МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ Аліна САВЧЕНКО

"____"_____2021 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ

"МАГІСТР"

ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

Тема: " Web-сервіс для менеджменту замовлень ретуш-центра"

Виконавець: Дзьобко Руслан Вікторович

Керівник: к.т.н., доцент Моденов Юрій Борисович

Нормоконтролер: _____Ігор РАЙЧЕВ

Київ – 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології"

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Аліна САВЧЕНКО
«____»_____2021р.

ЗАВДАННЯ

на виконання дипломної роботи студента

Дзьобко Руслана Вікторовича
(прізвище, ім'я, по батькові)

1. **Тема роботи:** «Web-сервіс для менеджменту замовлень ретуш-центра» затверджена наказом ректора від 12.10.2021 р. за № 2228/ст.
2. **Термін виконання роботи:** з 12.10.2021 р. по 31.12.2021 р.
3. **Вихідні дані до роботи:** функціональні вимоги до веб-сервісу, база даних MS SQL Server, фреймфорки: Angular, .NET Core.
4. **Зміст пояснювальної записки:** аналіз веб-сервісів та технологій розробки, аналіз інструментів для генерації звітностей, створення веб-сервісу.
5. **Перелік обов'язкового ілюстративного матеріалу:** Скріншоти користувацького інтерфейсу сервісу та результати генерації звітів.

6. Календарний план-графік

№ з/п	*Завдання*	*Термін виконання*	*Підпис керівника*
1.	Дослідження та аналіз предметної області використання	12.10.2021– 15.10.2021	
2	Опрацювання інформації за тематикою дипломного проекту	18.10.2021– 20.10.2021	
3	Розробка дизайну web-сервісу	20.10.2021– 22.10.2021	
4	Розробка клієнтської і серверної частин модуля	25.10.2021– 03.11.2021	
5	Розробка модуля звітностей	04.11.2021 – 17.11.2020	
6	Написання пояснювальної записки дипломного проекту	17.11.2021– 30.11.2020	
7	Оформлення та друк пояснювальної записки.	02.12.2021 – 11.12.2021	
8	Підготовка демонстраційного матеріалу та доповіді	12.12.2021 – 20.12.2021	

7. Дата видачі завдання: 12.10.21 р.

Керівник дипломної роботи _____ Юрій МОДЕНОВ
(підпис керівника)

Завдання прийняв до виконання _____ Руслан ДЗЬОБКО
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи “Web-сервіс для менеджменту замовлень ретуш-центра” складається із вступу, трьох розділів, загальних висновків, списку використаних джерел і містить 88 сторінок тексту, 52 рисунки та 15 таблиць. Список використаних джерел містить 7 найменувань.

Метою дипломної роботи є аналіз теоретичних та практичних матеріалів для створення веб-сервісу для менеджменту ретуш-центру та генерації звітності.

Об’єктом дослідження є предметна область ретуш центру.

Предметом дослідження є розробка веб-сервісу для ретуш центрів.

Ключові слова: ВЕБ-СЕРВІС, ЗВІТ, БАЗА ДАНИХ, АДМІНІСТРАТОР, РЕТУШЕР.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ 6
ВСТУП 7
РОЗДІЛ 1. ОСНОВНІ АРХІТЕКТУРНІ КОНЦЕПЦІЇ ВЕБ-СЕРВІСУ 11
1.1. Про веб додатки 11
1.2. Клієнт-серверна архітектура 14
1.3. Rest 15
1.4. Багаторівнева архітектура 20
1.5. Бази даних 22
1.6. Авторизація і автентифікація 30
1.7. Бізнес звіти 33
1.8. Клієнтська частина 35
Висновок до розділу 1 37
РОЗДІЛ 2. АНАЛІЗ БІЗНЕС ЧАСТИНИ 38
2.1. Дослідження предметної області ретуш-центра 38
2.2. Опис бізнес моделей системи 40
2.3. ER діаграма 43
2.4. Діаграма варіантів використання 48
Висновок до розділу 2 55
РОЗДІЛ 3. СТВОРЕННЯ ФУНКЦІОНАЛУ САЙТУ ТА РОЗРОБКА МОДУЛЯ ЗВІТНОСТІ 56
3.1. Основні теоретичні відомості про HTML, CSS, JS 56
3.2. Дизайн та функціонал сайту 59
3.3. Генерація звітності 70
Висновок до розділу 3 85
ВИСНОВКИ 86
СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ 88

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – база даних

UML –Unified Modelling Language

URI – Uniform Resource Identifier

HTTP – HyperText Transfer Protocol

XML – Extensible Markup Language

REST – Representational State Transfer

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

SQL – Structured Query Language

SSRS – SQL Server Reporting Services

ВСТУП

В основі предметної області ретуш-центра є можливість менеджменту, прийому замовлень та ведення статистики і збереження інформації по замовленням клієнтів. Використовуючи дане програмне забезпечення, будь-яка компанія, яка займається даним видом роботи може використовувати його для оптимізації процесів та налагодження повноцінного робочого процесу.

Термін «ретуш» означає акт покращення зовнішнього вигляду зображення. У фотографії ретушування означає видалення певних дефектів із зображення. Це можуть бути незначні об’єкти, такі як пил або бруд на об’єктиві або сенсорі камери. Ретушування може бути використано для видалення деяких фізичних дефектів на шкірі моделі, як це часто можна побачити в модних публікаціях. Клієнти звертаються в спеціалізовані ретуш-центри задля такої обробки фото. Ретуш-центр орагнізує взаємодію між замовниками роботи та виконавцями(ретушерами). Найчастіше даний вид обробки фото застосовується для ритуальних послуг, проте може використовуватись і в інших сферах, які потребують певної обробки або виправлення дефектів зображення.

У портретній фотографії в процесі ретуші можна використовувати маски, щоб приховати дефекти, гладку шкіру, а також відбілити зуби. Ширину обличчя можна регулювати. Очі також можна збільшити. Колір волосся також можна змінити. Тіло можна зміцнити. У фотографії продуктів ретушування може видалити відбитки пальців або зробити поверхню виробу більш гладкою.

До продукту можна нанести відблиски для додаткової глибини. Ці коригування застосовуються для того, щоб зробити кінцеве зображення більш привабливим для потенційних клієнтів.

У послугах ретуші фотографій коригування вносяться відповідно до побажань їх клієнтів. У таких випадках отримані зображення повинні бути узгоджені з брендом клієнта.

Клієнт надає менеджеру деякі зображення, а також інформацію необхідну для обробки. До основних деталей відносять – зміну зовнішнього вигляду особи, заміна фону, заміну одягу, тощо. Також надається інформація про необхідні терміни виконання роботи, основні відомості про клієнта – ФІП, країна, місто, деталі з виконання замовлення.

Однією із головних цілей системи є структуроване збереження зображень та виконаної роботи. Всі замовлення зберігаються, і при необхідності клієнта отримати фото старого замовлення (у випадку втарти клієнтом готового продукту), ретуш-сервіс може надати будь-яке замовлення із історії клієнта.

Також, програма надає можливості генерації звітності. Вона включає в себе можливість отримати набір даних за допомогою фільтрації – по даті, по сумі, по статусам, по географічній інформації, тощо. Модуль звітності включає в собі основні базові види звітностей, але за потребностей ретуш-центр може звернутись до розробників задля розширення функціоналу певним набором звітів.

Головною ідеєю є створення універсальної системи для менеджменту, збереження, відстежування стану замовлень для будь якої компанії, яка займається обробкою фотографій. Додаток має бути гнучким у налаштуваннях, легко налаштовуватись під конкретні бізнес задачі, а також мати можливість розширювати види звітностей для конкретного клієнта.

З точки зору розробки, даний модуль буде використовувати самі передові технології веб-додатків, матиме систему авторизації та аутентифікації, розділення користувачів на ролі. Інтерфейс представлятиме найбільш легкий для розуміння та взаємодії процес.

В основі розробки даної системи вибрано розробка багато функціонального Веб-додатку. Він поділяється на дві головні частини – клієнтську(фронт енд) та серверну(бекенд). Для можливості розширення клієнтської взаємодії, наприклад розробки мобільного додатку або десктопної версії, сервер організує взаємодію з клієнтом через Web API.

В основі серверної архітектури застосована багарівнева архітектура. В даному модулі використовується підхід трирівневої архітектури. Ця архітектура надає модель, яка допоможе гнучко розширювати систему, а також легко перевикористовувати певні елементи або модулі. Трирівнева архітектура зазвичай складається з рівня уявлення, рівня бізнес-логіки й рівня зберігання даних.

Для збереження усіх даних буде інтегрована реляційна база даних. З її допомогою можна легко зберігати усі клієнтські дані, налаштування та різні бізнес потреби.

Важливою частиною є модуль генерації звітностей. Бізнес-звіт – це оцінка певного питання, сукупності обставин або фінансових операцій, які стосуються діяльності підприємства. Його головне призначення – надати відповідну інформацію коротко та ефективно. Він часто створюється під потреби керівництва компанії, і часто має форму службової записки з доданим звітом. Бізнес-звіт розроблений у скороченому стилі, що дозволяє читачеві орієнтуватися у ньому швидко та визначити ключові елементи. Він використовує заголовки, підзаголовки, маркери, діаграми та таблиці, щоб передати відповідну інформацію. Ділові звіти можуть варіюватися від коротких звітів на одну-дві сторінки, до звітів на сотню і більше сторінок. Зазвичай він включає в себе наступні елементи : таблиці, рекомендації, висновки, коментарі, дати, розрахунки.

Додаток включатиме в собі велику кількість базових звітів, а також можливість створення нового типу звітності за допомогою замовлення у розробника за потребами.
В розробці використовуються різноманітні архітектурні підходи та патерни проектування.
У програмній інженерії шаблон проектування є загальним повторюваним рішенням поширеної проблеми в розробці програмного забезпечення. Шаблон дизайну — це не готовий дизайн, який можна перетворити безпосередньо в код. Це опис або шаблон для вирішення проблеми, який можна використовувати в багатьох різних ситуаціях.
Шаблони проектування можуть прискорити процес розробки, надаючи перевірені парадигми розробки. Ефективний дизайн програмного забезпечення вимагає розгляду

проблем, які можуть стати видимими лише пізніше в процесі впровадження. Повторне використання шаблонів проектування допомагає запобігти непомітним проблемам, які в майбутньому можуть викликати серйозні проблеми, і покращує читабельність коду для програмістів і архітекторів, знайомих із шаблонами.

Актуальність. На даний момент послуги ретушування є дуже популярними в різноманітних сферах, в особливості в ритуальних послугах. Тому виникає потреба універсального сервісу, який буде задовольняти різноманітним бізнес потребам та буде гнучким в налаштуваннях. Окрім того, дана сфера потребує можливість роботи з різноманітної кількістю бізнес звітів, а також можливості створення нових під конкретні потреби керівництва деякого ретуш-центру.

РОЗДІЛ 1.

ОСНОВНІ АРХІТЕКТУРНІ КОНЦЕПЦІЇ ВЕБ-СЕРВІСУ

1.1. Про веб додатки

Веб-додаток (Web app) — це прикладна програма, яка зберігається на віддаленому сервері та доставляється через Інтернет через інтерфейс браузера. Веб-сервіси за визначенням є веб-програмами, і багато веб-сайтів, хоча й не всі, містять веб-програми.

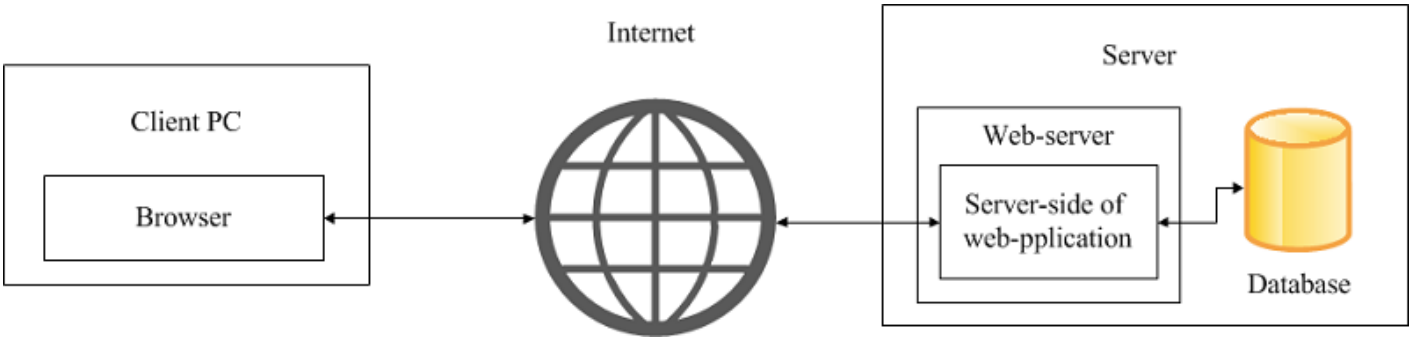

Рис. 1.1. Загальна схема роботи веб-додатку

Можна виділити наступні елементи – клієнтська частина, Інтернет та серверна частина. Веб-програми не потрібно завантажувати, оскільки доступ до них здійснюється через мережу. Користувачі можуть отримати доступ до веб-програми через веб-браузер, такий як Google Chrome, Mozilla Firefox або Safari. Щоб веб-програма працювала, їй потрібні веб-сервер, сервер додатків і база даних. Веб-сервери керують запитами, які надходять від клієнта, тоді як сервер додатків виконує запитане завдання. Базу даних можна використовувати для зберігання будь-якої необхідної інформації.

Виділимо основні переваги, а також і недоліки веб-додатків :

Переваги :

- Рентабельність. Це є однією із головних переваг в будь якій розробці програмного забезпечення. Процес розробки складається зі створення посилань між URL-адресою та програмою, а оскільки це відносно простіше зробити, на розробку потрібно витрачати набагато менше часу.
- Простота оновлення нових версій. Веб додатки не потребують такої частоти оновлення як наприклад десктопні або мобільні версії. Оновлюється лише версія на сервері, а так як всі користувачі отримують доступ через посилання, то всі користувачі працюватимуть з найновішою версією додатку. В порівнянні – мобільні або дестктопні версії потрібно оновлювати вручну через менеджери оновлень такі як Google Play, App Store, Microsoft Store, що ускладнює підтримку нових версій між усіма клієнтами і потребує сумісність різних версій.
- Незалежність від операційної системи. Так як серверна програма працює віддалено, а взаємодія клієнта відбувається за допомогою веб браузера, то будь-яка операційна система, яка підтримує роботу з браузером надає можливість взаємодіяти з різноманітними сайтами.
- Гнучкість та легкість масштабування додатку. Так як оновлення версій для веб додатків є однією з найбільших переваг, то відповідно впровадження нових бізнес потреб спрощується і система стає простіша в розширенні новими модулями. Також, при виникненні проблем, сервер легко можна замінити без особливих складнощів, що зменшує час простою при такому переміщенні.
- Легкість інтеграції веб-сервісів в інших додатках(веб, десктоп, мобільні). Часто, багато веб сервісів розробляються таким чином, щоб можна було використовувати відкритий API інтерфейс для інтеграції певного функціоналу в інших застосунках. Особливо популярним є реалізація автентифікації за допомогою сторонніх

сервісів, наприклад Facebook, Google. До того ж досить просто інтегрувати частини веб-сторінок в мобільні або десктоп застосунки при потребі.

Серед вищезгаданих переваг необхідно виділити декілька недоліків веб-програм :

- Залежність від Інтернет з'єднання. Хоча ми живемо в епоху Інтернету перебої та проблеми з ним явище досить звичайне. А без доступу до нього ми втрачаємо будь-яку можливість взаємодіяти з додатком. Тому для перегляду веб-сайту та завжди необхідне надійне інтернет-з'єднання та висока швидкість.
- Залежність на веб-сайт. Веб-додаток повністю залежить на роботу з браузерами. Це надає величезну кількість переваг, але також є і певні обмеження.
- Веб-програма повністю зав'язана на своєму веб-браузері. Хоча, як правило, це забезпечує купу переваг, повна залежність такого роду також може бути обмежуючим фактором. Якщо веб-сервер виходить з ладу або не відповідає, додаток також не буде працювати. Саме тому, при веб розробці необхідно враховувати багато деталей та правил розробки. Це може бути час завантаження сторінки, проблеми з авторизацією та автентифікацією .
- Швидкість. Іноді веб-додаток може працювати повільніше ніж програма на локальному сервері. Саме тому, в деяких випадках веб не може витіснити мобільні додатки. До того ж , будь який сайт залежить від роботи Інтернету, через що якість роботи може варіюватись.
- Часто веб-програма працює відносно повільніше, ніж програма, розміщена на локальному сервері, і з цих причин не може повністю замінити мобільні програми. Він також безпосередньо пов'язаний з нашим браузером, через що розмір його програми має тенденцію збільшуватися. Тому велика програма працює значно повільніше, ніж настільна програма.
- Безпека. Хоча на даний момент розроблено безліч методологій та технологій захисту веб додатків, вони досі являються найбільш вразливими для хакерів.

1.2. Клієнт-серверна архітектура

Клієнт-серверна архітектура — це обчислювальна модель, в якій декілька компонентів працюють у визначених ролях для комунікації. Сервер хостить, управляє та передає дані для використання клієнтом. В клієнт-серверній архітектурі сервер виступає як виробник, а клієнт — як споживач. Сервер надає клієнтам різноманітні дані та складні обчислення за конкретним запитом. Серед найбільш популярних послуг можуть бути доступ до зовнішніх програм, програм або сервісів, зберігання даних, спільний доступ до файлів, доступ до принтера та/або прямий доступ до необробленої обчислювальної потужності сервера.

Клієнт-серверна архітектура реалізується, коли клієнт надсилає запит на ресурс або процес на сервер через мережеве з'єднання, який потім обробляється та доставляється клієнту. Сервер може одночасно керувати кількома клієнтами, тоді як один клієнт може бути підключений до кількох серверів одночасно, кожен з яких надає різний набір сервісів.

