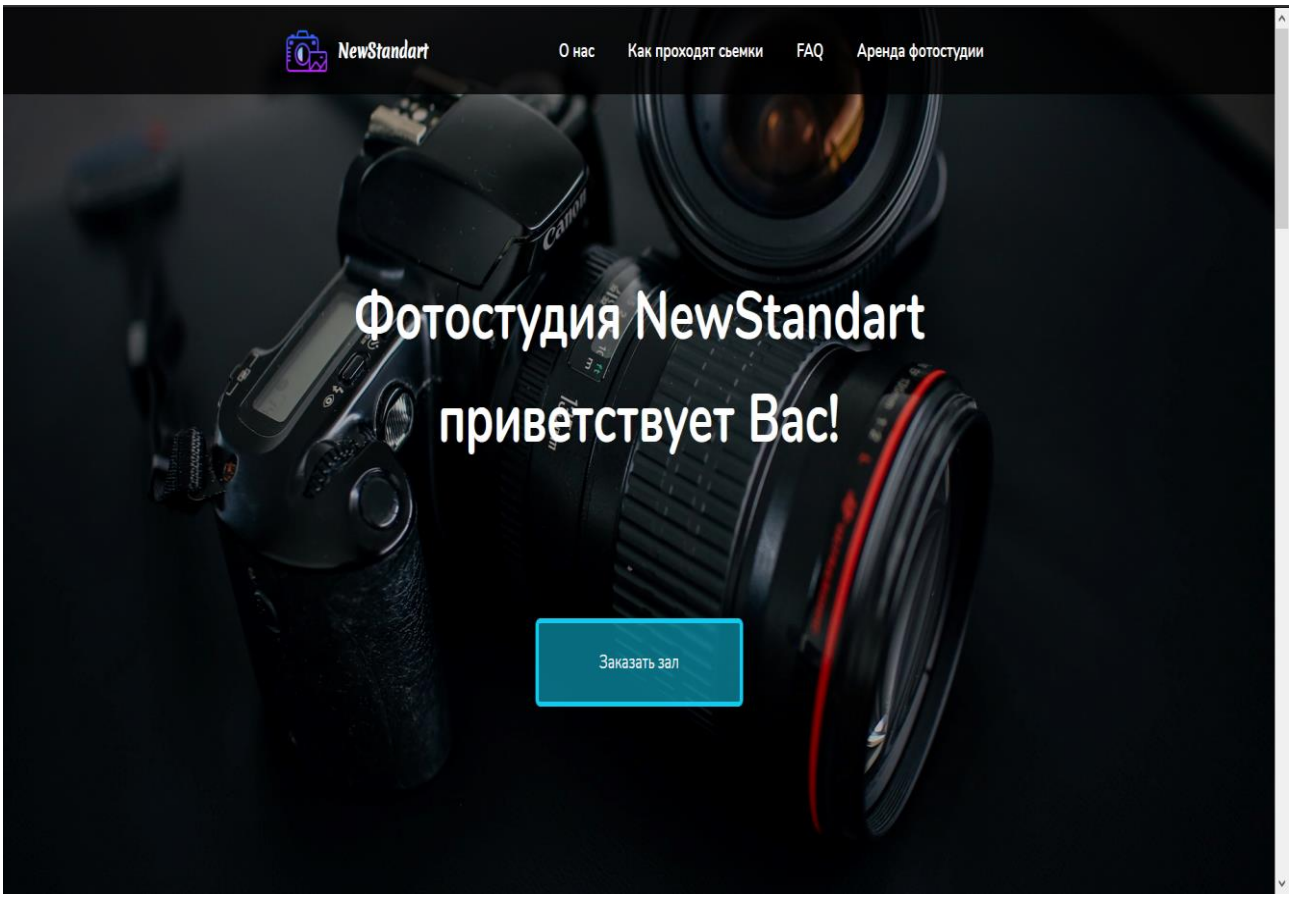


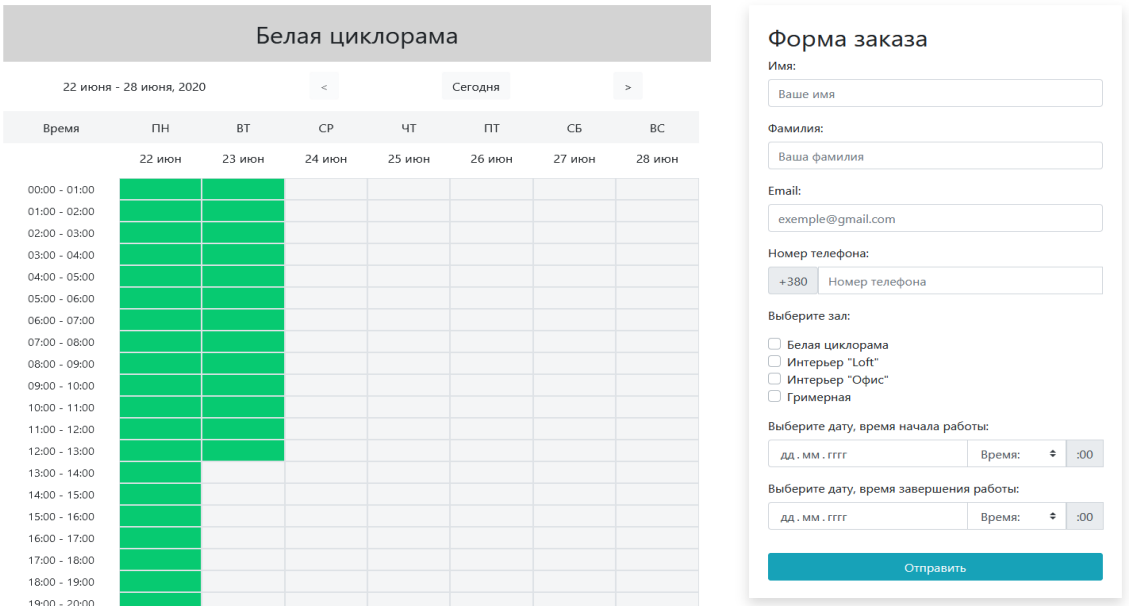
Рисунок 1 – інтерфейс головної сторінки веб-сайту

Перед обличчям користувача зразу спливає кнопка замовлення залів. Ті хто прийшовши вперше перейдуть нижче та прочитають про компанію, а

досвідчені юзери цього сайту перейдуть зразу ж на замовлення . У верхній частині сторінки відображено навігаційну панель, за допомогою якої користувач може :

Продивитися головну сторінку, рухаючись по міткам
Зробити замовлення

Також на сайті повинні бути реалізована форма з відображенням завантаження залів які зображені на рисунку 2 :



Рисунку 2 – Форма з відображенням календарю завантаженості залів

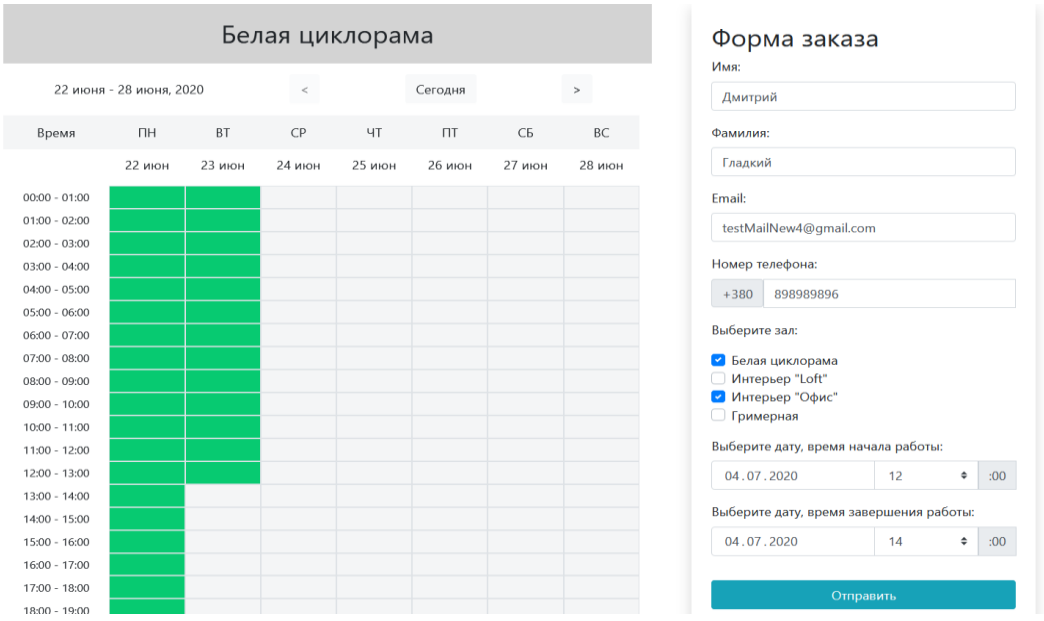


Рисунок 3 – Сторінка створеного замовлення

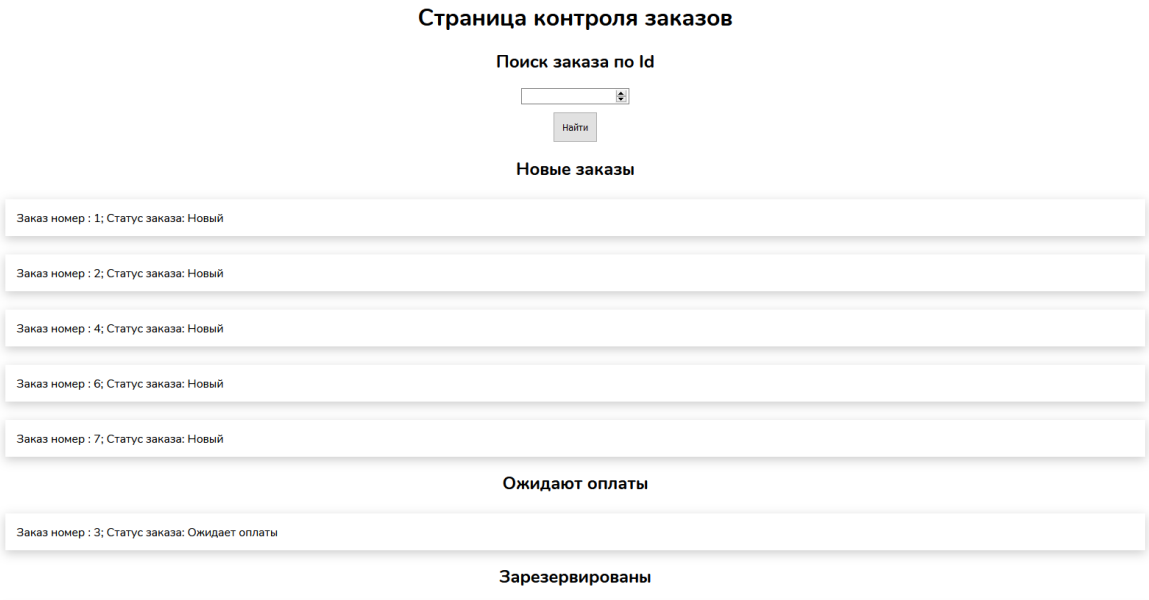


Рисунок 4 – Сторінка менеджера з відображенням усіх замовлень

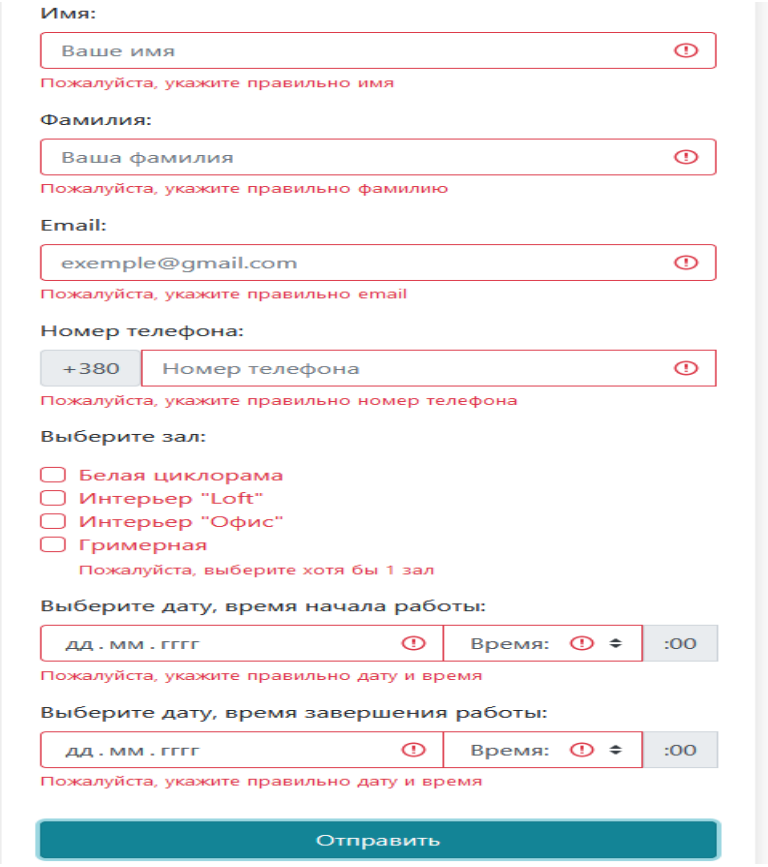


Рисунок 5 – Повідомлення про незаповнене поле

Інтерфейс для користувача складається з багатьох сторінок, які пов’язані між собою посиланнями. Коли споживач натискає на якусь кнопку – контролер

обробляє його запит та відправляє на відповідну сторінку.

Використання ASP.NET дуже зручне для створення багатосторінкового сайту. При створені проекту відразу підключається пакети CSS, які були підготовлені заздалегідь, стиль для нашого сайту. Також автоматично створюється навігаційна панель з якою дуже зручно працювати користувачу сайта.

Для розробки графічної частини сайту використовувалося– HTML, CSS, бібліотекаBootstrap, а також не значна частина функціоналу JavaScriptта JQuery.

Бібліотека Bootstrapдозволяє розробити високоякісний графічний інтерфейс лише при взаємодії об'єктів HTMLта класів, запрограмованих в ній.

JavaScriptта JQuery– допомагає реалізувати діалог користувача с веб-сторінкою та додати обмеження на введенні дані – валідацію полів.

Висновок: на основі аналізу предметної галузі, а також визначених основних бізнес-процесів були сформовані чіткі функціональні вимоги до клієнтської частини оренди залів фотостудії за якими розроблено потрібний інтерфейс.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний технічний університет
«Харківський політехнічний інститут»
Кафедра «Програмної інженерії та
інформаційних технологій управління»

Лабораторна робота №5

з курсу

«Економіка та організація виробництва систем»
за темою «Розробка проекту. Розробка алгоритмів.»

Перевірила: стар. вик.
Єршова С.І.
Виконав: студент КН-218б
Парахін З.Г.