Ось приклад того, як працює комунікація клієнт/сервер. За допомогою браузера для доступу до веб-сайту, користувач або клієнт вводить URL-адресу. DNS-сервер шукає IP-адресу веб-сервера та передає її браузеру. Браузер генерує запит HTTP або HTTPS, а сервер, як генератор даних, надсилає файли. Клієнт отримує їх, а потім, як правило, надсилає наступні запити.

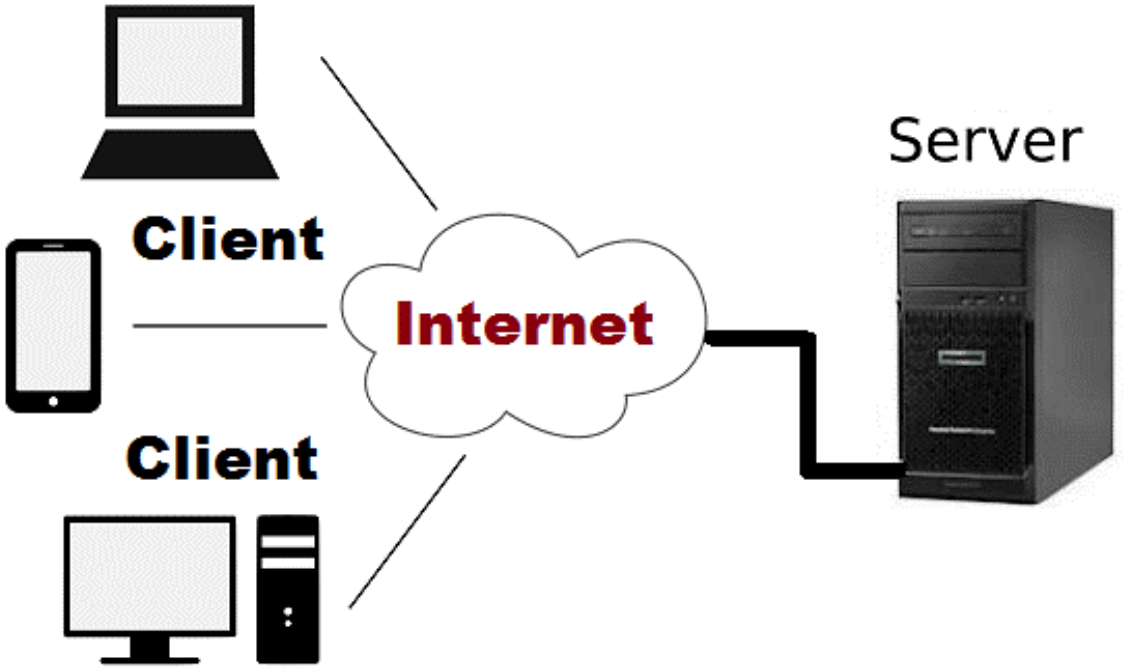

Рис 1.2. Клієнт-серверна архітектура

В ролі клієнта може виступати різноманітні девайси, такі як комп’ютерні додатки, мобільні програми або веб сайти.

1.3. Rest

REST(Representational State Transfer, «передача репрезентативного стану») — це інтерфейс прикладної програми (API), який використовує HTTP-запити PUT, POST та DELETE .

REST, який використовують браузери, можна розглядати як мову Інтернету. Тепер, коли використання хмарних технологій зростає, з’являються різні інтерфейси програмування прикладних програм (API), які відкривають веб-сервіси, а REST є логічним вибором для створення API, які дозволяють кінцевим користувачам

підключатися та взаємодіяти з хмарними сервісами. API RESTful використовуються багатьма сайтами, включаючи Google, Amazon, Twitter і LinkedIn.

Рис. 1.3. Rest

Для REST архітектури притаманні такі основні характеристики як – клієнт-серверна архітектура, відсутність стану, кешування, уніфікований інтерфейс, абстракції.

- Клієнт-серверна архітектура. Архітектура клієнт-сервер, що складається з клієнтів, серверів і ресурсів, із запитами, які керуються через HTTP.
- Відсутність стану. Сервер не зберігає жодного стану про клієнтський сеанс на стороні сервера. Це обмеження називається відсутністю стану. Кожен запит від клієнта до сервера повинен містити всю необхідну інформацію для розуміння

запиту. Сервер не може скористатися перевагами будь-якого збереженого на сервері контексту.

- Кешування. Кешування - збереження відповіді сервера на стороні клієнта, щоб клієнту виконував запит сервера на той самий ресурс. Відповідь сервера має містити інформацію про те, як має виконуватися кешування, щоб клієнт кешував відповідь протягом певного періоду часу або ніколи не кешував відповідь сервера.
- Уніфікований інтерфейс, щоб інформація передавалася у стандартній формі. Для цього потрібно притримуватись декількох правил. Ресурси є ідентифікованими та відокремленими від представлень, надісланих клієнту. Гіпертекст/гіпермедіа доступний, що означає, що після доступу до ресурсу, клієнт повинен мати можливість використовувати гіперпосилання, щоб знайти всі інші доступні на даний момент дії, які він може виконати.
- Багатошарова система.

Ресурс. Архітектура REST розглядає контент як ресурс. Такими ресурсами можуть бути текстові файли, сторінки Html, зображення, відео або динамічні бізнес-дані. Сервер REST просто надає доступ до ресурсів, а клієнт REST отримує доступ і змінює ресурси. Тут кожен ресурс ідентифікується за допомогою URI/глобальних ідентифікаторів Найпопулярнішими представленнями ресурсів є XML і JSON.

Ресурс у REST — схожий до об’єктів в об’єктно-орієнтованому програмуванні або схожий на сутність у базі даних. Після ідентифікації ресурсу необхідно визначити його представлення за допомогою стандартного формату, щоб сервер міг надіслати ресурс у вищезгаданому форматі, а клієнт міг зрозуміти той самий формат.

```xml
<user>
   <id>1</id>
   <name>Mahesh</name>
   <profession>Teacher</profession>
</user>
```

Рис. 1.4. Представлення ресурса користувача у форматі xml

На Рис. 1.4 зображено ресурс користувача, який представляє модель з деякими полями – ідентифікатор, ім’я, професія.

Аналогічне представлення ресурсу можна відтворити за допомогою формату JSON.

```json
{
   "id":1,
   "name":"Mahesh",
   "profession":"Teacher"
}
```

Рис. 1.5. Представлення ресурса користувача у форматі JSON

Адресація відноситься до визначення місцезнаходження ресурсу або кількох ресурсів, що лежать на сервері. Кожен ресурс в архітектурі REST ідентифікується своїм URI (Уніфікований ідентифікатор ресурсу). URI має такий формат – на Рис. 1.6.

```
<protocol>://<service-name>/<ResourceType>/<ResourceID>
```

Рис. 1.6. Стандартний вигляд ресурсу

При створення стандартного URI необхідно враховувати наступні пункти :

- Іменник у множині — використовуйте іменник у множині для визначення ресурсів. Наприклад, ми використовували користувачів для ідентифікації користувачів як ресурсу.
- Уникайте використання пробілів – використовуйте символ підкреслення (_) або дефіс (-), коли використовується довга назва ресурсу. Наприклад, треба використовувати authorized_users замість of authorized%20users.
- Використовуйте малі літери — хоча URI не чутливий до регістру, доцільно зберігати URL-адресу лише малими літерами.
- Підтримуйте зворотну сумісність – оскільки веб-сервіс є загальнодоступною службою, URI, щойно оприлюднений, завжди повинен бути доступним. Якщо URI оновлюється, переспрямуйте старий URI на новий URI, використовуючи код статусу HTTP 300.
- Використовуйте методи доступа HTTP – Завжди використовуйте методи HTTP, наприклад GET, PUT та DELETE, щоб виконувати операції з ресурсом. Недобре використовувати ім’я операції в URI.

Приклад некоректного та коректного найменування адресації вказано на Рис. 1.7.

```
http://localhost:8080/UserManagement/rest/UserService/getUser/1
```

```
http://localhost:8080/UserManagement/rest/UserService/users/1
```

Рис. 1.7. Приклади найменування адресацій

Методи HTTP складають основну частину нашого обмеження «уніфікований інтерфейс» і надають відповідну дію до ресурсу на основі іменника. Основними або найбільш часто використовуваними дієсловами HTTP (або методами, як їх правильно називають) є POST, GET, PUT, PATCH і DELETE. Вони відповідають операціям

створення, читання, оновлення та видалення (або CRUD) відповідно. Є також ряд інших дієслів, але вони вживаються рідше. З цих менш частих методів OPTIONS і HEAD використовуються частіше за інші.

На Рис. 1.8 зображено приклад основних CRUD операцій для ресурсу статті. POST відповідає за створення нового ресурсу, PUT - для оновлення даних існуючої статті, GET – отримання деякої статті або списку, DELETE – видалення деякої статті. В більшості запитів, після імені ресурсу вказується унікальний ідентифікатор – ID, який вказує на конкретний елемент.

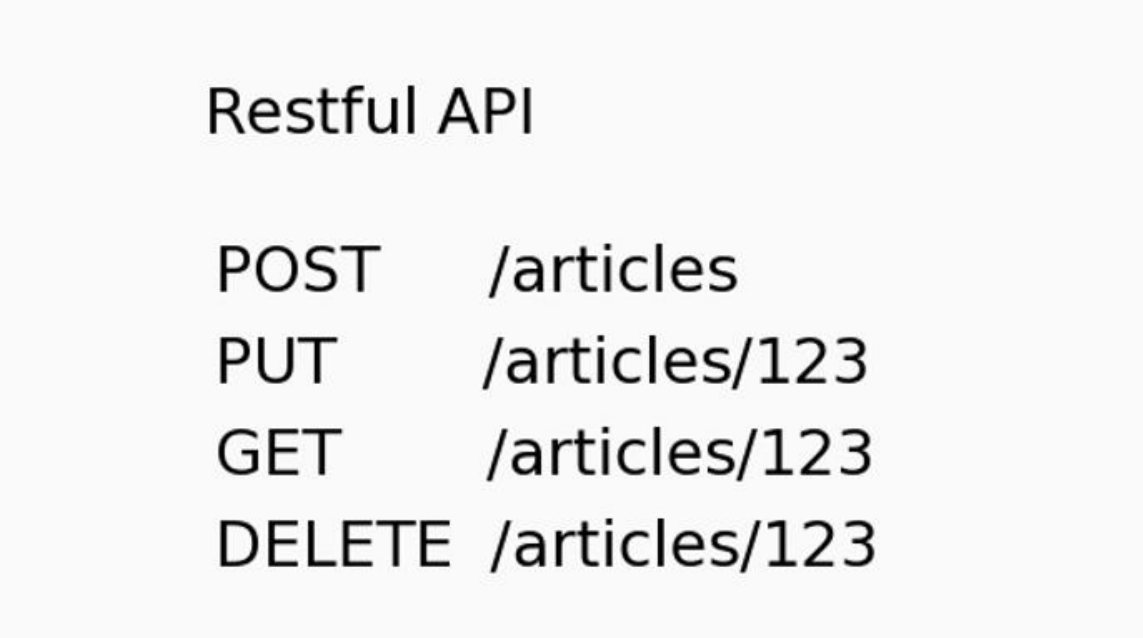

Рис. 1.8. Ресурс

1.4. Багаторівнева архітектура

В основному багаторівнева архітектура — це розділення логіки програми на певні шари. Шаблон архітектури є зрілою архітектурою і просто відноситься до додатків, які розділяють різні логічні рівні на окремі фізичні рівні.

Зазвичай виділяють три основні рівні :

- Інтерфейс користувача

- Бізнес рівень
- Рівень доступу до даних

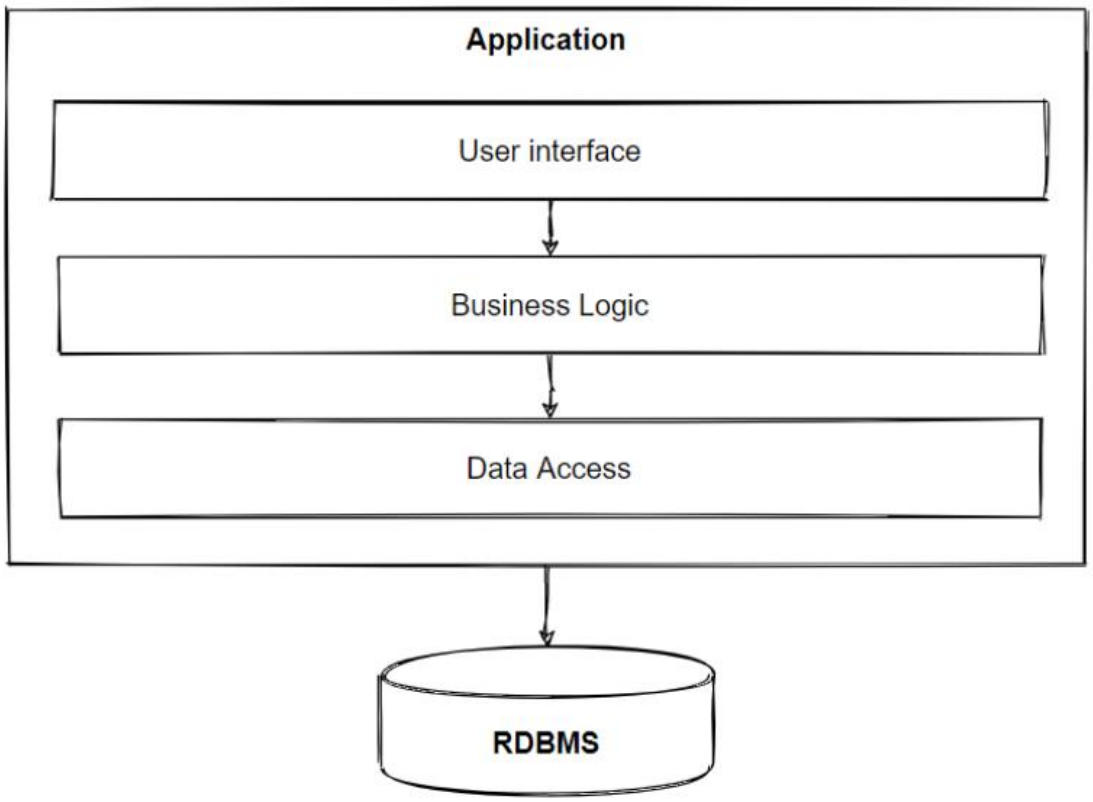

Рис. 1.9. Загальна схема багаторівневої архітектури

На Рис. 1.9 зображені рівні багаторівневої архітектури, в загальному ми можемо описати їх наступним чином :

• Рівень презентації, наприклад веб-програма.
• Бізнес-рівень, включаючи реалізації варіантів використання та надати їх, наприклад REST API.
• Рівень даних, наприклад база даних SQL.

Інші рівні можуть включати проміжне програмне забезпечення, пакетну обробку та API. Важливо відзначити, що шари є логічними. Хоча вони розроблені ізольовано, усі вони можуть бути розгорнуті на одній цільовій платформі.

Рівень презентації складається із інтерфейсу взаємодії з клієнтом – контролери та їх методи, моделі представлення та їх валідація. Також може описувати систему автентифікації/авторизації та мати основні налаштування веб-серверу. Взаємодіє з бізнес рівнем через інтерфейс сервісів. Бізнес рівень містить основну логіку домена(предметна область), проводить основну обробку даних, валідацію, обрахунки. Взаємодіє з рівнем доступу до даних для отримання або внесення даних в БД.

Рівень доступу даних служить для взаємодії між доменом програми та базою даних. Основна задача – отримувати, записувати, змінювати, видаляти дані в БД за допомогою мапінга моделей бази даних на моделі програми.

1.5. Бази даних

Дані – це сукупність окремої невеликої одиниці інформації. Їх можна використовувати в різноманітних формах, як-от текст, числа, медіа, байти тощо. Їх можна зберігати на паперових чи електронній пам’яті тощо.

Слово «Дані» походить від слова «datum», що означає «окрема частина інформації». Це множина від слова дані. У обчислювальній техніці дані – це інформація, яку можна перевести у форму для ефективного переміщення та обробки. Дані є взаємозамінними.

База даних — це організована колекція даних, що забезпечує легкий доступ до них і керування ними. Можна впорядковувати дані в таблиці, рядки, стовпці та індексувати їх, щоб полегшити пошук відповідної інформації. Розробники баз даних створюють базу даних таким чином, що лише один набір програмного забезпечення забезпечує доступ до даних усім користувачам. Основне призначення бази даних — оперувати великим обсягом інформації шляхом зберігання, вилучення й керування даними.

Існує багато доступних баз даних, таких як MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server тощо. Сучасні бази даних керуються системою управління базами даних (СУБД). SQL або мова структурованих запитів використовується для роботи з даними, що зберігаються в базі даних.

Рис. 1.10. Види баз даних

Реляційна база даних – це найпопулярніша модель даних, що використовується в розробці. Вона заснована на роботі з SQL. Дані зберігаються в різних таблицях, кожна з яких має первинний ключ, завданням якого є ідентифікація кожного рядка. Таблиці або файли з даними називаються відносинами, які допомагають позначити рядок або запис, а стовпці посилаються на атрибути або поля. Кілька прикладів: MYSQL, база даних Oracle , сервер Microsoft SQL і DB2.