Харків 2022

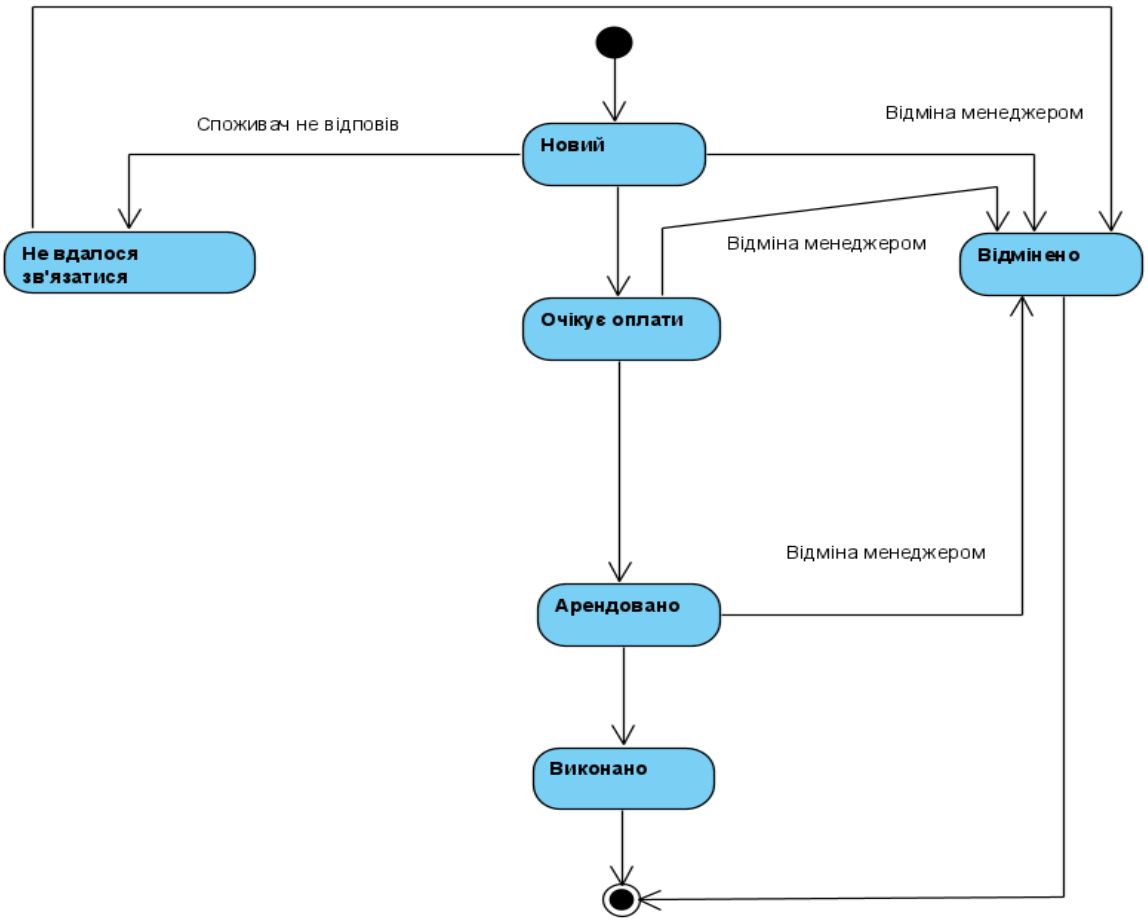
Мета роботи:

Отримати практичні навички розробки алгоритмів вирішення задачі .

Хід роботи

1. Розробити алгоритми вирішення задачі.
2. Оформити звіт про виконання лабораторної роботи.

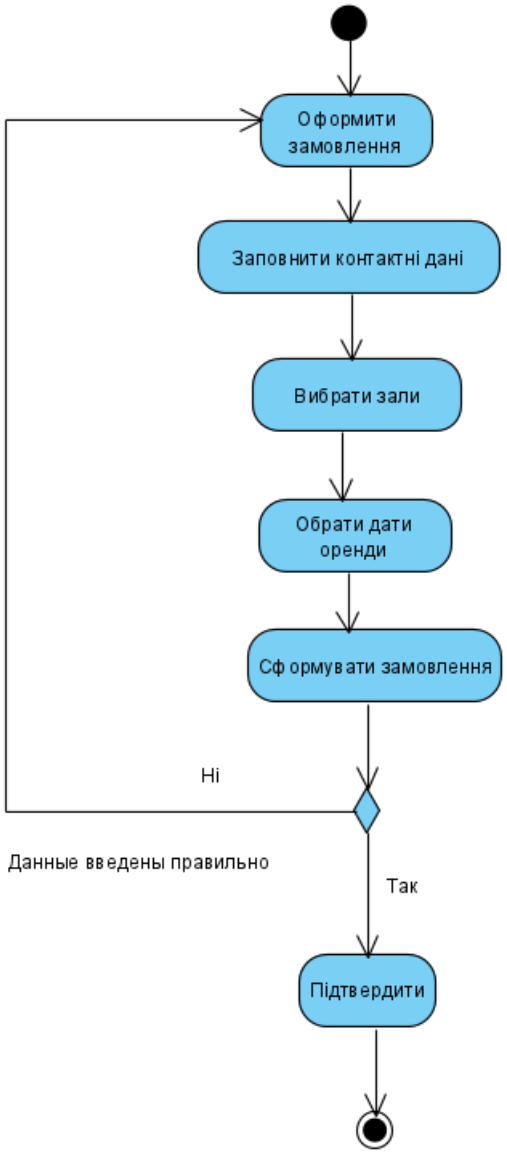
Для представлення алгоритмів вирішення задачі, а саме для бізнес-процесів використовуємо діаграми діяльності та програмний код, так як саме програмний код є найбільш необхідний для розуміння роботи алгоритмів у спроектованій системі. Задля також зобразимо діаграму станів для розуміння які саме задачі повинні вирішуватися. Діаграма станів є графом спеціального виду, що представляє певний автомат. Вершинами графа є можливі стани автомата, зображувані відповідними графічними символами, а дуги позначають його переходи зі стану в стан. Діаграму представлена на рисунку 1



Рисунку 1 Діаграма станів замовлення

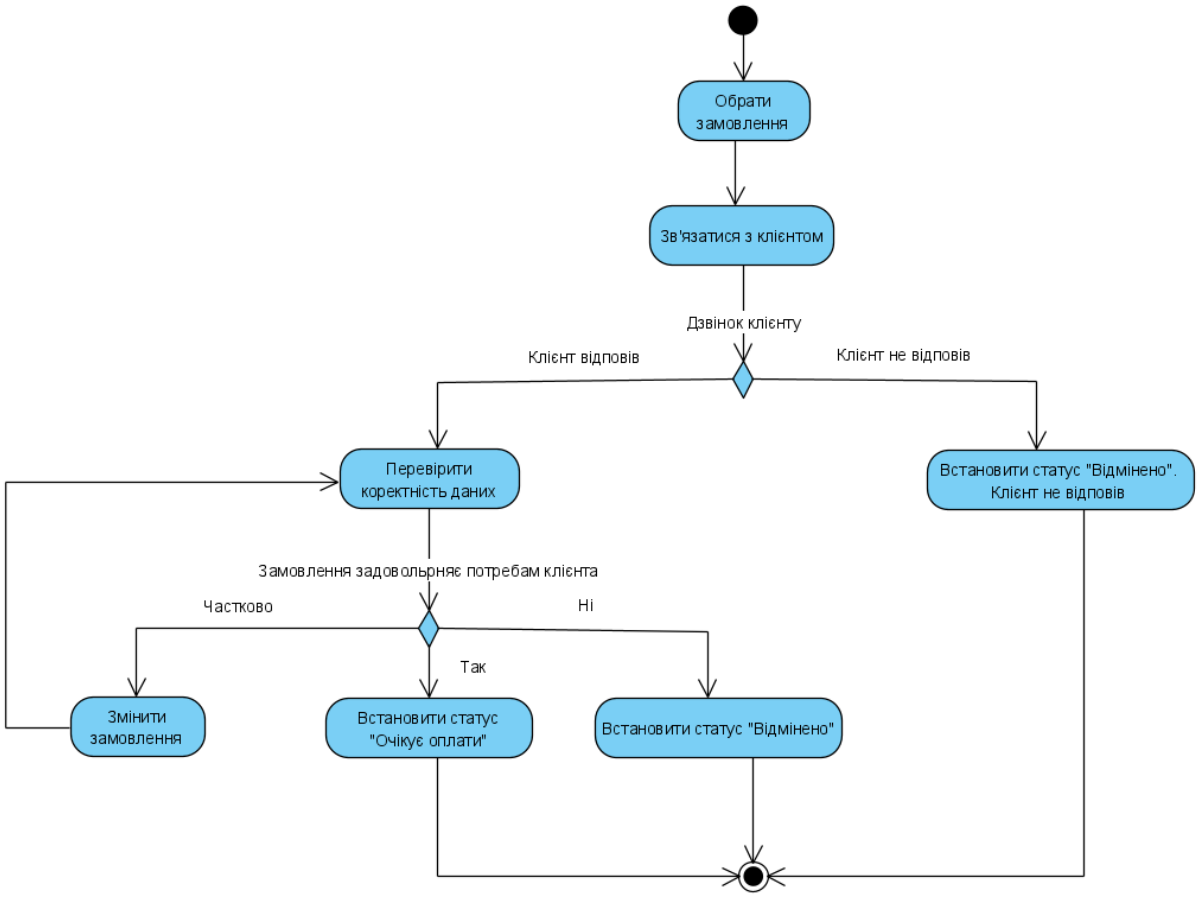
1.2.1 ActivityDiagram(діаграма діяльності)

Діаграма діяльності потрібна для моделювання процесу виконання операцій, вершинами діаграми є стани, а дуги – переходи від одного стану до іншого стану. Діаграму для оформлення замовлення представлено на рисунку 2.14.



Рисунку 2.14 – Діаграма діяльності для оформлення замовлення

На діаграмі представлено послідовність перетікання подій, процес оформлення замовлення включає тільки послідовні блоки.



Рисунку 2.15 – Діаграма діяльності для зв’язку з клієнтом

Діаграма зображує послідовну схему для зміни статусу замовлення залежно від відповіді клієнта та коректності даних.

Висновок: було розроблено алгоритми та необхідний програмний код для проекту. Під час проектування розроблені всі необхідні діаграми для представлення системи у зручному та коректному вигляді

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний технічний університет
«Харківський політехнічний інститут»
Кафедра «Програмної інженерії та
інформаційних технологій управління»

Лабораторна робота №6

з курсу

«Економіка та організація виробництва систем»
за темою «Розробка програмного забезпечення оцінки економічних показників ПЗ .»

Перевірила: стар. вик.
Єршова С.І.
Виконав: студент КН-218б
Парахін З.Г.

Харків 2022

Мета роботи:

Отримати практичні навички реалізації оцінки економічних показників ПЗ, яке розроблюється.

Хід роботи

1. Визначити засоби та технології розробки програмного коду. Обґрунтувати свій вибір.
2. Розробити програмний код. Задокументувати результати.
3. Оформити звіт про виконання лабораторної роботи.

1.Визначити засоби та технології розробки програмного коду. Обґрунтувати свій вибір.

1.1 Обґрунтування вибору мови програмування

В якості мови програмування для серверної частини було використано мову C#. Фреймворком для створення веб-застосунків був ASP.NET. У якості архітектурного шаблону використовувався MVC (модель-вигляд-контролер)

C# - об’єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Вона має високий рівень інтеграції з іншими продуктами Microsoft. C# є мовою з C-подібним синтаксисом. Так як мова C# э об’єктно-орієнтована, то вона підтримує поліморфізм, успадкування, перевантаження операторів та статичну типізацію. Ця мова має велику кількість бібліотек.

Технологія ASP.NET – це дуже простий інструмент для створення веб-сайтів. ASP.NET має перевагу у швидкості в порівнянні з іншими технологіями, заснованими на скриптах. Він має розширюваний набір елементів управління і бібліотек класів, що дозволяє швидко розробляти застосунки.

MVC – цей шаблон поділяє нашу систему на три пов’язані між собою частини: модель даних, вигляд (інтерфейс користувача) та контролер. За допомогою такої архітектури дуже зручно і швидко написати програму по вже

готовому проекту.

1.2 Обґрунтування вибору СУБД

Для зберігання даних, які надходять від клієнта обрано СУБД MS SQL SERVER.

Microsoft SQL Server – система управління базами даних, розроблена корпорацією Microsoft. Основна мова запитів – Transact-SQL. Transact-SQL є реалізацією стандарту ANSI/ISO. Ця СУБД використовується для роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства.

Э вбудованою в пакет програмного забезпечення VisualStudio, а тому має повну інтеграцію з цим середовищем розробки, що спрощує процес розробки бази даних.

Використовуючи MSSQLExpressManagementStudio 2019, можна створити візуальне відображення діаграм.

Ця діаграма представлена у вигляді діаграми типу «сутність – зв’язок» на рисунку 1

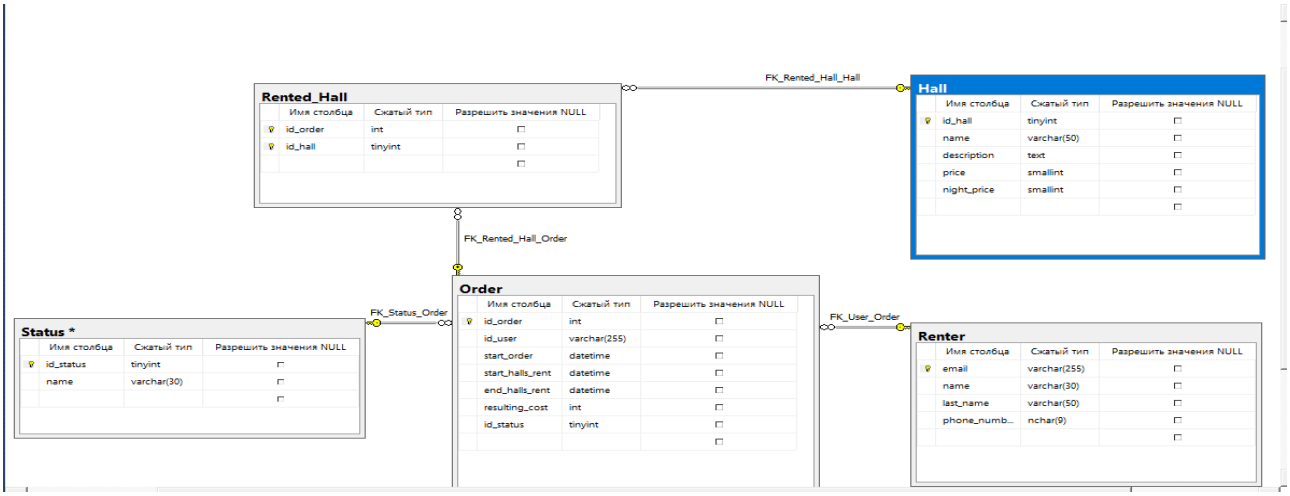


Рисунок 1 – Діаграма створеної моделі даних у середовищі розробки

1.3 Розробка інтерфейсу клієнтської частини системи

Інтерфейс для користувача складається з багатьох сторінок, які пов’язані між собою посиланнями. Коли споживач натискає на якусь кнопку – контролер

обробляє його запит та відправляє на відповідну сторінку.

Використання ASP.NET дуже зручне для створення багатосторінкового сайту. При створені проекту відразу підключається пакети CSS, які були підготовлені заздалегідь, стиль для нашого сайту. Також автоматично створюється навігаційна панель з якою дуже зручно працювати користувачу сайта.

Для розробки графічної частини сайту використовувалося– HTML, CSS, бібліотекаBootstrap, а також не значна частина функціоналу JavaScriptта JQuery.

Бібліотека Bootstrapдозволяє розробити високоякісний графічний інтерфейс лише при взаємодії об'єктів HTMLта класів, запрограмованих в ній.

JavaScriptта JQuery– допомагає реалізувати діалог користувача с веб-сторінкою та додати обмеження на введенні дані – валідацію полів.

ОЦІНКА ТРУДОМІСТКОСТІ, ТРИВАЛОСТІ ТА ПОТРЕБИ У СПІВРОБІТНИКАХ ДЛЯ ПІЛОТНОГО ПРОЕКТУ

У цьому розділі наводиться розрахунки оцінкитрудомісткості, тривалості та потреби у співробітниках для виконанняпілотного проекту з використанням методу функціональних точок.