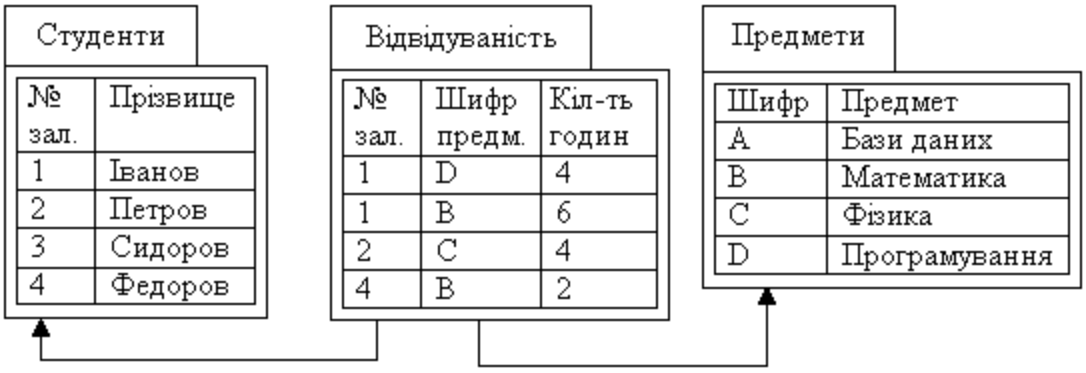

Рис. 1.11. Реляційна база даних

Об’єктно-орієнтована база даних – інформація тут представлена у формі об’єкта, який використовується в об’єктно-орієнтованому програмуванні. Він додає функціональність бази даних до мов об’єктного програмування. Він вимагає менше коду, використовує більше природних даних. Прикладом є ObjectDB (програмне забезпечення ObjectDB).

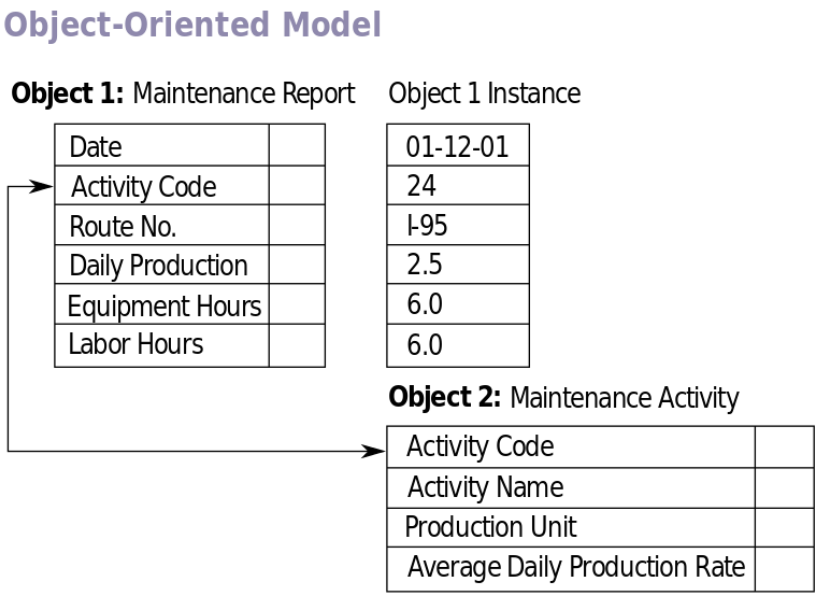

Рис. 1.12. Об’єктно-орієнтована БД

Ієрархічна база даних – у ній інформація про групи батьківських або дочірніх відносин присутня в записах, що подібне до структури дерева. Тут дані слідують за серією записів, набір значень, доданих до них. Вони використовуються в розробці на платформах мейнфреймів. Прикладами є IMS (IBM), реєстр Windows (Microsoft).

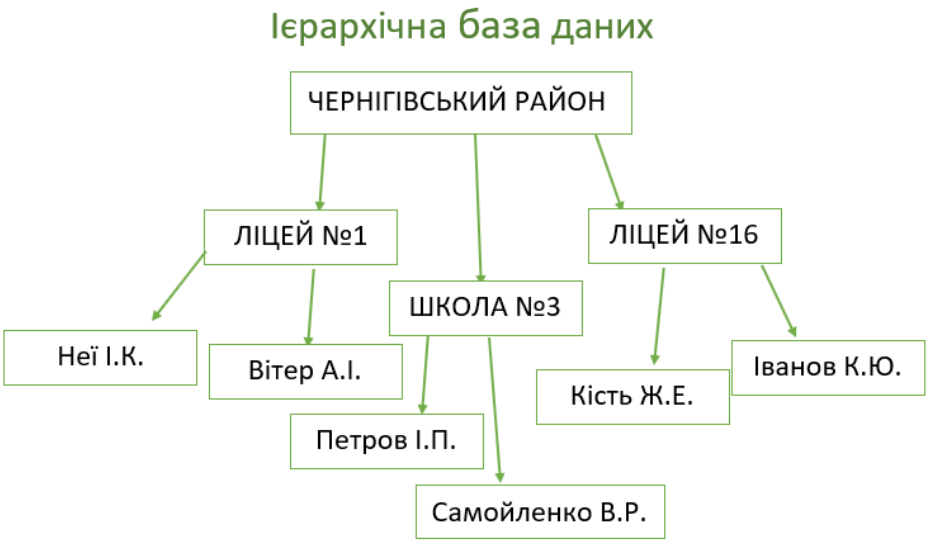

Рис. 1.13. Ієрархічна БД

Мережева база даних – в основному використовується на великих цифрових комп'ютерах. Якщо підключень більше, то ця база даних ефективна. Вони схожі на ієрархічну базу даних, виглядають як павутина або взаємопов'язана мережа записів. Прикладами є CA-IDMS (комп'ютерні партнери), IMAGE (HP).

Рис. 1.14. Мережева БД

Також класифікація може бути за характеристикою кількості користувачів.

- Один користувач – як вказує сама назва, може підтримувати лише одного користувача одночасно. Здебільшого використовується з персональним комп'ютером, на якому зберігаються дані, доступним для однієї особи. Користувач може розробляти, підтримувати та писати програми для баз даних.
- Декілька користувачів – підтримує кілька користувачів одночасно. Дані можуть бути як інтегрованими, так і спільними, база даних повинна бути інтегрована, коли ту саму інформацію не потрібно записувати в двох місцях. Наприклад, студент у коледжі повинен мати базу даних, що містить його інформацію. Він повинен бути доступним для всіх пов'язаних з ним відділів. Наприклад, відділ бібліотеки та відділ плати повинні мати інформацію про студентську базу даних. Тому в такому випадку ми можемо інтегрувати, і навіть якщо база даних знаходиться лише в одному місці, обидва відділи матимуть до неї доступ.

На основі сайтів, через які розповсюджена мережа :

- Централізована система баз даних – СУБД і база даних зберігаються на одному сайті, який також використовується кількома іншими системами. Можна просто сказати, що дані тут зберігаються на централізованому сервері.
- Паралельна мережева база даних – перевага цієї системи полягає в підвищенні швидкості введення та виведення обробки. В основному використовується в програмах, які мають запити до більшої бази даних. Він утримує декілька центральних процесорів і дисків для зберігання даних паралельно. Система розподіленої бази даних – у цих даних і програмне забезпечення СУБД розподілені на кількох сайтах, але підключені до одного комп'ютера.

Виходячи з вартості :

- Низька вартість СУБД – вартість цих систем варіюється від $100 до $3000.
- Середня вартість СУБД – вартість варіюється від $10000 до $100000.
- Висока вартість СУБД – вартість цих систем зазвичай перевищує 100 000 доларів США.

Microsoft SQL Server — це система керування реляційною базою даних від Microsoft. Система розроблена і побудована для управління та зберігання інформації. Система підтримує різні операції бізнес-аналітики, аналітичні операції та обробку транзакцій. Інформація, що зберігається на сервері, зберігається в реляційній базі даних. Однак, оскільки система — це набагато більше, ніж база даних, вона також складається з системи керування. SQL - Structured Query Language, комп'ютерна мова, яка керує сервером і адмініструє його. Існує багато версій сервера SQL, кожна наступна версія є вдосконаленою моделлю свого попередника.

Microsoft SQL Server має безліч додатків у діловому світі. Перший і найочевидніший — база даних використовується для зберігання та керування інформацією. Проте компанії, які зберігають конфіденційну інформацію клієнтів, таку як особисті дані, дані кредитної картки та іншу конфіденційну інформацію, отримають

переваги від підвищеної безпеки. Система також дозволяє обмінюватися файлами даних комп'ютерами в одній мережі, що підвищує надійність. Сервер SQL також використовується для збільшення швидкості обробки даних, що дозволяє з легкістю виконувати великі операції. Завдяки інформації, що зберігається в базі даних, підприємства матимуть надійну систему резервного копіювання.

Рис. 1.15. Емблема SQL Server

Переваги :

- Програмне забезпечення для управління корпоративним рівнем. Microsoft SQL Server включає професійне програмне забезпечення для керування базами даних корпоративного рівня. Кілька конкурентів, таких як MySQL, розробили подібне програмне забезпечення в останні роки, але Microsoft SQL Server легший у використанні та має більше можливостей. Повна підтримка тригерів, наприклад, підтримується в продукті Microsoft. MySQL нещодавно представив тригери, але вони не підтримуються повністю. Програмне забезпечення, яке пропонує Microsoft, також забезпечує тісну інтеграцію з платформою .NET, що не стосується продуктів-конкурентів.

- Відмінна підтримка відновлення даних. Пошкоджені дані завжди викликають занепокоєння, коли відбувається втрата живлення або неправильне відключення. Microsoft SQL Server має ряд функцій, які сприяють відновленню та відновленню даних. Хоча окремі таблиці не можна створити резервну копію або відновити, доступні варіанти повного відновлення бази даних. Завдяки використанню файлів журналів, кешування та резервного копіювання продукт Microsoft дозволяє вам бути впевненим, що варіантів аварійного відновлення існує безліч.

Недоліки :

- Ціна. Одним із основних недоліків використання Microsoft SQL Server замість альтернативної системи керування реляційною базою даних є те, що варіанти ліцензування досить дорогі. Хоча використання програмного забезпечення з метою розробки або навчання є безкоштовним, будь-яке ділове використання вимагає ліцензійної плати. Для SQL Server 2008, наприклад, SQL Server Standard Edition коштує 7171 доларів США за процесор. Згідно з веб-сайтом Microsoft, видання SQL Server Datacenter коштує 54 990 доларів США за процесор. Для малого бізнесу та приватних осіб, які керують комерційними веб-сайтами, це недоступно. Конкуруюче програмне забезпечення, таке як MySQL, часто безкоштовне для використання. Однак у тих випадках, коли це не так, найдорожчий пакет MySQL Enterprise коштує 4999 доларів США на сервер щороку. Це значно дешевше, ніж навіть пакет Microsoft SQL Standard Edition.
- Обмежена сумісність. Microsoft SQL Server призначений лише для роботи на серверах на базі Windows. З різних причин, включаючи витрати на ліцензування та проблеми безпеки, розробники можуть вирішити розмістити свої веб-сайти на машинах на базі Unix. У цьому випадку вони не зможуть використовувати SQL Server. Конкуруючі продукти часто можуть працювати на інших платформах. На відміну від Microsoft SQL Server, MySQL підтримується на всіх основних платформах, включаючи Windows, Linux, Mac OSX та інші варіанти Unix. На

додаток до неможливості запуску на платформах, відмінних від Windows, також можуть виникнути проблеми з сумісністю щодо взаємодії з програмами, які працюють на інших платформах.

1.6. Авторизація і автентифікація

Аутентифікація – це процес ідентифікації користувачів і підтвердження того, ким вони себе видають. Одним з найбільш поширених і очевидних факторів для аутентифікації особи є пароль. Якщо ім’я користувача збігається з обліковими даними пароля, це означає, що ідентичність дійсна, і система надає користувачеві доступ.

Цікаво, що підприємства, які перебувають без пароля, використовують сучасні методи аутентифікації, як-от одноразові паролі (OTP) за допомогою SMS або електронної пошти, єдиного входу (SSO), багатофакторної автентифікації (MFA) та біометричних даних тощо, щоб автентифікувати користувачів і розгорнути безпеку, яка виходить за рамки того, що зазвичай надають паролі.

Популярні методи аутентифікації :

- Аутентифікація на основі пароля — це простий метод аутентифікації, який вимагає введення пароля для підтвердження особи користувача.
- Аутентифікація без пароля – це коли користувача перевіряють за допомогою одноразової паролі або магічного посилання, що надходить на зареєстровану електронну пошту або номер телефону.
- 2FA/MFA вимагає більше одного рівня безпеки, наприклад, додатковий PIN-код або таємне запитання, щоб ідентифікувати користувача та надати доступ до системи.
- Єдиний вхід (SSO) дозволяє користувачам отримувати доступ до кількох програм за допомогою одного набору облікових даних.

- Соціальна автентифікація перевіряє та автентифікує користувачів із наявними обліковими даними з платформ соціальних мереж.

Популярні методи авторизації :

- Контроль доступу на основі ролей (RBAC) може бути реалізований для керування привілеями від системи до системи та від користувача до системи.
- Веб-токен JSON (JWT) є відкритим стандартом для безпечної передачі даних між сторонами, і користувачі авторизуються за допомогою пари відкритих і закритих ключів.
- SAML – це стандартний формат єдиного входу (SSO), в якому інформація про автентифікацію обмінюється через документи XML, підписані цифровим підписом.
- Авторизація OpenID перевіряє ідентичність користувача на основі аутентифікації сервера авторизації.
- OAuth дозволяє API автентифікувати та отримати доступ до запитуваної системи або ресурсу.

Рис. 1.16. Різниця між авторизацією та автентифікацією

В даній системі для автентифікації використовується JWT Web Token. При успішній автентифікації користувача в системі, на стороні сервера генерується унікальний код в вигляді Base64. Він складається з 3 основних частин заголовка, вмісту та підпису. Заголовок описує алгоритм шифрування та тип токена, вміст включає деякий набір даних про користувача – ім'я, пошта, роль. Для генерування підпису – заголовок та вміст кодується, а потім хешується за допомогою секретного ключа, визначено на стороні серверу.

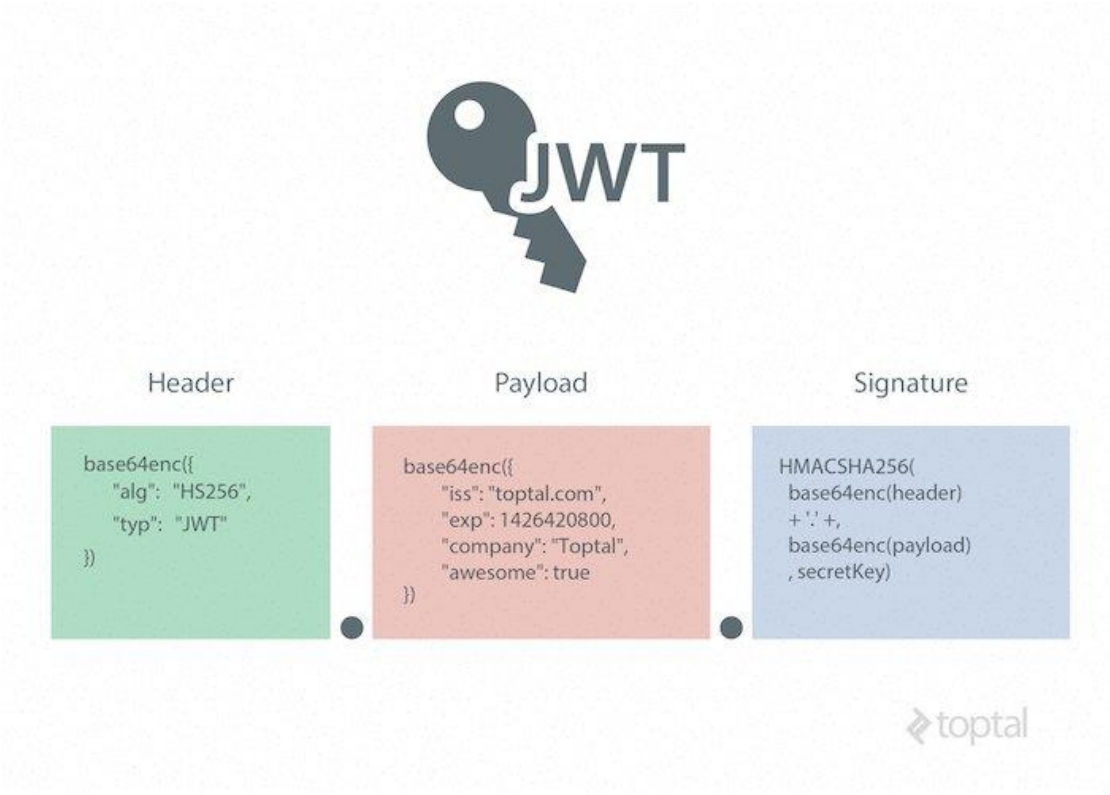

Рис. 1.17. Структура токена

Сервіс може надавати різноманітні права та функціонал, в залежності від ролі. Для коректної авторизації на сервері використовується .NET Identity Server, який надає готові методи для зберігання користувачів, їхніх даних – паролі, нікнейми, ролі. Identity

має функціонал для визначення ролей користувачів та визначення прав на певний ресурс.

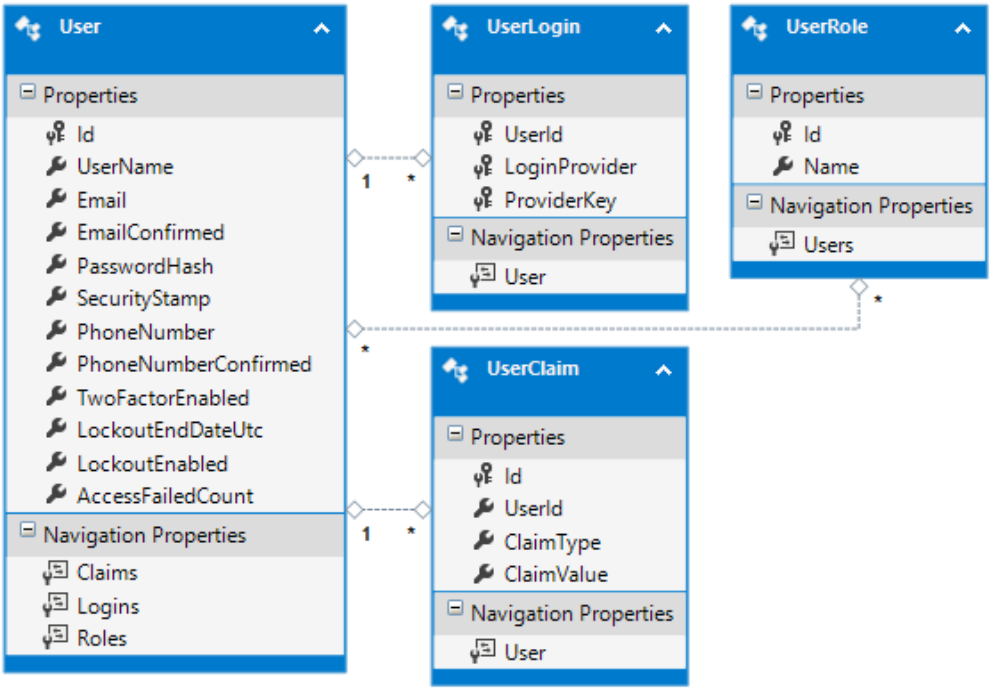

Рис. 1.18. Базова схема БД Identity

Базова схема БД представляє таблиці для зберігання користувачів, ролей та прав.