В основу класичного методу функціональних точок лягло уявленняреалізованої системи, як множини елементів, які належать до двох основних підгруп: дані та транзакції. У цій галузі поняття "транзакція"являє собою елементарний неподільний замкнутий процес, який представляєзначення для користувача і переводить продукт з одного консистентногостану в інше.

У класичному методі функціональних точок характеристики саме цих елементівдозволяють визначати кількість нескоректованих функціональнихточок (UFP), виходячи з якої, розраховується кількість нескоректованихфункціональних балів для створюваної системи за формулою:

$$UFP = \sum_{i=1}^{n_{ILF}} UFP_i + \sum_{j=1}^{n_{EIF}} UFP_j + \sum_{k=1}^{n_{EI}} UFP_k + \sum_{l=1}^{n_{EO}} UFP_l + \sum_{m=1}^{n_{EQ}} UFP_m$$

(5.1)

$\sum_{i=1}^{n_{ILF}} UFP_i$ - сума нескоректована функціональних точок, що характеризує складність структур даних, які використовуються створюваним продуктом в ході виконання своїх функцій;

Nilf – кількість сутностей в БД. В нашому випадку це 10

Виходячи з DETта RETмаємо низьке(low)значення функцій складності оброблюваних даних.

Отже $\sum_{i=1}^{n_{ILF}} UFP_i$дорівнює 70.

$\sum_{j=1}^{n_{EIF}} UFP_j$ дорівнює 0 так, як інформаційна система не включає в себе зовнішніх баз даних.

$\sum_{k=1}^{n_{EI}} UFP_k$- сума нескоректованих функціональних точок, що характеризує складність зовнішніх інтерфейсів для введення даних (зовнішніх вводів (EI)), які використовуються створюваним продуктом для введення даних від користувача або пристрою збору інформації в ході виконання своїх функцій;

n_{EI} - кількість зовнішніх вводів (EI), які використовуються створюваним продуктом для введення даних від користувача або пристрою збору інформації в ході виконання своїх функцій;

Виходячи з DETта RETмаємо низьке(low) значення функцій складності оброблюваних даних.

Отже $\sum_{k=1}^{n_{EI}} UFP_k$ дорівнює 12.

$\sum_{l=1}^{n_{EO}} UFP_l$ - сума нескоректованих функціональних точок, що характеризує складність зовнішніх інтерфейсів для виводу даних (зовнішніх виводів (EO)), які використовуються створюваним продуктом для виведення результатів виконання своїх функцій;

n_{EO} - кількість зовнішніх виводів (EO), які використовуються створюваним продуктом для виведення результатів виконання своїх функцій;

Виходячи з DETта RETмаємо низьке(low) значення функцій складності оброблюваних даних.

Отже$\sum_{l=1}^{n_{EQ}} UFP_l$дорівнює 23.

$\sum_{m=1}^{n_{EQ}} UFP_m$ - сума нескоректованих функціональних точок, що характеризує складність зовнішніх запитів (EQ), які використовуються створюваним продуктом для надання інформації користувачеві або іншій системі у відповідь на задані умови пошуку;

n_{EQ} - кількість зовнішніх запитів (EQ), які використовуються створюваним продуктом для надання інформації користувачеві або іншій системі у відповідь на задані умови пошуку.

Виходячи з DETта RETмаємо низьке(low) значення функцій складності оброблюваних даних.

Отже$\sum_{m=1}^{n_{EQ}} UFP_m$дорівнює 40.

Виходячи з формули UFP = 70+0+12+23+40= 145.

Для врахування впливу загальносистемних характеристик і особливостей проектування в методі функціональних точок використовується фактор вирівнювання VAF. Значення VAF визначається за формулою:

$$VAF = (TDI * 0{,}01) + 0{,}65,$$

(5.2)

Де:

$$TDI = \sum_{a=1}^{14} DI_a,$$

(5.3)

DI_a- системні характеристики, значення яких знаходяться в діапазоні від 0 до 5 і визначаються по табл. 5.1:

Таблиця 5.1 - Перелік системних характеристик

№ п / п	*Найменування характеристики*	*Значення характеристики*
1	*Обмін даними*	*0*

2	*Розподілена обробка даних*	*4*
3	*продуктивність*	*5*
4	*Обмеження по апаратних ресурсів*	*3*
5	*Транзакційна навантаження*	*2*
6	*Інтенсивність взаємодії з користувачем*	*5*
7	*Ергономіка*	*4*
8	*Інтенсивність зміни даних (ILF) користувачами*	*4*
9	*Складність обробки*	*3*
10	*Повторне використання*	*3*
11	*Зручність інсталяції*	*3*
12	*Зручність адміністрування*	*2*
13	*Можливість портування*	*4*
14	*Гнучкість*	*5*

Виходячи з таблиці сума системних характеристик(TDI) дорівнює 47.

Фактор вирівнювання VAFдорівнює 1,12.

Тоді кількість скоригованих функціональних точок для розроблюваної ІС або її окремого завдання визначається за формулою:

$DFP = UFP * VAF$=162,4.

Для розрахунку трудомісткості виконання проекту, тривалості розробки і потреби в персоналі треба перевести скориговані функціональні точки в рядки коду. Обрана мова для розробки інформаційної системи C#, значення для обраної мови за таблицею:

	avg	median	low	high
C # *	54	59	29	70

Рисунок 5.1 – Кількість рядків коду, які необхідні для реалізації однієї точки

Розраховуємо кількість рядків коду на одну функціональну точку за методикою PERT:

$$k = \frac{Low+4*Average+High}{6}=52.5$$

Після цього розраховуємо кількість рядків коду LOC, яке треба написати для реалізації оцінюваної функціональної завдання по формулі:

$$LOC = DFP * k = 8526$$

Знаючи величину LOC, можна отримати оцінки трудомісткості, тривалості та потреби в персоналі з застосуванням моделі COCOMO. При цьому будемо виходити з того, що для самостійного виконання роботи використовується органічний режим виконання IT-проекту. Для даного режиму трудомісткість розробки функціональної завдання в людино-місяцях можна оцінити за формулою:

$$E = 2{,}4 * (LOC/1000)^{1{,}05}=22{,}7769$$

Тривалість розробки функціональної завдання в місцях можна оцінити за формулою:

$$TDEV = 2{,}5 * (E)^{0{,}38}=8{,}19951$$

Потреба персоналу можна оцінити за формулою:

$$SS = E/TDEV = 2{,}77783$$

Отже порахувавши всі формули проект має 8526 строк коду. Потребує 22,7769 людино-місяців, це означає, що потрібно 8,19951 місяців та 2,77783 людини.

Лістинг проекту:

Лістинг коду проекту 1:

```
USE [webserver]
GO

CREATETABLE [dbo].[Hall](
	[id_hall] [tinyint] IDENTITY(1,1)NOTNULL,
	[name] [varchar](50)NOTNULL,
	[description] [text] NOTNULL,
	[price] [smallint] NOTNULL,
	[night_price] [smallint] NOTNULL,
CONSTRAINT [PK_Hall] PRIMARYKEYCLUSTERED
(
	[id_hall] ASC
)WITH(PAD_INDEX=OFF,STATISTICS_NORECOMPUTE=OFF,IGNORE_DUP_KEY=OFF,ALLOW_ROW_LOCKS=ON,ALLOW_
```

```
PAGE_LOCKS=ON)ON [PRIMARY]
)ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]

GO

CREATETABLE [dbo].[Renter](
       [email] [varchar](255)NOTNULL,
       [name] [varchar](30)NOTNULL,
       [last_name] [varchar](50)NOTNULL,
       [phone_number] [nchar](9)NOTNULL,
CONSTRAINT [PK_Renter] PRIMARYKEYCLUSTERED
(
       [email] ASC
)WITH(PAD_INDEX=OFF,STATISTICS_NORECOMPUTE=OFF,IGNORE_DUP_KEY=OFF,ALLOW_ROW_LOCKS=ON,ALLOW_
PAGE_LOCKS=ON)ON [PRIMARY]
)ON [PRIMARY]

GO

CREATETABLE [dbo].[Status](
       [id_status] [tinyint] IDENTITY(1,1)NOTNULL,
       [name] [varchar](30)NOTNULL,
CONSTRAINT [PK_Status] PRIMARYKEYCLUSTERED
(
       [id_status] ASC
)WITH(PAD_INDEX=OFF,STATISTICS_NORECOMPUTE=OFF,IGNORE_DUP_KEY=OFF,ALLOW_ROW_LOCKS=ON,ALLOW_
PAGE_LOCKS=ON)ON [PRIMARY]
)ON [PRIMARY]

GO
CREATETABLE [dbo].[Order](
       [id_order] [int] IDENTITY(1,1)NOTNULL,
       [id_user] [varchar](255)NOTNULL,
       [start_order] [datetime] NOTNULL,
       [start_halls_rent] [datetime] NOTNULL,
       [end_halls_rent] [datetime] NOTNULL,
       [resulting_cost] [int] NOTNULL,
       [id_status] [tinyint] NOTNULL,
CONSTRAINT [PK_Order] PRIMARYKEYCLUSTERED
(
       [id_order] ASC
)WITH(PAD_INDEX=OFF,STATISTICS_NORECOMPUTE=OFF,IGNORE_DUP_KEY=OFF,ALLOW_ROW_LOCKS=ON,ALLOW_
PAGE_LOCKS=ON)ON [PRIMARY]
)ON [PRIMARY]

GO
CREATETABLE [dbo].[Rented_Hall](
       [id_order] [int] NOTNULL,
       [id_hall] [tinyint] NOTNULL,
CONSTRAINT [PK_Rented_Hall] PRIMARYKEYCLUSTERED
(
       [id_order] ASC,
       [id_hall] ASC
)WITH(PAD_INDEX=OFF,STATISTICS_NORECOMPUTE=OFF,IGNORE_DUP_KEY=OFF,ALLOW_ROW_LOCKS=ON,ALLOW_
PAGE_LOCKS=ON)ON [PRIMARY]
)ON [PRIMARY]

GO


using System;
usingSystem.Collections.Generic;
usingSystem.Linq;
usingSystem.Threading.Tasks;
usingMicrosoft.AspNetCore.Mvc;
usingPhotostudio_NewStandart.Models;
usingPhotostudio_NewStandart.MSSQLConnection;
```

```
usingPhotostudio_NewStandart.OtherModels;

namespacePhotostudio_NewStandart.Controllers
{
publicclassHomeController : Controller
    {
        [HttpGet]
publicIActionResultIndex()
        {
returnView();
        }

        [HttpGet]
publicViewResultStudioRent()
        {
returnView();
        }

        [HttpPost]
publicIActionResultStudioRent(Order order)
        {

DaoFactorydbConnection = newDaoFactory();
order.Status = new Status();
order.Status.IdStatus = 1;
for (inti = 0; i< 4; i++)
            {
if (order.Halls[i])
                {
order.ResultingCost += int.Parse(((order.EndHallsRent - order.StartHallsRent).TotalHours *
dbConnection.GetHallById(i + 1).Price).ToString());
                }
            }

intidOrder;
boolcompletedCreating;
            List<int>notAvailableHalls;

dbConnection.CreateOrder(order, outidOrder, outcompletedCreating, outnotAvailableHalls);
if (completedCreating)
            {

returnRedirectToAction("Index");
            }
else
            {
return View(order);
            }

        }

publicJsonResultGetHallsPrice()
        {
DaoFactorydbConnection = newDaoFactory();
            List<Hall>listHalls = dbConnection.GetAllHallsPrices();
return Json(listHalls);
        }

publicIActionResultGetFreedomHallTime() // обработка ajax запросов
        {
int id = int.Parse(Request.Query["id"]);
intaction = int.Parse(Request.Query["action"]); // получаем 3 критерия поиска дат по залам
DateTime date = DateTime.MinValue;
if (action != 2)
            {
                date = DateTime.Parse(Request.Query["date"]);
            }
```

```
DateTimedayStartFindWeek;
DateTimedayEndFindWeek;  // переменныехраненияdатзадающихпромежуток

GetStartAndEndWeekDays(date, action, outdayStartFindWeek, outdayEndFindWeek); //
определяемпромежутоквнеделидлявыводанаэкран

varlistDates = newDaoFactory().GetCalendarHall(id, dayStartFindWeek, dayEndFindWeek); //
получаемдатыдлязаполненияkалендаря

            Calendar calendar = newCalendar(id, dayStartFindWeek, dayEndFindWeek,
listDates);
returnPartialView(calendar); // генерация нового календарика для клиента
}
privatevoidGetStartAndEndWeekDays(DateTimedate,intaction,outDateTimestartWeek,
outDateTimeendWeek)
        {
if (action == 1)
            {
startWeek = date.AddDays(-7);
endWeek = date.AddDays(-1).AddHours(23);
            }
elseif (action == 2)
{
DateTimenowDate = DateTime.Now.Date;
intdayOfWeek = (int)nowDate.DayOfWeek;

if (dayOfWeek == 0)
                {
dayOfWeek = 7;
                }

startWeek = nowDate.AddDays(-(dayOfWeek - 1));
endWeek = startWeek.AddDays(6).AddHours(23);
            }
elseif (action == 3)
            {
startWeek = date.AddDays(7);
endWeek = startWeek.AddDays(6).AddHours(23);
            }
else
            {
startWeek = DateTime.MinValue;
endWeek = DateTime.MinValue;
}
        }
    }


}using System;
usingSystem.Collections.Generic;
usingSystem.Linq;
usingSystem.Threading.Tasks;
usingMicrosoft.AspNetCore.Mvc;
usingPhotostudio_NewStandart.Models;
usingPhotostudio_NewStandart.MSSQLConnection;

namespacePhotostudio_NewStandart.Controllers
{
publicclassAdminAccessController : Controller
    {
        [HttpGet]
publicViewResultAdminIndex()
        {
DaoFactorydb = newDaoFactory();
            List<Order> orders = db.GetOrderIdByStatus(4);
```

```
return View(orders);
        }
        [HttpPost]
publicIActionResultAdminIndex(Order order)
        {
returnLocalRedirect($"/AdminAccess/Order?id={order.IdOrder}");
        }

        [HttpGet]
publicViewResultOrder(int id)
        {
DaoFactorydb = newDaoFactory();
            Order order = db.GetOrderById(id);
return View(order);
        }

        [HttpPost]
publicIActionResultOrder(Order order)
        {
DaoFactorydb = newDaoFactory();
            List<int>notAvailableHalls;
boolcompliteWrite = true;

if (order.Status.IdStatus == 1)
            {
db.UpdateOrderForStatus1(order, outnotAvailableHalls, outcompliteWrite);

            }
else
            {
db.UpdateOrderForOtherStatus(order);

            }
if (compliteWrite)
            {
returnRedirectToAction("AdminIndex");
            }
else
            {
return View(order);
            }
        }
    }

using Photostudio_NewStandart.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Data.Common;
using System.Data.SqlClient;

namespace Photostudio_NewStandart.MSSQLConnection
{
    public class DaoFactory
    {
        public Hall GetHallById(int idHall)
        {
SqlConnection con = ConnectorMSSQL.GetDBConnection();
            Hall hall = new Hall();

            string query = $"select * from [hall] where id_hall = {idHall};";
SqlCommandcmd = new SqlCommand(query,con);

con.Open();
            using (DbDataReader reader = cmd.ExecuteReader())
            {
```

```
                if (reader.HasRows)
                {
reader.Read();
hall.IdHall = idHall;
hall.Name = reader.GetString(1);
hall.Description = reader.GetString(2);
hall.Price = reader.GetInt16(3);
hall.NightPrice = reader.GetInt16(4);
                }
            }
con.Close();
            return hall;
        }
        public List<Hall>GetAllHallsPrices()
        {
SqlConnection con = ConnectorMSSQL.GetDBConnection();
            List<Hall>listHalls = new List<Hall>();
            string query = "select id_hall,price,night_price from [hall];";
SqlCommandcmd = new SqlCommand(query, con);
con.Open();
            using (DbDataReader reader = cmd.ExecuteReader())
            {
                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
listHalls.Add(new Hall { IdHall = reader.GetByte(0), Price = reader.GetInt16(1), NightPrice
= reader.GetInt16(2)});
                    }
                }
            }
con.Close();
            return listHalls;

        }
        public List<DateTime[]>GetCalendarHall(int id,DateTimestartWeek, DateTimeendWeek)
        {
SqlConnection con = ConnectorMSSQL.GetDBConnection();