1.7. Бізнес звіти

Бізнес-звіт – це оцінка певного питання, сукупності обставин або фінансових операції, які стосуються діяльності підприємства. Його головне призначення – надати відповідну інформацію коротко та ефективно. Він часто створюється під потреби керівництва компанії, і часто має форму службової записки з доданим звітом.

Служби звітів SQL Server (SSRS) — це програмне забезпечення для створення звітів, яке дозволяє створювати відформатовані звіти з таблицями у вигляді даних, графіків, зображень і діаграм. Ці звіти розміщуються на сервері, який можна виконувати

в будь-який час за допомогою параметрів, визначених користувачами. Він є частиною пакету служб Microsoft SQL Server.

Переваги SSRS:

- SSRS – це покращений інструмент у порівнянні з Crystal Reports
- Швидша обробка звітів як щодо реляційних, так і багатовимірних даних
- Дозволяє користувачам краще та точніший механізм прийняття рішень
- Дозволяє користувачам взаємодіяти з інформацією без залучення IT-спеціалістів
- Він забезпечує підключення до World Wide Web для розгортання звітів. Таким чином, звіти можна отримати через Інтернет
- SSRS дозволяє експортувати звіти в різних форматах. Ви можете надсилати звіти SSRS за допомогою електронної пошти
- SSRS надає безліч функцій безпеки, які допомагають вам контролювати, хто має доступ до якого звіту

SSRS має досить складну архітектуру. Архітектура служб звітів включає засоби розробки, засоби адміністрування та засоби перегляду звітів.Ось важливі компоненти SSRS :

- Конструктор звітів . Це спеціальний інструмент для публікації звітів, який виконується на комп’ютері клієнта. Він має простий у використанні інтерфейс перетягування.
- Конструктор звітів. Інструмент конструктора звітів допомагає розробляти всі типи звітів. Це інструмент для публікації, який розміщується у Visual Studio або Business Intelligence Development Studio (BIDS).
- Менеджер звітів. Менеджери звітів перевіряють звіт, узгоджуючи його з заданими вимогами. Вони приймають рішення на основі цих звітів.
- Сервер звітів. Це сервер, який використовує механізм баз даних SQL Server для зберігання інформації метаданих.

- База даних сервера звітів. У ньому зберігаються метадані, визначення звітів, ресурси, параметри безпеки, дані доставки тощо.
- Джерела даних. Служби звітності отримують дані з джерел даних, таких як реляційні та багатовимірні джерела даних.

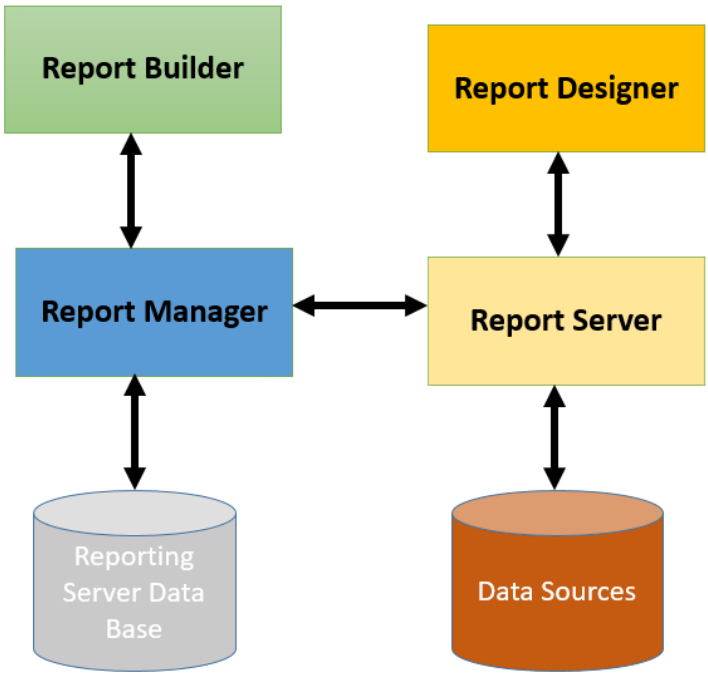

Рис. 1.19. Архітектура SSRS

1.8. Клієнтська частина

На стороні клієнтської частини використовується веб фреймворк Angular. Angular (зазвичай так називають фреймворк Angular 2 або Angular 2+, тобто вищі версії) — написаний на TypeScript front-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій. Angular — це AngularJS, який був переосмислений та перероблений тією ж командою розробників. Ангуляр надає можливості легко

створювати інтерактивні сайти, використовуючи готові бібліотеки, які спрощуються створення дизайну, логіку роботи сайту та взаємодію із сервером. Для створення сучасного та зручного дизайну використовується бібліотека Bootstrap.

Основною ідеєю та перевагою використання фреймворку є концепція, що кожна модель має свою розмітку, стилі, а також скриптовий код, що в свою чергу надає можливість перевикористовування одного і того ж елемента багато разів надаючи йому різні параметри. Наприклад, кожен товар із списку Інтернет магазину представляє деякий компонент із різними атрибутами(назва, характеристики, фото).

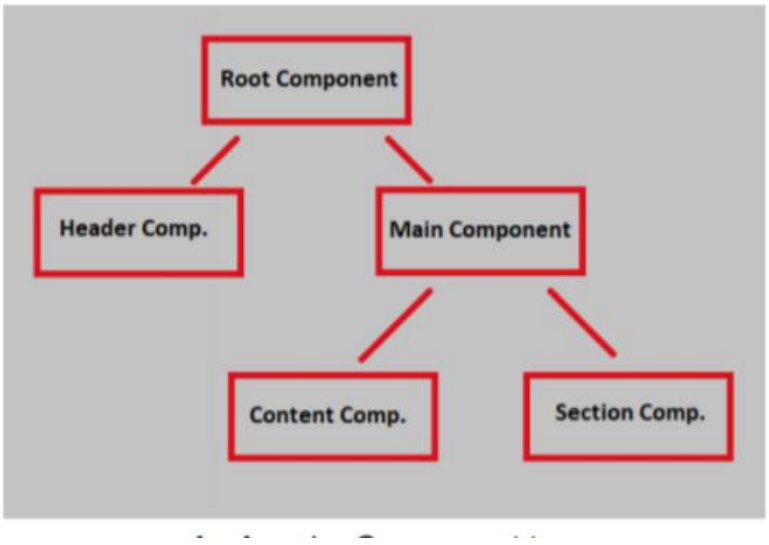

Рис. 1.20. Дерево компонентної структури Ангуляра

Висновок до розділу 1

Даний розділ описує головні технології та інструменти, які необхідні для розробки сервісу ретуш-центру.

На даний момент веб-розробка – найбільш пріоритетний та поплулярний вид розробки для бізнес цілей. Це зумовлюється такими факторами як рентабельність, простота оновлень, незалежність від операційної системи. Для розробки використовуються самий новий стек технологій. Для серверної частини - .NET Core, клієнтської – Angular, для репортингу – SSRS, для автентифікації та авторизації – JWT Web Token та .NET Identity. Було описано короткі характеристики кожної технології, переваги та недоліки.

Перед вибором даних технологій була проведена детальна робота по вивченні актуальності, доцільності використання та легкості підтримки в майбутньому.

РОЗДІЛ 2.

АНАЛІЗ БІЗНЕС ЧАСТИНИ

2.1. Дослідження предметної області ретуш-центра

В основі предметної області ретуш центра є можливість менеджменту, прийому замовлень та ведення статистики та збереження інформації по замовленням клієнтів. Використовуючи данне програмне забезпечення, будь-яка компанія, яка займається даним видом роботи може використовувати його для оптимізації процесів та налагодження повноцінного робочого процесу.

Ретуш (фр. retouche) — виправлення зображень (малюнків, фотознімків тощо) за допомогою програмного забезпечення або промальовуванням їх олівцями чи фарбами, вискоблюванням окремих ділянок або хімічною обробкою. У поліграфії застосовується для підготовки оригіналів і виправлення негативів та діапозитивів, призначених для фотомеханічного виготовлення друкованих форм. Дана технологія часто використовується для обробки, редагування та нанесення певних ефектів перед відтворенням зображення на пам`ятники. Клієнти звертаються в спеціалізовані ретуш-центри задля такої обробки фото. Ретуш центр орагнізує взаємодію між замовниками роботи та виконавцями(ретушерами).

Клієнт надає менеджеру деякі зображення, а також інформацію необхідну для обробки. До основних деталей відносять – зміну зовнішнього вигляду особи, заміна фону, заміну одягу , тощо. Також надається інформація про необхідні терміни виконання роботи, основні відомості про клієнта – ФІО, країна, місто, деталі з виконання замовлення.

Адміністратор центру вносить дані в систему. Спочатку зберігаються дані про клієнта, після чого генерується замовлення, яка може складатись з багатьох пунктів(фотографій). Адміністратор, в залежності від типу, дати викоання та складності роботи виставляє ціну виконання за кожну фотографію, система виставляє загальну суму по замовленню.

В залежності від історії замовлень клієнта, в системі можна налаштовувати логіку нарахування знижок та бонусів. Такі знижки можуть бути визначені адміністратором в залежності від бізнес особливостей конкретного ретуш центру.

Після генерації замовлення, воно з'являється в списку для виконання. В свою чергу, ретушери, які працюють з даною системою, мають доступ до даної сторінки, де вони можуть вибрати собі нове завдання на вибір. Крім того, закріплення ретушера може відбуватись за вибором адміністратора, в залежності від навантаженості ретушерів. Або може бути застосований змішаний підхід, який дозволяє використання обох стратегій. Коли замовлення потрапляє до конкретного ретушера, він переводить статус роботи в “In Progress” , тим часом може приступати до роботи по ретушуванню зображення.

Після того, як обробка фото завершена, ретушер на сторінці даного замовлення прикріпляє зображення та переводитьс статус роботи на “Очікування підтвердження від клієнта”. Наступним кроком адміністатор, завантажує відретушоване зображення та відправляє клієнту на пошту з очікуванням зворотньої відповіді. Якщо клієнт задоволений зробленою роботою, адміністратор переводить статус замовлення на “Виконано”.

Якщо клієнт має певні зауваження до виконання певного зображення, він може надати додаткову інформацію, після чого адміністратор змінює статус правильно виконаних пунктів замовлення на “Виконано”, а пункти, які потребують переробки або виправлення переводить в статус “Допрацювання” в вказує деталі для допрацювання.

Однією із головних цілей системи є структуроване збереження зображень та виконаної роботи. Всі замовлення зберігаються, і при необхідності клієнта отримати

фото старого замовлення (у випадку втарти клієнтом готового продукту), ретуш сервіс може надати будь-яке замовлення із історії клієнта.

Також, програма надає можливості генерації звітності. Вона включає в себе можливість отримати набір даних за допмогою фільтрації – по даті, по сумі, по статусам, по географічній інформації, тощо. Модуль звітності включає в собі основні базові види звітностей, але за потребностей ретуш центр може звернутись до розробників задля розширення функціоналу певним набором звітів.

2.2. Опис бізнес моделей системи

Моделювання предметної області – це спосіб опису та моделювання сутностей реального світу та зв’язків між ними, які спільно описують проблемний простір предметної області. Отримане на основі розуміння вимог системного рівня, визначення сутностей домену та їх взаємозв’язків забезпечує ефективну основу для розуміння та допомагає практикам розробляти системи для розробки, тестування та поступового розвитку. Оскільки часто існує розрив між розумінням проблемної області та інтерпретацією вимог, моделювання предметної області є основною областю моделювання в Agile-розробці в масштабі. Частково керуючись підходами до об’єктно-орієнтованого проектування, моделювання домену передбачає рішення як набір об’єктів домену, які співпрацюють для виконання сценаріїв системного рівня. При розробці даного модулю розроблено різні моделі, які представляють бізнес дані.

Таблиця 2.1

Опис моделі Особа

Атрибут	Тип даних	Коментарі
Id	integer	Унікальний ідентифікатор
First Name	string	Ім’я
Last Name	string	Прізвище
City	string	Місто
Country	string	Країна

Дана модель представляє основні характеристики особи і вситупає як базова модель, яку наслідують інші моделі в системі.

Таблиця 2.2

Опис моделі Клієнт

Атрибут	Тип даних	Коментарі
Email	string	Email клієнта
Phone	string	Телефон клієнта

Дана модель представляє основні характеристики, які визначатимуть дані клієнта ретуш центру. Дана модель наслідує базову модель Person.

Таблиця 2.3

Опис моделі Ретушер

Атрибут	Тип даних	Коментарі
Hired	datetime	Дата приняття на роботу
Fired	datetime	Дата звільнення

Дана модель представляє основні характеристики, які визначатимуть дані ретушера. Дана модель наслідує базову модель Person.

Таблиця 2.4

Опис моделі Підпункт

Атрибут	Тип даних	Коментарі
Id	integer	Унікальний ідентифікатор
RetoucherId	integer	Зовнішній ключ – зв’язок з таблицею Retoucher
Price	decimal	Ціна за виконання роботи
StartedAt	date	Початок виконання ретушером
FinishedAt	date	Дата закінчення роботи
PlanDate	date	Запланована дата виконання
StatusId	integer	Зовнішній ключ – зв’язок з таблицею Status
OrderId	integer	Зовнішній ключ – зв’язок з таблицею Order
Name	string	Імена осіб на зображенні
Comment	string	Коментарі для виконання роботи
PhotoId	integer	Зовнішній ключ – зв’язок з таблицею Photo

Дана модель представляє бізнес модель замовлення клієнта ретуш центру. Кожне замовлення має дані про клієнта, дані про виконавця(ретушера) якого вказує адміністратор, ціну замовлення.

- StartedAt – дата початку виконання роботи ретушером, оновляється ретушером або адміністратором.
- FinishedAt – дата закінчення роботи ретушером, дане значення оновлюється адміністратором.
- PlanDate – запланована дата виконання, вказується клієнтом як бажана дата виконання, але може бути оговорена з адміністратором, якщо вказані терміни не підходять.
- OrderId – зв’язує з таблицею замовлення. Ця структура являє собою список підпунктів, із одного замовлення клієнта.
- RetoucherId - зв’язує з таблицею ретушера. Кожне замовлення пов’язане з одним клієнтом, проте його підпункти можуть бути назначені адміністратором для виконання одним ретушером або різними.
- Name – поле даних, яке зберігає ім’я/ імена людей на зображенні.
- Comment – коментарі для виконання, вказуються клієнтом для ретушера.
- PhotoId - зв’язує з таблицею фото. При створенні замовлення, клієнт надсилає ретуш центру зображення, яке використовується для обробки ретушером.
- StatusId - зв’язує з таблицею статусів. При створенні підпункту

Таблиця 2.5

Опис моделі Фото

Атрибут	Тип даних	Коментарі
Id	integer	Унікальний ідентифікатор
Link	string	Шлях до зображення

2.3. ER діаграма

Діаграма зв’язків між об’єктами (ER) — це тип блок-схеми, яка ілюструє, як «суб’єкти», такі як люди, об’єкти або поняття, пов’язані один з одним у системі.

Діаграми ER найчастіше використовуються для проектування або налагодження реляційних баз даних у сферах програмної інженерії, бізнес-інформаційних систем, освіти та досліджень. Також відомі як ERD або моделі ER, вони використовують певний набір символів, таких як прямокутники, ромби, овали та сполучні лінії, щоб відобразити взаємозв'язок сутностей, відносин та їхніх атрибутів. Вони відображають граматичну структуру з сутностями як іменники, а відносини як дієслова.