            List<DateTime[]>listDates= new List<DateTime[]>();


            string query = "select [order].start_halls_rent,[order].end_halls_rent from
[hall] " +
                "join [Rented_Hall] on [hall].id_hall = [Rented_Hall].id_hall " +
                "join [order] on [Rented_Hall].id_order = [order].id_order " +
                $"where [hall].id_hall = {id} " +
                $"and [order].id_status in (1,2,3) " +
                $"and([order].start_halls_rent between '{startWeek}' and '{endWeek}');";

SqlCommandcmd = new SqlCommand(query, con);
con.Open();
            using (DbDataReader reader = cmd.ExecuteReader())
            {
                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
listDates.Add(new DateTime[2] { reader.GetDateTime(0), reader.GetDateTime(1) });
                    }
                }
            }
con.Close();
            return listDates;
        }

        public Order GetOrderById(int idOrder)
        {
```

```
SqlConnection con = ConnectorMSSQL.GetDBConnection();
            Order order = null;

            string query = $"select id_order,
id_user,[renter].name,last_name,phone_number,start_order,start_halls_rent, end_halls_rent,
resulting_cost, [order].id_status, [status].name from [order] join [renter] on
[order].id_user = [renter].email join [status] on [order].id_status = [status].id_status
where id_order = {idOrder};";
SqlCommandcmd = new SqlCommand(query, con);

con.Open();
            using (DbDataReader reader = cmd.ExecuteReader())
            {
                if (reader.HasRows)
                {
reader.Read();
                    order = new
Order(reader.GetInt32(0),reader.GetString(1),reader.GetString(2),reader.GetString(3),reader
.GetString(4),reader.GetDateTime(5),reader.GetDateTime(6),
reader.GetDateTime(7),reader.GetInt32(8),reader.GetByte(9),reader.GetString(10));
                }
            }
            query = $"select [hall].id_hall,name from rented_hall join hall on hall.id_hall
= rented_hall.id_hall where id_order = {idOrder}";
cmd.CommandText = query;
            using (DbDataReader reader = cmd.ExecuteReader())
            {
                while (reader.Read())
                {
order.HallRented.Add(new Hall(reader.GetByte(0),reader.GetString(1)));
                }
            }
con.Close();
            for (int i = 0; i<order.Halls.Length; i++)
            {
                for (int j = 0; j <order.HallRented.Count; j++)
                {
                    if (order.HallRented[j].IdHall == i+1)
                    {
order.Halls[i] = true;
                    }
                }
            }
            return order;
        }

        public Order GetOrderByEmailAndId(int id, string email)
        {
SqlConnection con = ConnectorMSSQL.GetDBConnection();
            Order order = null;

            string query = $"select id_order,
id_user,[renter].name,last_name,phone_number,start_order,start_halls_rent, end_halls_rent,
resulting_cost, [order].id_status, [status].name from [order] join [renter] on
[order].id_user = [renter].email join [status] on [order].id_status = [status].id_status
where id_user = '{email}';";
SqlCommandcmd = new SqlCommand(query, con);

con.Open();
            using (DbDataReader reader = cmd.ExecuteReader())
            {
                if (reader.HasRows)
                {
reader.Read();
                    order = new Order(reader.GetInt32(0), reader.GetString(1),
reader.GetString(2), reader.GetString(3), reader.GetString(4), reader.GetDateTime(5),
reader.GetDateTime(6),
```

```
reader.GetDateTime(7), reader.GetInt32(8), reader.GetByte(9), reader.GetString(10));
                }
            }
            query = $"select [hall].id_hall,name from rented_hall join hall on hall.id_hall
= rented_hall.id_hall where id_order = {id}";
cmd.CommandText = query;
            using (DbDataReader reader = cmd.ExecuteReader())
            {
                while (reader.Read())
                {
order.HallRented.Add(new Hall(reader.GetByte(0), reader.GetString(1)));
                }
            }
con.Close();
            return order;
        }

        public List<Order>GetOrderIdByStatus(int idStatus)
        {
SqlConnection con = ConnectorMSSQL.GetDBConnection();
            List<Order>listOrdersId = new List<Order>();

            string query = $"select [order].id_order,[order].id_status,[status].name from
[order] join [status] on [order].id_status = [status].id_status where [order].id_status<
{idStatus};";
SqlCommandcmd = new SqlCommand(query,con);
con.Open();
            using (DbDataReader reader = cmd.ExecuteReader())
            {
                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
listOrdersId.Add(new Order() { IdOrder = reader.GetInt32(0), Status = new Status() {
IdStatus = reader.GetByte(1), NameStatus = reader.GetString(2)} });
                    }
                }
            }
con.Close();
            return listOrdersId;

        }

        public void UpdateOrderForStatus1(Order order,out List<int> _notAvailableHalls, out
bool rewriteOrder)
        {
SqlConnection con = ConnectorMSSQL.GetDBConnection();
SqlCommandcmd = new SqlCommand();

            List<int>tempHallOrdered = new List<int>();

            for (int i = 0; i<order.Halls.Length; i++)
            {
                if (order.Halls[i])
                {
tempHallOrdered.Add(i+1);
                }
            }

            string numberRentedHalls = string.Empty;
rewriteOrder = true;
            _notAvailableHalls = null;


            for (int i = 0; i<order.Halls.Length; i++)
            {
                if (order.Halls[i])
```

```
                {
numberRentedHalls += (i + 1) + ",";
                }
            }
numberRentedHalls = numberRentedHalls.Remove(numberRentedHalls.Length - 1);

            string query = $"select distinct id_hall " +
               $"from [rented_hall] " +
               $"join [order] on [order].id_order = [rented_hall].id_order " +
               $"where '{order.StartHallsRent}' < [order].end_halls_rent " +
               $"and '{order.EndHallsRent}' > [order].start_halls_rent " +
               $"and [rented_hall].id_hall in ({numberRentedHalls}) " +
               $"and [order].id_status not in (5) " +
               $"and [order].id_order not in ({order.IdOrder});";

cmd.CommandText = query;
cmd.Connection = con;
con.Open();
            using (DbDataReader reader = cmd.ExecuteReader())
            {
                if (reader.HasRows)
                {
rewriteOrder = false;
                    _notAvailableHalls = new List<int>();
                    while (reader.Read())
                    {
                        _notAvailableHalls.Add(reader.GetByte(0));
                    }
                }
            }
con.Close();
            if (rewriteOrder)
            {
                query = $"select id_hall from rented_hall where id_order =
{order.IdOrder};";
cmd.CommandText = query;
                List<int>oldRentedHalls = null;
con.Open();
                using (DbDataReader reader = cmd.ExecuteReader())
                {
                    if (reader.HasRows)
                    {
oldRentedHalls = new List<int>();
                        while (reader.Read())
                        {
oldRentedHalls.Add(reader.GetByte(0));
                        }
                    }
                }
con.Close();

                bool delete = true;
                if (oldRentedHalls != null)
                {
                    for (int i = 0; i<oldRentedHalls.Count; i++)
                    {
                        for (int j = 0; j <order.HallRented.Count; j++)
                        {
                            if (oldRentedHalls[i] == tempHallOrdered[j])
                            {
                                delete = false;
tempHallOrdered.RemoveAt(j);
                                break;
                            }
                        }
                        if (delete)
                        {
```

```
                            query = $"delete from [rented_hall] where id_order =
{order.IdOrder} and id_hall = {oldRentedHalls[i]}";