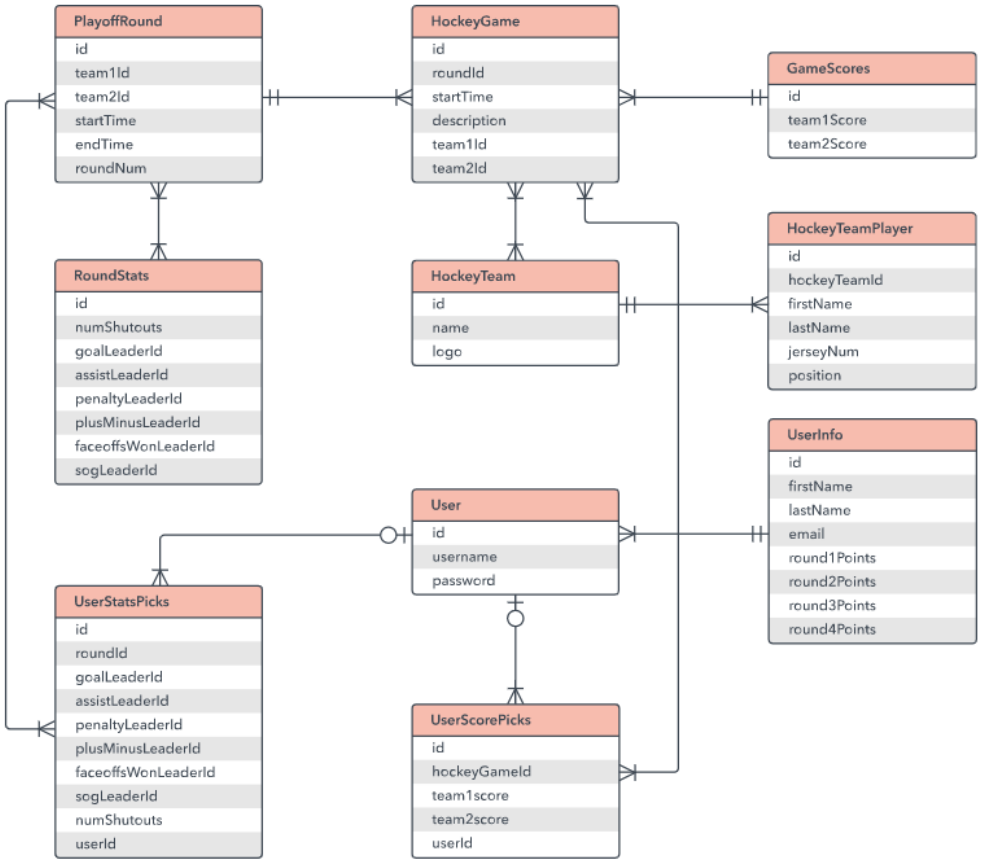

Рис. 2.1. Приклад ER діаграми

Діаграми ER складаються з сутностей, зв'язків та атрибутів. Вони також зображують потужність, яка визначає відносини в термінах чисел. Ось глосарій:

- Суб'єкт

- Об’єкт, який можна визначити, наприклад людина, об’єкт, концепція чи подія, про який можуть зберігатися дані. Подумайте про сутності як іменники. Приклади: клієнт, студент, автомобіль чи продукт. Зазвичай показано у вигляді прямокутника.
- Тип сутності: група визначених речей, таких як студенти або спортсмени, тоді як сутність буде конкретним студентом або спортсменом. Інші приклади: клієнти, автомобілі чи продукти.
- Набір об’єктів: той самий, що й тип сутності, але визначений у певний момент часу, наприклад учні, зараховані до класу в перший день. Інші приклади: клієнти, які придбали минулого місяця, автомобілі, зареєстровані у Флориді. Пов’язаним терміном є екземпляр, у якому конкретна особа або автомобіль буде екземпляром сукупності об’єктів.

Потужність визначає можливу кількість випадків в одній сутності, яка пов’язана з кількістю випадків в іншому. Наприклад, ОДНА команда має БАГАТО гравців. При наявності в ERD сутність Команда і Гравець взаємопов’язані відносинами «один до багатьох».

На діаграмі ER потужність представлена як гусяча лапка на кінцях з'єднувача. Три поширені кардинальні відносини: один до одного, один до багатьох і багато до багатьох.

Приклад потужності один до одного. Відношення «один до одного» здебільшого використовується для поділу сутності на дві частини, щоб надати інформацію коротко та зробити її більш зрозумілою. На малюнку нижче показано приклад зв’язку один до одного.

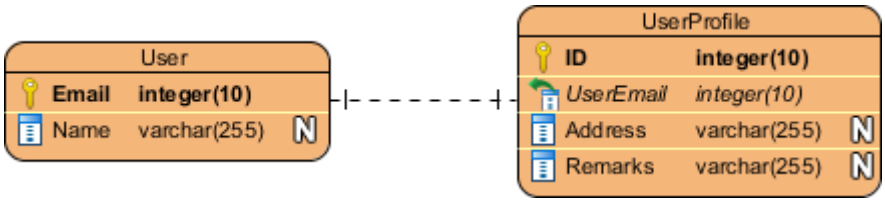

Рис. 2.2. Зв’зок один до одного

Приклад потужності "один до багатьох".Відношення «один до багатьох» відноситься до відносин між двома сутностями X і Y, у яких екземпляр X може бути пов'язаний з багатьма екземплярами Y, але екземпляр Y пов'язаний лише з одним екземпляром X. На малюнку нижче показано приклад відносини один до багатьох.

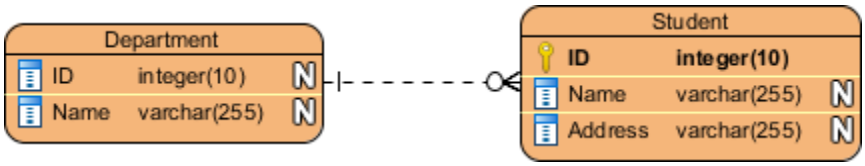

Рис. 2.3. Зв’язок один до багатьох

Відношення багато до багатьох відноситься до відносин між двома сутностями X і Y, в яких X може бути пов'язаний з багатьма екземплярами Y і навпаки. На малюнку нижче показано приклад відношення багато до багатьох. Зауважте, що відношення «багато до багатьох» розбивається на пару зв’язків «один до багатьох» у фізичному ERD. У наступному розділі ви дізнаєтеся, що таке фізична ERD.

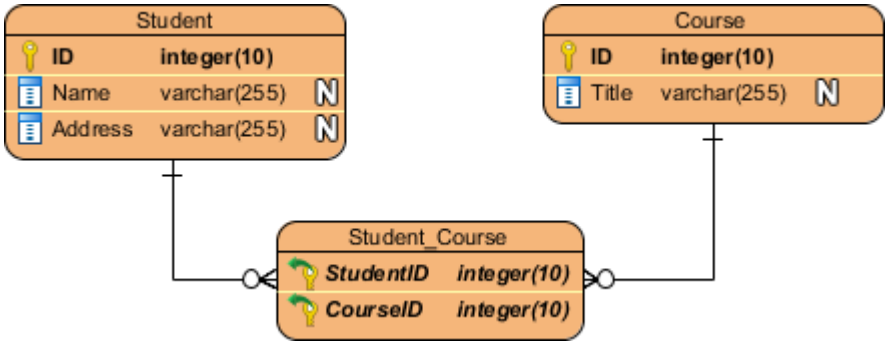

Рис. 2.4. Зв’язок багато до багатьох

На Рис наведено ER діаграму даного модуля.

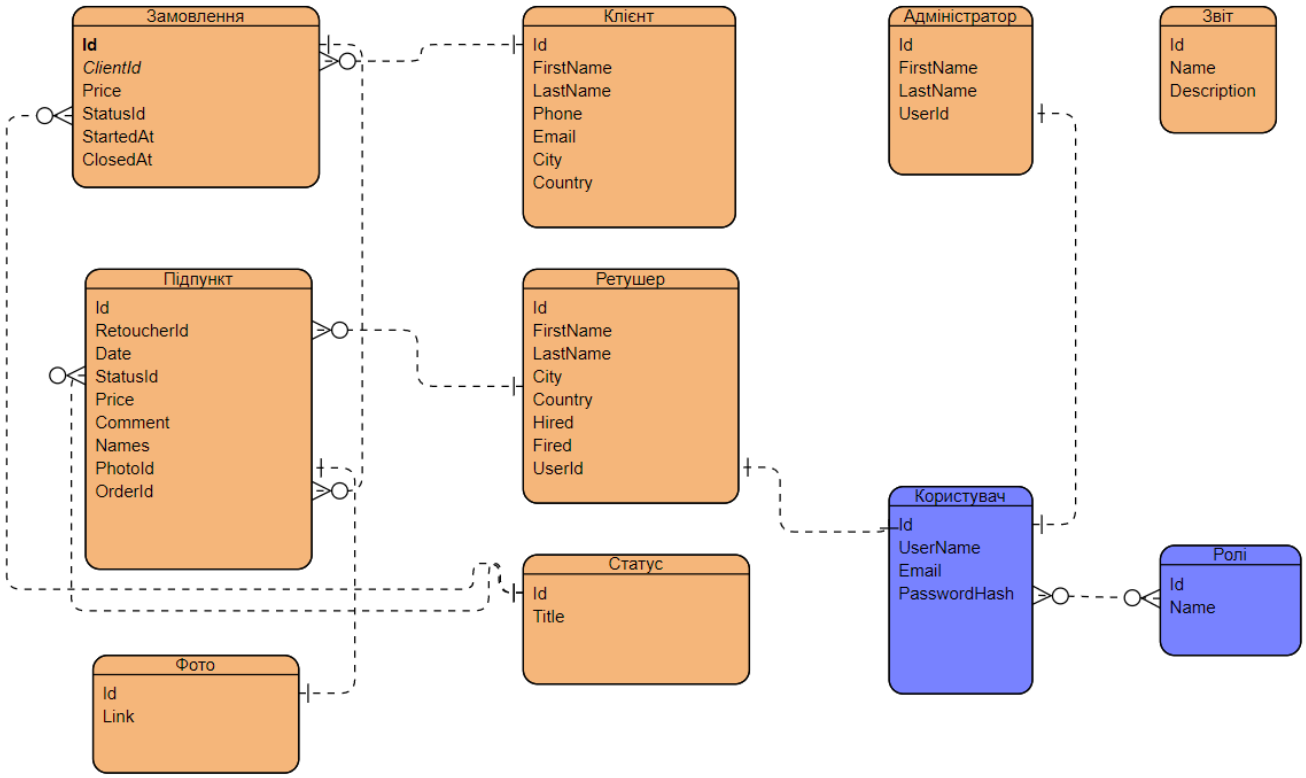

Рис. 2.5. ER діаграма ретуш-центру

Таблиця 2.6

Зв'язки у ER діаграмі

Таблиця 1	Таблиця 2	Зв'язок
Замовлення	Підпункт	One to many
Замовлення	Статус	One to many
Замовлення	Клієнт	One to many
Підпункт	Ретушер	One to many
Підпункт	Статус	One to many
Підпункт	Фото	One to one
Адміністратор	Користувач	One to one

Ретушер	Користувач	One to one
Користувач	Ролі	Many to many

2.4. Діаграма варіантів використання

В UML діаграми варіантів використання моделюють поведінку системи та допомагають охопити вимоги системи.

Діаграми варіантів використання описують функції високого рівня та область застосування системи. Ці діаграми також визначають взаємодії між системою та її акторами. Варіанти використання та дійові особи на діаграмах варіантів використання описують, що робить система і як учасники її використовують, але не те, як система працює всередині.

Діаграми варіантів використання ілюструють і визначають контекст і вимоги або всієї системи, або важливих частин системи. Ви можете змоделювати складну систему за допомогою однієї діаграми варіантів використання або створити багато діаграм варіантів використання для моделювання компонентів системи. Зазвичай ви розробляєте діаграми варіантів використання на ранніх етапах проекту і звертаєтеся до них протягом усього процесу розробки.

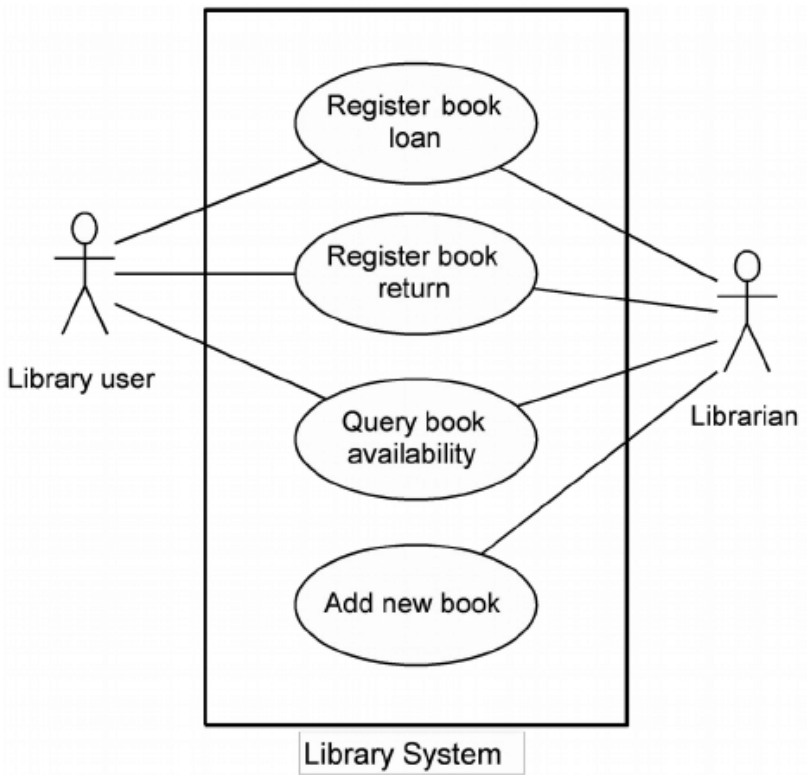

Рис. 2.6. Приклад Use case діаграми

В системі представлено дві основні ролі – адміністратор та ретушер. Для них побудуємо відносну Use case діаграму, яка зображена на Рис. 2.7.

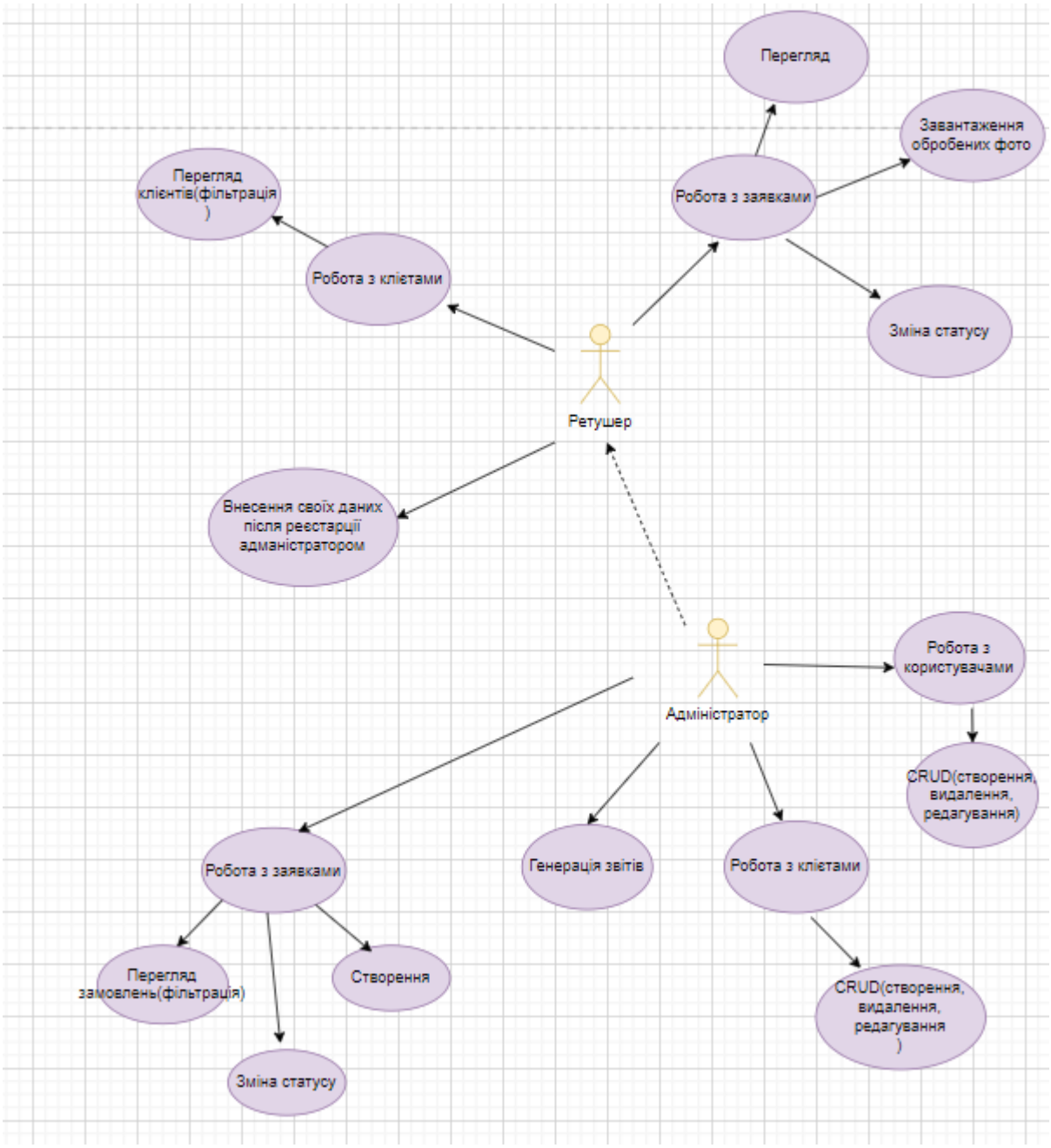

Рис. 2.7. Use Case діаграма ретуш-сервісу

Таблиця 2.7

Опис дійових осіб

Дійова особа	Варіанти використання
Ретушер	Вносить дані після реєстрації адміністратором
	Перегляд клієнтів, фільтрація
	Перегляд замовлень
	Завантаження оброблених фото
	Зміна статусу замовлень
Адміністратор	Створення заявок
	Перегляд замовлень
	Зміна статусу замовлень
	Створення нових користувачів
	Редагування даних користувачів
	Видалення користувачів
	Генерація звітів
	Створення нових клієнтів
	Редагування даних клієнтів
	Видалення клієнтів

Таблиця 2.8

Опис варіантів використання

Назва	Опис
Вносить дані після реєстрації адміністратором	Внесення даних користувачем після створення користувача адміністратором
Перегляд клієнтів, фільтрація	Можливість переглядати клієнтів, шукати їх за вибраними критеріями, фільтрація
Перегляд замовлень	Можливість переглядати дані замовлення та його пункти
Завантаження оброблених фото	Після виконання роботи, ретушер має завантажити готове фото в систему для адміністратора
Зміна статусу замовлень	Можливість зміни статусу замовлення або його пунктів
Створення заявок	Можливість створення заявки, внесення усіх даних, завантаження вихідного зображення, встановлення термінів та визначення виконавця роботи
Створення нових користувачів	Створення акаунтів для ретушерів, виконує адміністратор
Редагування даних користувачів	Зміна даних ретушерів
Видалення користувачів	Видалення аккаунта ретушера
Генерація звітів	Можливість генерації бізнес звітів, з використанням фільтрації
Створення нових клієнтів	Можливість створення нового клієнта
Редагування даних клієнтів	Можливість редагування даних клієнта
Видалення клієнта	Можливість видалення клієнта

Структура даного модуля представлена на Рис. 2.8. Можна поділити на 2 частини – клієнтську та серверну. До клієнтської відносимо – додаток на Ангулярі. До серверної – Web API, бібліотеки .NET Core та провайдер БД.

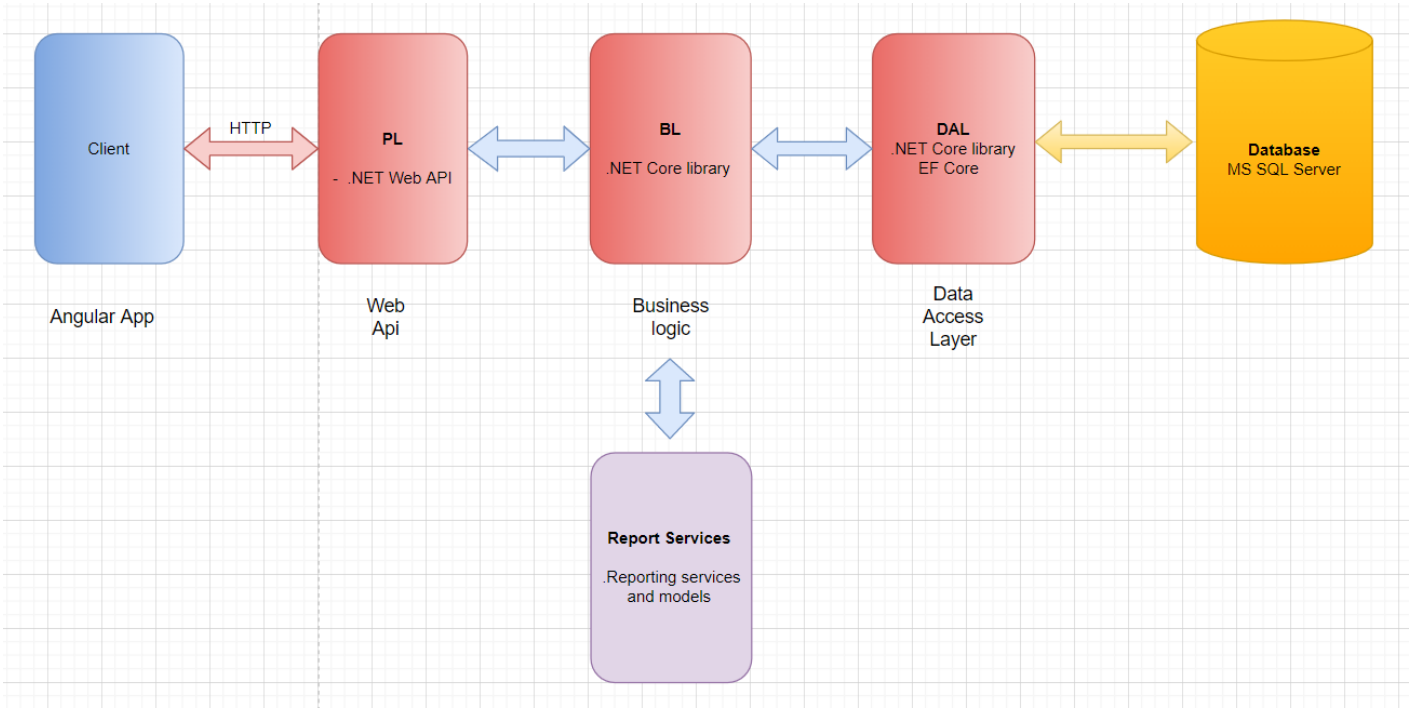

Рис. 2.8. Діаграма програмного модулю ретуш-сервісу

Опишемо детально кожний підпункт в таблиці 2.9.

Таблиця 2.9

Структура програмних модулей системи

Рівень	Технологія	Призначення
Client – Angular App	Angular Single Application	Представляє клієнтську частину додатка. Описує веб-сторінки, стилі, моделі, форми, скрипти, точки взаємодії із сервером.
PL(Web Api)	.NET Core Web API	Точка взаємодії із сервером. Описує основні контролери взаємодії, точки для авторизації

		та автентифікації, моделі представлення та їх валідація.
BL(Business logic)	.NET Core library	Основна бізнес логіка додатку, взаємодія із рівнем даних, валідація та мапінг даних, взаємодія із сервісами звітів.
Reporting Services	.NET Core library	Зберігає шаблони звітів, логіку обрахунків даних та їх мапінг з шаблонами.
DAL (Data Access Layer)	.NET Core library	Взаємодіє з БД. Описує основні моделі, контекст, міграції.
Database	SQL Server	Провайдер БД.

Висновок до розділу 2

Даний розділ має на меті надати основні завдання з точки зору бізнес процесі в системі. Описаний основний сценарій роботи додатку, окреслені основні цілі та можливості для користувача. Проведена розробка діаграм сутностей для БД, діаграма використання для бізнесу та діаграма представлення модулей. Описані основні моделі в системі, їхні атрибути та призначення. Проведено опис дійових осіб та опис варіантів використання.

Надано основну теоретичну інформацію для правильної побудови ER та Use case діаграм, визначено правила та підходи.

РОЗДІЛ 3.

СТВОРЕННЯ ФУНКЦІОНАЛУ САЙТУ ТА РОЗРОБКА МОДУЛЯ ЗВІТНОСТІ

3.1. Основні теоретичні відомості про HTML, CSS, JS

HTML, або мова гіпертекстової розмітки, дозволяє користувачам Інтернету створювати та структурувати розділи, абзаци та посилання за допомогою елементів, тегів та атрибутів.

HTML має багато варіантів використання, а саме:

- Веб-розробка. Розробники використовують HTML-код, щоб розробити скелет сторінки сайту.
- Інтернет навігація. Користувачі можуть легко переміщатися та вставляти посилання між пов'язаними сторінками та веб-сайтами, оскільки HTML активно використовується для вбудовування гіперпосилань.
- Веб-документація. HTML дозволяє впорядковувати та форматувати документи, подібно до Microsoft Word.

Середній веб-сайт включає кілька різних HTML-сторінок. Наприклад, домашня сторінка, сторінка про інформацію та сторінка контактів мають окремі файли HTML.

Документи HTML – це файли, які закінчуються розширенням .html або .htm. Веб-браузер зчитує файл HTML і відтворює його вміст, щоб користувачі Інтернету могли його переглядати.

Усі сторінки HTML мають ряд елементів HTML, що складаються з набору тегів та атрибутів. Елементи HTML є будівельними блоками веб-сторінки. Тег повідомляє веб-браузеру, де починається і закінчується елемент, тоді як атрибут описує характеристики елемента.

Три основні частини елемента:

- Відкриваючий тег – використовується для вказівки, де елемент починає діяти.
- Вміст – це результат, який бачать інші користувачі.
- Закриваючий тег – такий же, як і початковий тег, але з косою рискою перед назвою елемента. Наприклад, </p>, щоб закінчити абзац.

Комбінація цих трьох частин створить елемент HTML:

```
<p>This is how you add a paragraph in HTML.</p>
```

Рис. 3.1. Приклад застосування тегів

Каскадна таблиця стилів або CSS — це мова таблиць стилів, яка визначає, як мають виглядати елементи вашого веб-сайту. Ви можете керувати дизайном, макетом, шрифтом і кольором вмісту свого веб-сайту, вставивши файл CSS у свій HTML-документ.

Давайте подивимося, як працює CSS, розібравши синтаксис:

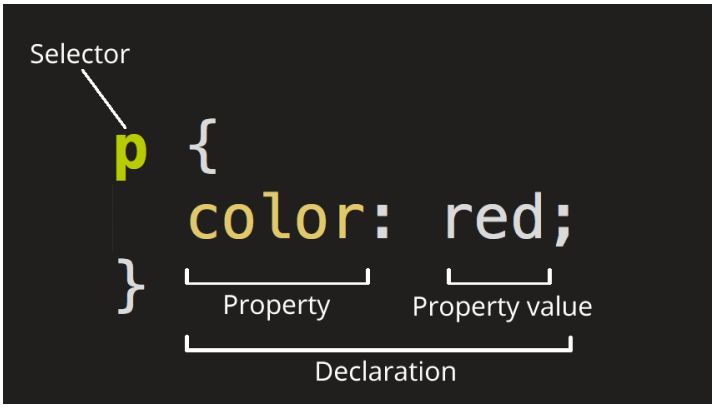

Рис. 3.2. Приклад селектора

Як бачите, синтаксис CSS складається з селектора і блоку оголошення. Селектор — це HTML-елемент, яким ви збираєтеся керувати. Тоді як блок оголошення містить ім'я властивості та значення елемента HTML – обидва вони розділені двокрапкою (:) і оголошені в фігурних дужках ({}).

JavaScript — це легка мова програмування, яку веб-розробники зазвичай використовують для створення більш динамічних взаємодій під час розробки веб-сторінок, програм, серверів і навіть ігор. Розробники зазвичай використовують JavaScript поряд з HTML і CSS. Мова добре працює з CSS у форматуванні елементів HTML. Однак JavaScript все ще підтримує взаємодію з користувачем, чого CSS не може зробити сам.

Розробка фреймворків JavaScript, що складаються з бібліотек коду JavaScript, дозволяє розробникам використовувати попередньо написаний код JavaScript у своїх проектах. Це заощаджує час і зусилля від необхідності кодування функцій програмування з нуля.

Кожна платформа JavaScript має функції, які спрямовані на спрощення процесу розробки та налагодження.

Наприклад, інтерфейсні фреймворки JavaScript, такі як jQuery та ReactJS, покращують ефективність проектування. Вони дозволяють розробникам повторно

використовувати та оновлювати компоненти коду, не впливаючи один на одного, функції чи значення.

```
$(function() {

    function saveState() {
        if (this.options.saveState == true) {
            if (this.state == "normal") {
                $.ajax({
                    url : "bla"
                });
            } else {
                $.ajax({
                    url : "not bla"
                });
            }
        }
    }
})(jQuery);
```

Рис. 3.3. Приклад JS коду

3.2. Дизайн та функціонал сайту

Першою головно точкою веб-додатку буде форма входу в систему. Система передбачає розділення ролей на Адміністраторів та Ретушерів. Для входу розроблена форма, складається з 2 полів та кнопки підтвердження. Дана форма зображена на Рис. 3.4.

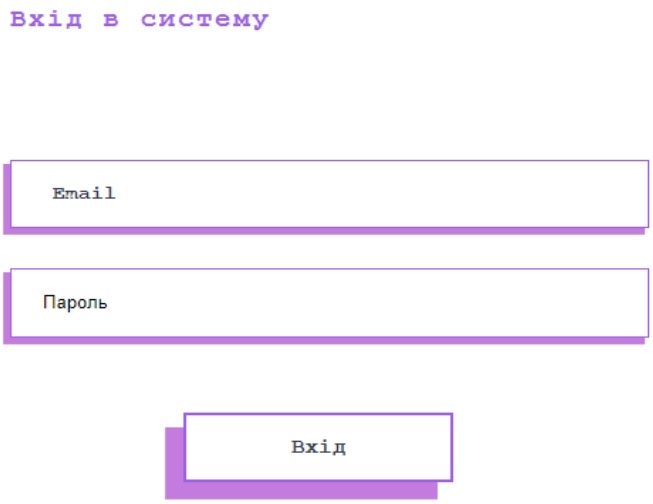

Рис. 3.4. Вхід в систему

Таблиця 3.1

Опис елементів сторінки входу в систему

Елемент	Компонент	Коментарі
Вхід в систему	Заголовок	Стилізований заголовок
Email	Input(type=Email)	Поле, із вказаним типом електроної пошти, передбачає валідацію
Пароль	Input(type=Password)	Поле, із вказаним типом паролю, передбачає валідацію(довжина більше 8 символів, має включати як літери так і числа)
Вхід	Button	Кнопка підтвердження на формі

Компонент NavBar. Представляє основні сторінки для користувачів, в залежності від типу користувача відрізняються підпункти меню. Для Адміністратора доступні меню – Клієнти, замовлення та звіти. Для Ретушера доступне лише Замовлення.

Рис. 3.5. Меню адміністратора

Рис. 3.6. Меню ретушера

Після входу в систему автоматично відкривається меню Замовлення, оскільки це є головний функціонал модуля. Для адміністратора доступні всі пункти замовлень та можливість редагування даних в сітці. Також є можливість створення нового замовлення.

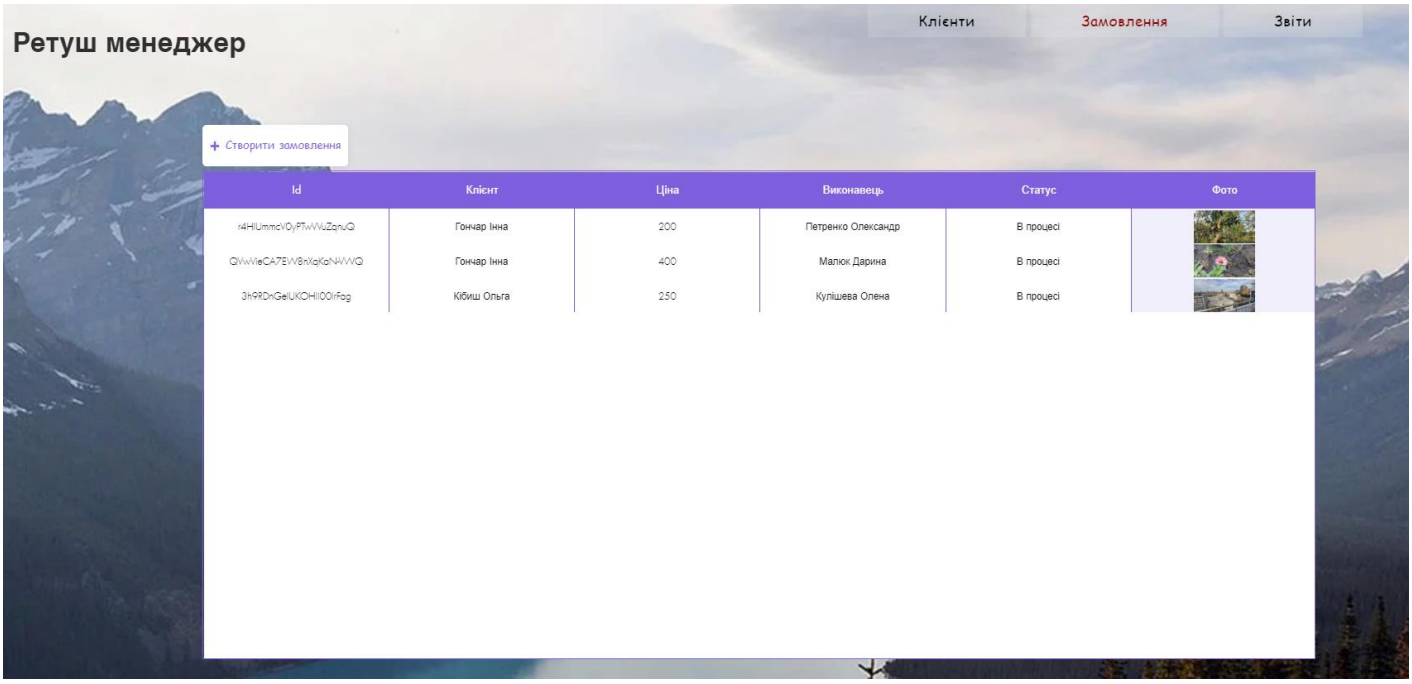

Рис. 3.7. Сторінка замовлень

Для ролі ретушера доступні лише ті пункти замовлення, які закріплені за даним ретушером. Також відсутня можливість редагування даних, окрім зміни статусу. Відсутня можливість створення нового замовлення.

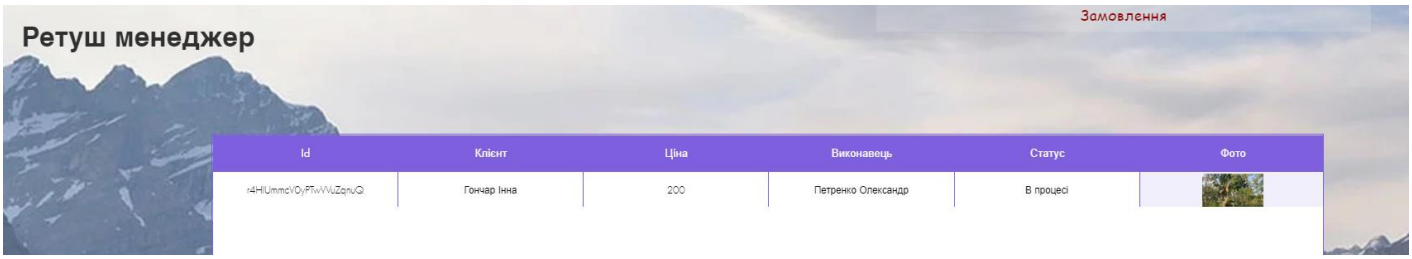

Рис. 3.8. Сторінка замовлень ретушера

При натисканні на рядок унікального ідентифікатора відкривається сторінка з детальною інформацією для підпункту. Для ретушера є можливість перегляду інформації, завантаження оригіналу фото, а також прикріплення готової роботи(фото).

Для адміністратора доступний весь функціонал, а також можливість редагувати дані на сторінці.