cmd.CommandText = query;
con.Open();
cmd.ExecuteNonQuery();
con.Close();
                        }
                    }
con.Open();
                    for (int i = 0; i<tempHallOrdered.Count; i++)
                    {
                        query = $"insert into [rented_hall] values ({order.IdOrder},
{tempHallOrdered[i]});";
cmd.CommandText = query;
cmd.ExecuteNonQuery();

                    }
con.Close();
                }

                query = $"update [order] set start_halls_rent = '{order.StartHallsRent}',
end_halls_rent = '{order.EndHallsRent}', resulting_cost = {order.ResultingCost} ,id_status
= {order.Status.IdStatus + 1} where id_order = {order.IdOrder}; ";
cmd.CommandText = query;
con.Open();
cmd.ExecuteNonQuery();
con.Close();
            }


        }
        public void UpdateOrderForOtherStatus(Order order)
        {
SqlConnection con = ConnectorMSSQL.GetDBConnection();

            string query = $"update [order] set id_status = {order.Status.IdStatus + 1}
where id_order = {order.IdOrder}";
SqlCommandcmd = new SqlCommand(query,con);
con.Open();
cmd.ExecuteNonQuery();
con.Close();
        }
        public void CancelOrder(Order order)
        {
SqlConnection con = ConnectorMSSQL.GetDBConnection();

            string query = $"update [order] set id_status = 5 where id_order =
{order.IdOrder}";
SqlCommandcmd = new SqlCommand(query, con);
con.Open();
cmd.ExecuteNonQuery();
con.Close();
        }

        public void CreateOrder(Order order, out int _idOrder, out bool _completedCreating,
out List<int> _notAvailableHalls)
        {
SqlConnection conn = ConnectorMSSQL.GetDBConnection();
SqlCommandcmd = new SqlCommand();

            string numberRentedHalls = string.Empty;
            List<int> halls = new List<int>();
            int idOrder = -1;
            List<int>notAvailableHalls = new List<int>();

// форомируем строку запроса для проверки доступности выбраных клиентом залов
```

```
for (int i = 0; i< 4; i++)
            {
                if (order.Halls[i])
                {
numberRentedHalls += (i + 1).ToString() + ",";
halls.Add(i + 1);
                }
            }
numberRentedHalls = numberRentedHalls.Remove(numberRentedHalls.Length-1);
            string query = string.Empty;

            query = $"select distinct id_hall " +
                $"from [rented_hall] " +
                $"join [order] on [order].id_order = [rented_hall].id_order " +
                $"where '{order.StartHallsRent}' < [order].end_halls_rent " +
                $"and '{order.EndHallsRent}' > [order].start_halls_rent " +
                $"and [rented_hall].id_hall in ({numberRentedHalls}) " +
                $"and [order].id_status not in (5);";

cmd.Connection = conn;
cmd.CommandText = query;


            bool continueWrite = true;

conn.Open();

// считываем значение с базы и останавливаем запись если хотя бы один зал занят
using (DbDataReader reader = cmd.ExecuteReader())
            {
                if (reader.HasRows)
                {
continueWrite = false;
                    while (reader.Read())
                    {
notAvailableHalls.Add(reader.GetByte(0));
}
                }
            }
conn.Close();

            _notAvailableHalls = notAvailableHalls;

            // проверяем на доступность записи: если true - начинаем запись; если false -
выходим из записи и возвращаем номера занятых залов;
if (!continueWrite)
            {
Console.WriteLine("Write false");
                _completedCreating = false;
            }
            else
            {

                // проверканаличияклиента в базе

                bool writeRenter = false;
                query = $"select * from [renter] where email = '{order.Renter.Email}';";
cmd.CommandText = query;
conn.Open();
                using (DbDataReader reader = cmd.ExecuteReader())
                {
                    if (!reader.HasRows)
                    {
writeRenter = true;
}
                }
conn.Close();
```

```
                // создание записи клиента в базе если его там нет

if (writeRenter)
                {
                    query = $"insert into [renter] values('{order.Renter.Email}',
'{order.Renter.Name}'," +
                        $" '{order.Renter.SecondName}', '{order.Renter.PhoneNumber}');";
cmd.CommandText = query;
conn.Open();
cmd.ExecuteNonQuery();
conn.Close();
                }


                // создание записи заказа в базе
query = $"insert into [order] values" +
                    $" ('{order.Renter.Email}' ," +
                    " CURRENT_TIMESTAMP ," +
                    $" '{order.StartHallsRent}' ," +
                    $" '{order.EndHallsRent}' ," +
                    $" '{order.ResultingCost}' ," +
                    $" '{order.Status.IdStatus}');";
cmd.CommandText = query;

conn.Open();
cmd.ExecuteNonQuery();
conn.Close();

                // изьятьидзаказа


                query = $"select max(id_order) from [order] where id_user =
'{order.Renter.Email}'; ";
cmd.CommandText = query;
conn.Open();

                using (DbDataReader reader = cmd.ExecuteReader())
                {
reader.Read();
idOrder = reader.GetInt32(0);
                }
conn.Close();

// создание записи принадлежнеости залов к заказу

conn.Open();
                for (int i = 0; i<halls.Count; i++)
                {
                    query = $"insert into [rented_hall] values('{idOrder}','{halls[i]}');";
cmd.CommandText = query;
cmd.ExecuteNonQuery();
                }
conn.Close();
                _completedCreating = true;
            }

            _idOrder = idOrder;

        }


    }
}using System;
usingSystem.Collections.Generic;
usingSystem.ComponentModel.DataAnnotations;
```

```
namespacePhotostudio_NewStandart.Models
{
publicclassOrder
    {
publicintIdOrder{ get; set; }
public User Renter { get; set; }
publicDateTimeStartOrder{ get; set; }
publicDateTimeStartHallsRent{ get; set; }
publicDateTimeEndHallsRent{ get; set; }
publicintResultingCost{ get; set; } = 0;
public Status Status{ get; set; }
publicbool[] Halls { get; set; } = newbool[4] { false, false, false, false};
public List<Hall>HallRented{ get; privateset; } = new List<Hall>();

publicOrder(int _idOrder, string _email, string _nameRenter, string _lastNameRenter, string
_phoneNumber, DateTime _startOrder, DateTime _startHallsRent, DateTime _endHallsRent, int
_resultingCost, int _idStatus, string _nameStatus)
        {
IdOrder = _idOrder;
            Renter = new User(_email,_nameRenter,_lastNameRenter,_phoneNumber);
StartOrder = _startOrder;
StartHallsRent = _startHallsRent;
EndHallsRent = _endHallsRent;
ResultingCost = _resultingCost;
            Status = new Status(_idStatus,_nameStatus);
}

publicOrder()
        {

        }

    }
}
using System;
usingSystem.Collections.Generic;
usingSystem.ComponentModel;
usingSystem.ComponentModel.DataAnnotations;
usingSystem.Linq;
usingSystem.Threading.Tasks;

namespacePhotostudio_NewStandart.Models
{
publicclassUser
    {

publicstring Name { get; set; }

publicstringSecondName{ get; set; }

publicstring Email { get; set; }

publicstringPhoneNumber{ get; set; }

publicUser()
        {

        }

publicUser(string _email, string _name, string _lastName, string _phoneNumber)
        {
            Email = _email;
            Name = _name;
SecondName = _lastName;
PhoneNumber = _phoneNumber;
        }
    }
```

}

Висновок: Під час проекту спроектовано інформаційну систему «Оренда залів фотостудії». Під час проектування проведено аналіз предметної області та визначені основні бізнес-процеси. Розроблено організаційну структуру управління, яка зображує сукупність елементів, що знаходяться між собою у стійких взаємостосунках. Оцінено трудомісткість, тривалість та потреби у співробітниках для проекту *(*Отже порахувавши всі формули проект має 8526 строк коду. Потребує 22,7769 людино-місяців, це означає, що потрібно 8,19951 місяців та 2,77783 людини)

Основним результат проекту є структура пілотного проекту інформаційної системи та сукупність візуальних моделей і пояснень до них, які послідовно виділяють особливості реалізації програмного забезпечення для інформаційної.