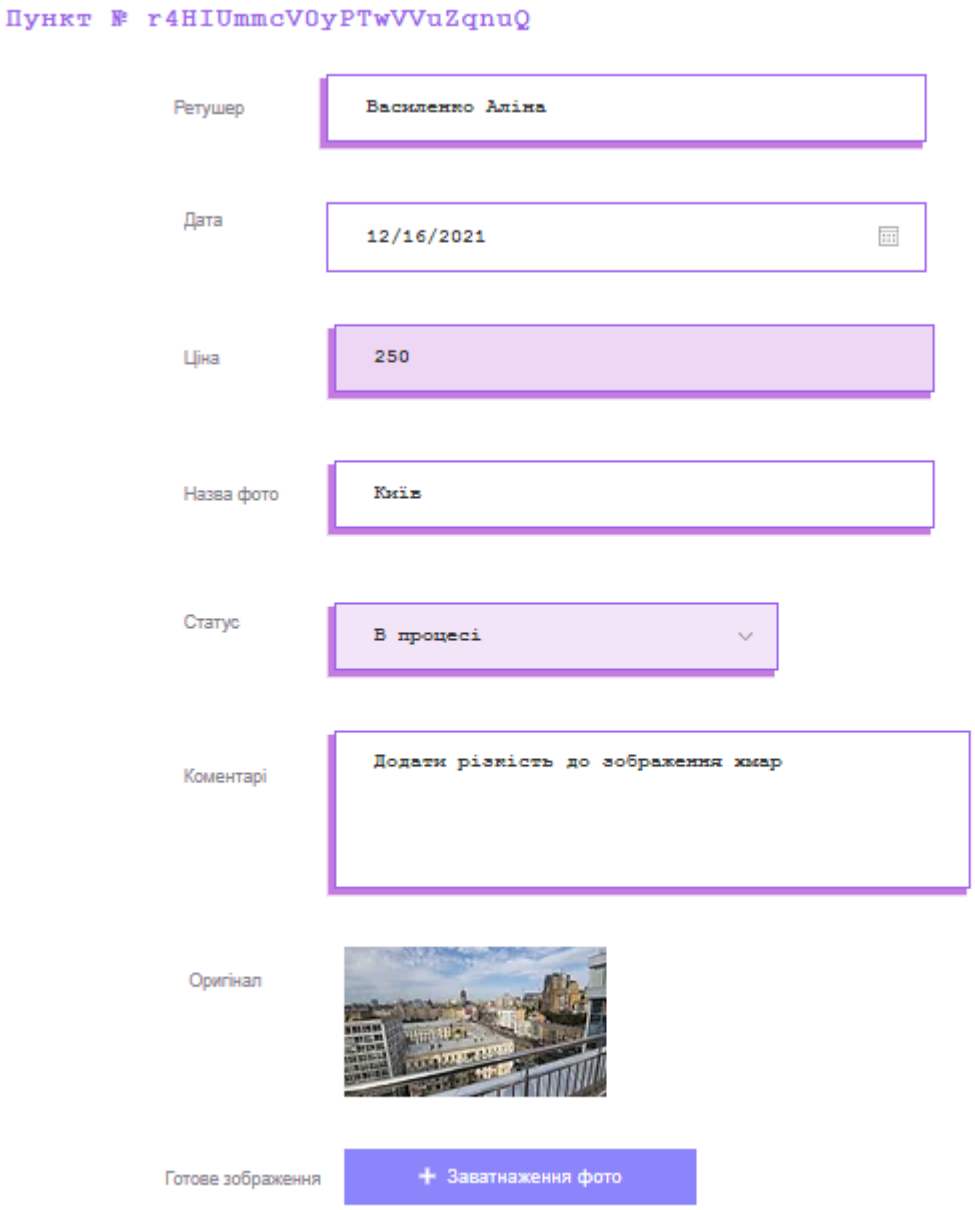

Рис. 3.9. Сторінка підпункту

Таблиця 3.2

Опис елементів сторінки перегляду підпункта

Елемент	Компонент	Коментарі
Пункт №	Стилізований заголовок	Даний заголовок вказує унікальний ідентифікатор(Guid) замовлtyyz
Ретушер	Текстове поле	Вказує ім’я та прізвище ретушера
Дата	Календар	Компонент, який відповідає за бажану дату виконання(вказує клієнт при замовленні)
Ціна	Текстове поле	Ціна за пункт замовлення, вказує адміністратор
Назва фото	Текстове поле	Вказується назва об’єкта зображення або імена людей на фото
Статус	Випадаючий список	Вказує на статус пункта, може змінюватись ретушером або адміністратором
Коментарі	Текстове поле(розширене)	Вказуєтться інформація для виконання ретуші(надається клієнтом), вноситься адміністратором
Оригінал	Компонент фото	Оригінал зображення
Готове зображення	Комопнент готового зображення	Представляє кнопку для завантаження зображення, а також компонент для подальшого його відображення

Для адміністратора, на сторінці замовлення доступна кнопка – створення замовлення

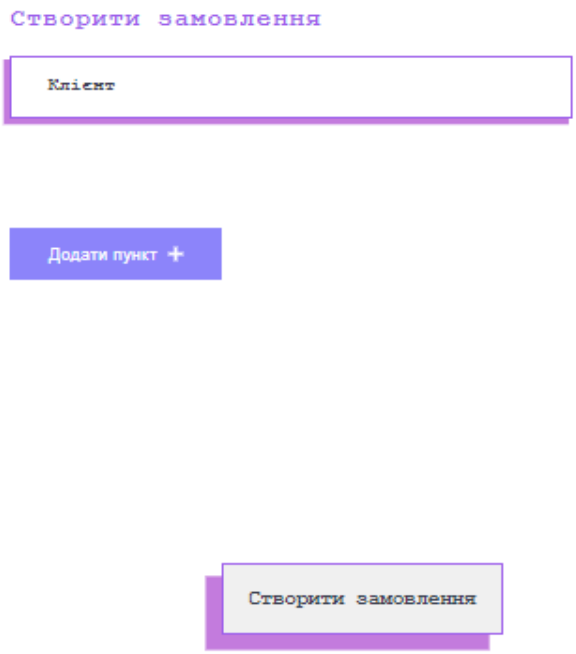

Рис. 3.10. Створення замовлення

Таблиця 3.3

Опис елементів сторінки створення замовлення

Елемент	Компонент	Коментарі
Створити замовлення	Заголовок	Стилізований заголовок
Клієнт	Компонент - Клієнт	Компонент для пошуку клієнта
Додати пункт	Компонент – Створення підпунтку	Компонент, який відкриває нове модальне вікно для створення пункту замовлення

Створити замовлення	Button	Кнопка підтвердження на формі

При натисканні на компонент Додати пункт відкривається модальне вікно для створення нового пункту замовлення. Дана логіка доступна лише для адміністратора.

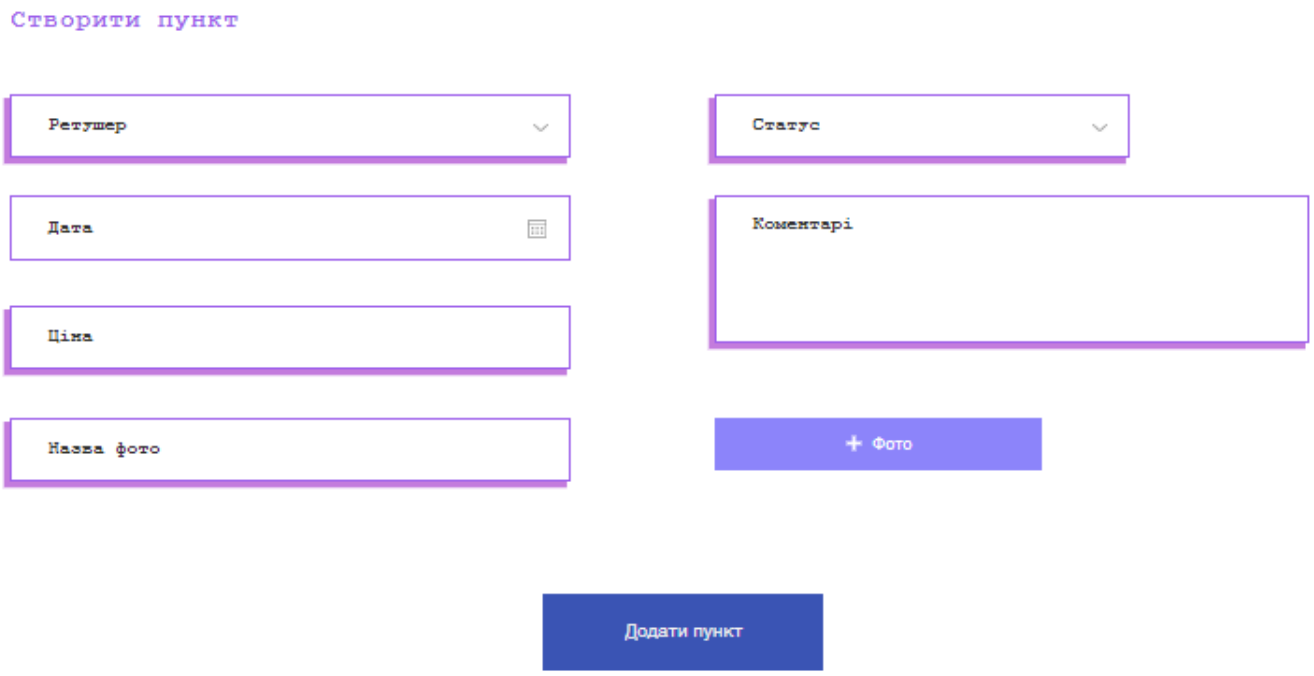

Рис. 3.11. Створення підпункту

При натисканні на кнопку додати пункт, створюється новий підпункт замовлення та відображається як новий елемент на формі замовлення. Для ролі адміністратора доступне меню – Клієнти, в якому можна переглядати список клієнтів, а також створювати нового.

Рис. 3.12. Сторінка клієнтач

При натисненні на кнопку Додати клієнта – відкривається модальне вікно з можливістю створення нового клієнта.

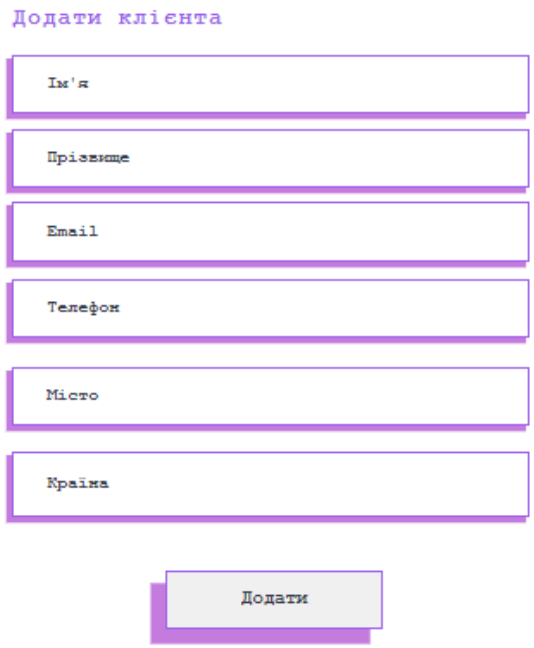

Рис. 3.13. Створення клієнта

Дана форма має основний набір полів для створення нового клієнта, детальніше опишемо в таблиці .

Таблиця 3.4

Опис елементів сторінки створення клієнта

Елемент	Компонент	Коментарі
Додати клієнта	Заголовок	Стилізований заголовок
Ім’я	Текстове поле	Поле для введення ім’я клієнта
Прізвище	Текстове поле	Поле для введення прізвища клієнта
Email	Input(type=Email)	Поле, із вказаним типом електроної пошти, передбачає валідацію
Телефон	Input	Поле, із вказаним типом номера телефону, передбачає валідацію
Місто	Текстове поле	Поле для введеня міста
Країна	Текстове поле	Поле для введеня країни
Додати	Button	Кнопка підтверження на формі

Також для адміністратора доступне меню Звітність. Представляє можливість вибору вида звітності із доступних в БД. Для генерації потрібно вибрати звіт із доступних в випадаючому списку та натиснути на кнопку генерація звіту. При виборі типу звіту, з’являється текстове поле із коротким описом даного звіту.

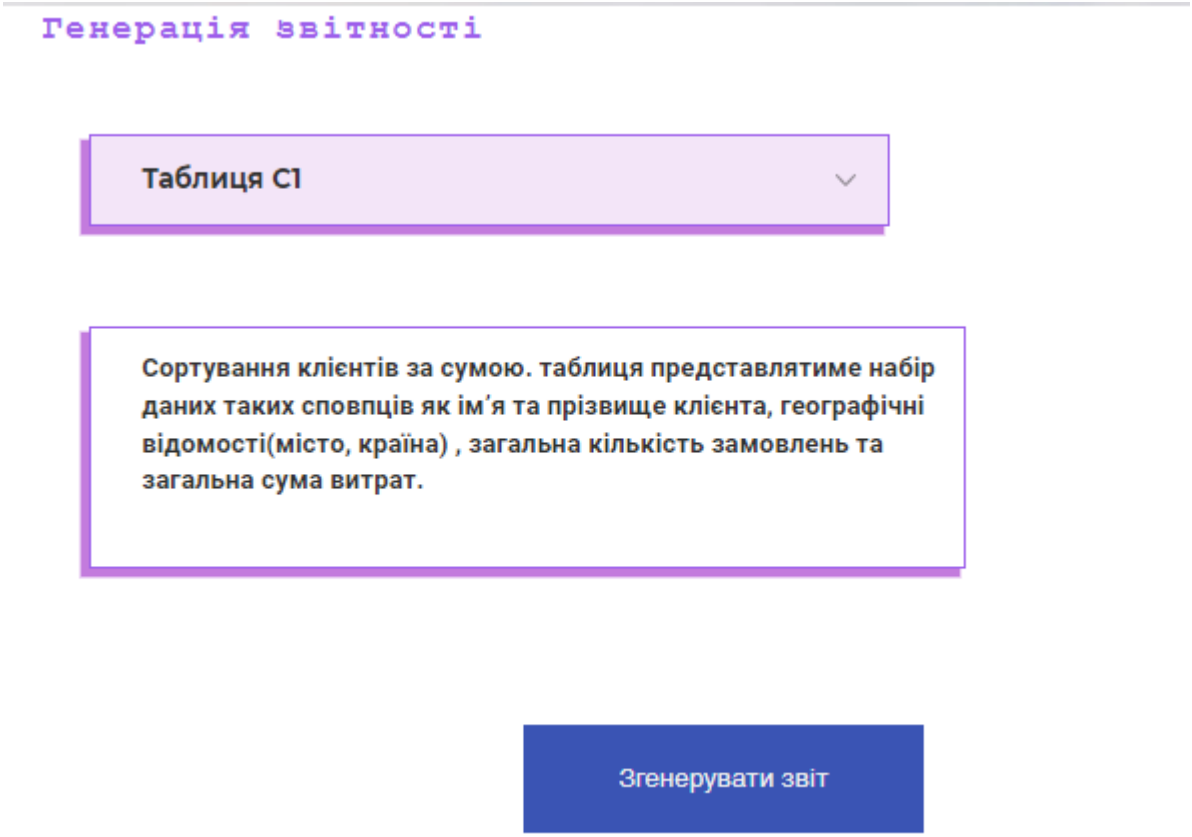

Рис. 3.14. Генерація звітності

Таблиця 3.5

Опис елментів сторінки звітності

Елемент	Компонент	Коментарі
Генерація звітності	Заголовок	Стилізований заголовок
Таблиця «Ім'я»	Випадаючий список	Включає всі види звітностей
Опис звіту	Текстове поле	Описує основні деталі звіту, з'являється в тому випадку коли користувач вибирає значення із випадаючого списку
Згенерувати звіт	Button	Кнопка підтверження на формі

3.3. Генерація звітності

SQL Server Reporting Services (SSRS) — це програмна система для створення звітів на основі сервера, створена Microsoft і використовується як рішення для компаній, яким потрібно створювати власні звіти з різноманітних джерел даних, таких як бази даних SQL та інші зовнішні джерела, що дає змогу адміністратори мають можливість ділитися звітами з користувачами на основі дозволів доступу та груп користувачів.

SSRS можна використовувати для підготовки та доставки різноманітних інтерактивних та друкованих звітів. Завдяки службам звітності SQL у поєднанні з SQL Server компанії мають уніфіковане рішення як для розробки звітів, так і для сховищ даних. SSRS може допомогти вам створювати табличні, графічні та довільні звіти з реляційних, багатовимірних і XML-джерел даних. Звіти також можуть бути опубліковані та доступні за запитом. SSRS також має вбудований інструмент планування для виконання базових розгортань звітів за допомогою електронної пошти, спільного доступу до файлів або SharePoint. SSRS можна використовувати різними способами та з іншими системами для розгортання бізнес-звітів. Служби звітності SQL можна використовувати з інструментами управління бізнес-процесами, такими як PBRS, для автоматизації звітів SSRS, переміщення даних між різними базами даних і навіть керування завданнями на основі подій.

Бізнес-звіт – це набір даних, що надає історичну інформацію, пов’язану з діяльністю компанії, виробництвом, ідеями окремих відділів, а також створює базу для майбутніх процесів прийняття рішень або фактичних даних, необхідних для організації бізнес-функцій.

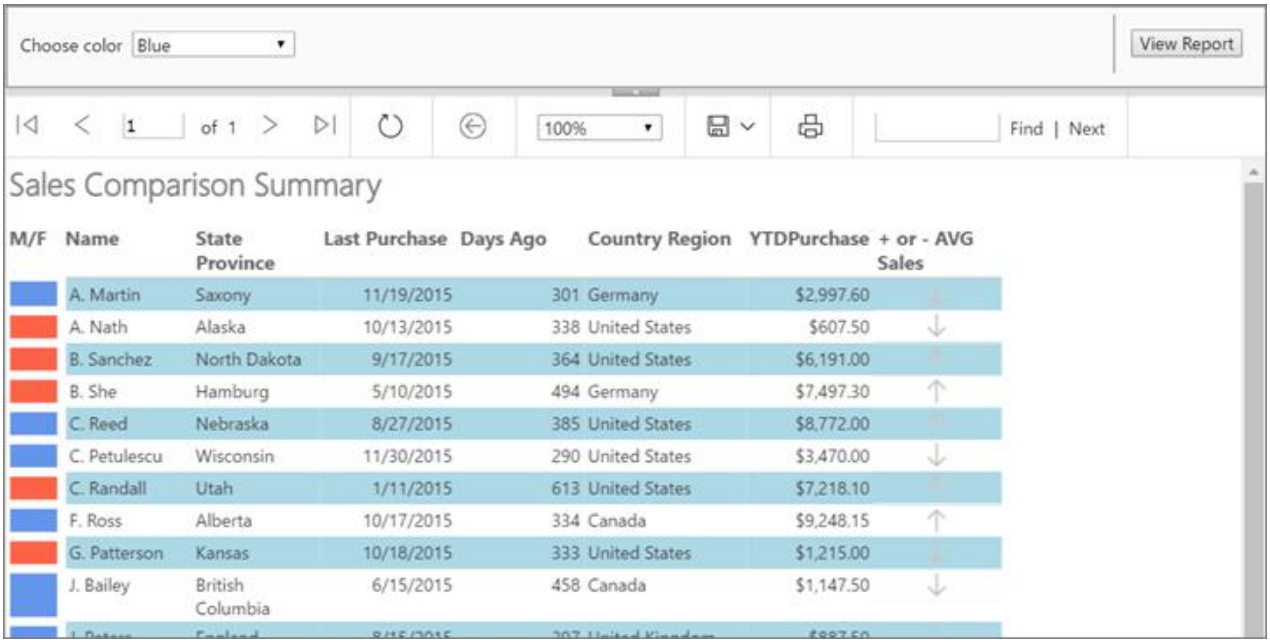

Рис. 3.15. Приклад звіту

Є сім основних переваг SSRS для звітності:

- Легкий доступ до форматів звітів: до звітів можна легко отримати доступ із Microsoft Dynamics GP. Існує також доступ одним клацанням миші з персоналізованого користувача списку часто використовуваних форматів «Мої звіти». З точки зору створення звіту, SSRS підтримує різноманітні формати, включаючи PDF або Excel. Додаткові формати можна використовувати за допомогою плагінів або зовнішнього інструменту, такого як PBRS.
- Дія деталізації: користувач може швидко отримати доступ до важливої інформації з можливістю деталізації звітів.
- Параметри діаграм. Служби звітності SQL пропонують різні варіанти макета звіту, включаючи можливості кругової діаграми, лінійної діаграми та стовпчастої діаграми. Їх можна використовувати для виділення ключової інформації та покращення бізнес-презентацій.
- Індивідуальна фільтрація: за допомогою служб звітності сервера SQL користувачі можуть фільтрувати дані звітів за допомогою динамічних параметрів.

- Гнучкі режими перегляду звітів. Служби звітів SQL дозволяють згортати звіт, щоб розгорнути розділи, зменшуючи складні звіти до керованих пропорцій.
- Підзвіти: користувачі можуть створювати підзвіти з основним звітом, а також основний звіт для одного або кількох підзвітів за допомогою набору параметрів.
- Перегляд таблиці: цей варіант макета звіту швидко представляє дані у форматі таблиці для кращого перегляду та розподілу звітів у різних підрозділах компанії.

Недоліки звітності в SSRS

- Для роботи необхідні певні знання, такі як SQL та SSRS. Без високотехнічної людини у вашій команді чи IT-відділу використання SSRS на весь свій потенціал буде значною мірою недоступним.
- Дещо обмежена можливість графічного відображення даних. Хоча корпорація Майкрософт нещодавно оновила це в останньому випуску і тепер включає SSRS в Power BI, щоб поєднати інтерактивне графічне відображення звітів Power BI зі звітами з розбитими сторінками в SSRS.
- SSRS означає, що у вас є інший сервер для обслуговування. Microsoft SSRS вимагає хостинг-сервера Windows, що ускладнює IT-підтримку.
- Обмежені можливості автоматизації або масової звітності. SSRS створювався більше як інструмент самообслуговування звітів, а не інструмент автоматизації звітів. Хоча видання Enterprise має можливості масової звітності, набір функцій невеликий у порівнянні з тим, що інші інструменти можуть виконати за допомогою SSRS.

Наприклад, навіть із додаванням Enterprise існують обмеження:

- Немає планування на основі подій
- Неможливо захистити експортовані звіти паролем
- Неможливо автоматично об’єднати робочі книги Excel і PDF-файли

У SSRS є шість типів звітів:

- Таблікс
- Матриця
- Діаграми
- Підзвіти
- Детальні звіти
- Детальні звіти

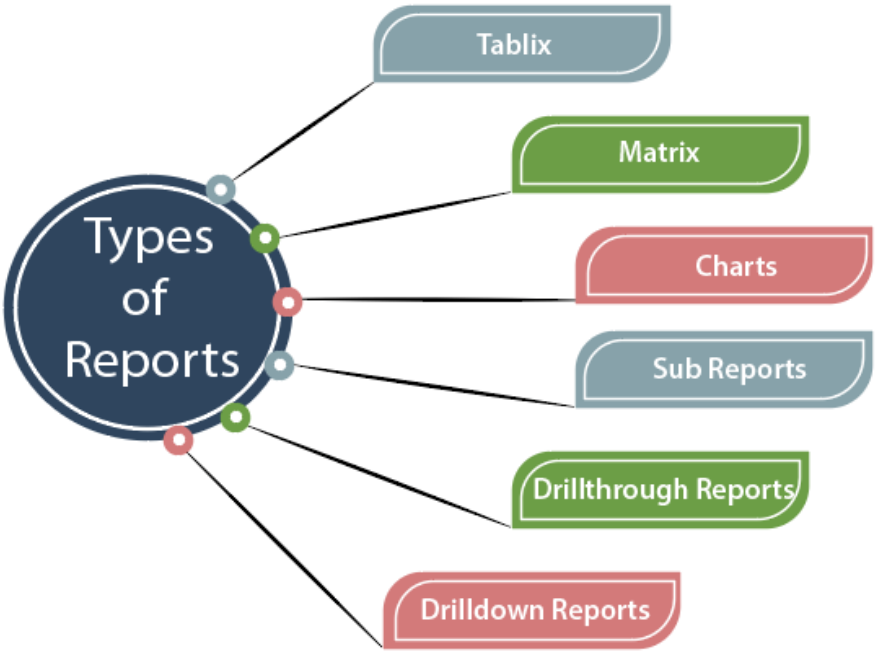

Рис. 3.16. Види звітів в SSRS

Таблікс. Формат звіту Tablix подібний до формату таблиці. Він представляє дані у вигляді таблиці. Зазвичай це одна таблиця джерела даних, яку ви представляєте.

Матриця. Він дуже схожий на звіт Tablix. Але різниця в тому, що тут ми працюємо з більш ніж одним значенням. І стовпець, і рядки стосуються деяких конкретних даних. Він використовується для групування даних за допомогою кількох полів у групі рядків і стовпців. У матричному звіті, коли дані об'єднуються під час виконання, звіт зростає по горизонталі та вертикалі. Ви навіть можете відформатувати групу рядків і стовпців, на яких ви хочете зробити акцент. Наприклад, ми хочемо знати обсяг продажів продукту, а також ми хочемо знати обсяг продажів щодо регіонів, тоді ми звертаємося до матричного звіту, а не до звіту в таблиці.

Діаграми. Діаграма складається з різних типів візуального представлення, як-от представлення стовпчастої діаграми, представлення кругової діаграми, представлення лінійної діаграми тощо. Усі ці графічні зображення представляють діаграми. Залежно від типу даних вибирається графічне зображення. Наприклад, якщо у нас є ряд даних, то стовпчаста діаграма вибирається, оскільки стовпчаста діаграма представляє графік по горизонталі, кругова діаграма представляє пропорцію в цілому, наприклад, співвідношення між чоловіками і жінками, або ви хочете представити співвідношення різних проданих товарів, тоді як лінійна діаграма складається з серії точок, з'єднаних з однією лінією, і вона оцінює дані за безперервний період часу, наприклад обсяг продажів за постійний період часу або оцінює кількість співробітників за фіксований період часу.

Підзвіти. Підзвіт – це сам звіт. Це в основному вбудовано в інший звіт. Це незалежний звіт. Залежно від ваших вимог, він може бути пов'язаний зі звітом, над яким ви зараз працюєте, або працювати самостійно. Припустимо, ви працюєте над великим звітом, тобто повідомляєте загальний обсяг продажів, а також хочете знати обсяг продажів щодо регіонів, тоді ми використовуємо підзвіт.

Детальні звіти. Детальні звіти залежать від способу обробки даних. Він показує взаємозв'язок між верхнім і нижнім рівнями. Ви можете помістити дані різними способами, щоб показати взаємозв'язок між верхнім і нижнім рівнями. Ви можете

впорядкувати дані у звіті, але вам потрібно зробити його прихованим, доки користувач не натисне на нього, щоб розкрити деталі. Ви можете відобразити дані в області даних, наприклад в таблиці або діаграмі, яка вкладена в іншу область даних, наприклад таблиці або матрицю.

Детальні звіти. Звіти деталізації також залежать від того, як обробляються дані. Це звіт, який користувач може переглянути, відкривши посилання в іншому звіті. Детальний звіт показує всі деталі елемента, який міститься в іншому звіті. Наприклад, звіт з продажу містить списки всіх замовлень на продаж, і коли ви клацнете будь-яке замовлення на продаж, ви побачите звіт, що містить деталі замовлення на продаж.

Даний модуль ретуш центра, окрім управління клієнтами, замовленнями та ретушерами надає можливість генерації специфічних бізнес звітів. Дані звіти потрібні для керівництва ретуш центру для аналізу та статистики. Таким чином можна бачити тенденцію, зміну показників.

В системі є три основні моделі – клієнти, замовлення, пункти(підпункти замовлення) , додатково можна виділити сутність решушера. До кожної із моделей визначено основні бізнес запити.

Модель клієнт. Основними критеріями пошуку є сума замовлень, географічні дані(місто, країна).

Модель ретушер. Основними критеріями є рейтинг ретушера та кількість виконаних робіт.

Модель замовлення. Основними критеріями є сума замовлення, кількість замовлень за певний термін.

Таким чином розділимо наші звіти на 3 групи – C(Clients), R(Retoucher), O(Order). Кожна група буде представляти звіти відповідної моделі. Опишемо наступним чином :

Таблиця 3.6

Таблиця характеристика типів звітів в системі

Група	Версія	Коментарі
C(Clients)	C1	Сортування клієнтів по сумі замовлень
	C2	Сортування клієнтів по групі - Місто
	C3	Сортування клієнтів по групі - Країна
R(Retoucher)	R1	Сортування за рейтингом ретушерів
	R2	Сортування за кількостю виконаних робіт
O(Orders)	O1	Загальна сума замовлень за роки
	O2	Загальна сума замовлень за місяці в році
	O3	Загальна сума замовлень за дні в місяці

Таблиця C1.

Мета – створити для бізнесу таблицю, яка надаватиме дані клієнтів відсортовані за загальною сумою їхніх витрат по спаданню.

Отримані дані – таблиця представлятиме набір даних таких сповпців як ім’я та прізвище клієнта, географічні відомості(місто, країна) , загальна кількість замовлень та загальна сума витрат.

Необхідні параметри для генерації . Присутня можливість не вказувати параметри пошуку, в такому випадку не будуть застосовуватись фільтри. Присутня можливість фільтрації набору даних по країні, містах, сумі та кількості замовлень.

Таблиця С1 - Сортування клієнтів за сумою				
Клієнт	Місто	Країна	Кількість замовлень	Сума
Петренко Іван	Дінпро	Україна	20	25024
Томілін Дмитро	Кам'янське	Україна	21	21200
Малюк Євген	Харків	Україна	28	20800
Кібиш Ольга	Сарни	Україна	21	16969
Ярош Володимир	Херсон	Україна	26	16400
Кривко Руслан	Кам'янське	Україна	30	12575
Вігор Владислав	Київ	Україна	30	11300
Канюк Анна	Львів	Україна	30	11000
Потапчук Інна	Київ	Україна	22	10560
Гонтарук Катерина	Київ	Україна	23	10020
Retouch Service - 2021 *Фільтри - не застосовані*				

Рис. 3.17. Таблиця С1

Таблиця С2.

Мета – створити для бізнесу таблицю, яка надаватиме дані клієнтів відсортовані за містами, які груповані і відсортовані по спаданню.

Отримані дані – таблиця представлятиме набір даних таких сповпців як ім’я та прізвище клієнта, географічні відомості(місто) , загальна кількість замовлень.

Необхідні параметри для генерації . Присутня можливість не вказувати параметри пошуку, в такому випадку не будуть застосовуватись фільтри. Присутня можливість кількості замовлень .

Таблиця С2 - Сортування клієнтів за містами

Клієнт	Місто	Кількість
Могильников Дмитро	Київ	47
Курач Віталій	Київ	40
Мартинюк Наталія	Київ	28
Андрущенко Лідія	Вінниця	31
Козлова Наталія	Вінниця	23
Бойко Володимир	Кам'янське	20
Альберт Давідов	Даллас	38
Ласло Пелєшкєй	Фінікс	36
Нерік Давідов	Лісабон	40
Пугач В'ячеслав	Брест	29

Retouch Service - 2021
Фільтри - не застосовані

Рис. 3.18. Таблиця С2

Таблиця С3.

Мета – створити для бізнесу таблицю, яка надаватиме дані клієнтів відсортовані за країнами, які груповані і відсортовані по спаданню.

Отримані дані – таблиця представлятиме набір даних таких сповпців як ім’я та прізвище клієнта, географічні відомості(країна) , загальна кількість замовлень.

Необхідні параметри для генерації . Присутня можливість не вказувати параметри пошуку, в такому випадку не будуть застосовуватись фільтри. Присутня можливість кількості замовлень .

Таблиця С3 - Сортування клієнтів за країнами

Клієнт	Країна	Кількість
Могильников Дмитро	Україна	38
Курач Віталій	Україна	30
Мартинюк Наталія	Україна	26
Андрущенко Лідія	Україна	20
Козлова Наталія	Україна	19
Бойко Володимир	Україна	17
Альберт Давідов	США	45
Ласло Пелєшкєй	США	36
Нерік Давідов	Португалія	28
Пугач В'ячеслав	Білорусь	26

Retouch Service - 2021
Фільтри - не застосовані

Рис. 3.19. Таблиця С3

Таблиця R1.

Мета – створити для бізнесу таблицю, яка надаватиме дані ретушерів відсортовані за допомогою фільтрації.

Отримані дані – таблиця представлятиме набір даних таких сповпців як ім’я та прізвище ретушера, його середній рейтинг, загальна кількість оцінок .

Необхідні параметри для генерації. Присутня можливість не вказувати параметри пошуку, в такому випадку не будуть застосовуватись фільтри. Присутня можливість сортування вибірки по рейтингу, кількості оцінок.

Таблиця R1 - Дані рейтинів ретушерів		
Ретушер	Середній рейтинг	Кількість оцінок
Іванова Олена	4.7	447
Коваль Олександр	4.9	421
Семінченко Дарина	4.5	554
Волкова Анна	4.8	763
Петрик Лариса	4.6	790
Резнік Людмила	4.1	531
Літвин Ірина	4.0	431
Неживий Дмитро	4.2	835
Дзюбенко Володимир	5.0	204
Плаксій Олександр	4.5	614
Retouch Service - 2021 *Фільтри - не застосовані*		

Рис. 3.20. Таблиця R1

Таблиця R2

Мета – створити для бізнесу таблицю, яка надаватиме дані ретушерів відсортовані за кількістю виконаних робіт.

Отримані дані – таблиця представлятиме набір даних таких сповпців як ім’я та прізвище ретушера, кількість виконаних робіт, дата працевлаштування.

Необхідні параметри для генерації. Присутня можливість не вказувати параметри пошуку, в такому випадку не будуть застосовуватись фільтри. Присутня можливість сортування вибірки по даті працевлаштування, кількості виконаних робіт.

Таблиця R2 - Ретушери за кількістю виконаних робіт		
Ретушер	Кількість виконаних робіт	Дата працевлаштування
Іванова Олена	180	30.04.2019
Коваль Олександр	164	20.08.2019
Семінченко Дарина	120	30.04.2021
Волкова Анна	108	30.04.2021
Петрик Лариса	103	30.04.2023
Резнік Людмила	88	30.04.2024
Літвин Ірина	68	30.04.2025
Неживий Дмитро	64	30.04.2026
Дзюбенко Володимир	56	30.04.2027
Плаксій Олександр	50	30.04.2028
Retouch Service - 2021 *Фільтри - не застосовані*		

Рис. 3.21. Таблиця R2

Таблиця O1.

Мета – створити для бізнесу таблицю, яка відображатиме кількість замовлень та загальну суму замовлень по рокам, також відображає найбільш активного клієнта за рік.

Отримані дані – таблиця представлятиме набір даних таких сповпців як рік, кількість замовлень, сума, середня сума та клієнт №1.

Необхідні параметри для генерації. Присутня можливість не вказувати параметри пошуку, в такому випадку не будуть застосовуватись фільтри. Присутня можливість сортування вибірки по даті рокам.

Таблиця О1 - Сума замовлень за рік

Рік	Кількість замовлень	Сума	Середня сума	Клієнт №1
2012	1074	607151	565.3175047	Курач Віталій
2013	1419	671398	473.1486963	Малюк Євген
2014	1357	598309	440.9056743	Томілін Дмитро
2015	1180	690161	584.8822034	Вакуленко Вікторія
2016	1277	655350	513.1949883	Слободяник Катерина
2017	1468	516233	351.6573569	Швець Олексій
2018	1160	597474	515.0637931	Кібиш Ольга
2019	1060	655363	618.2669811	Харченко Олег
2020	1252	663501	529.9528754	Джонсон Двейн
2021	1144	662830	579.3968531	Зановський Андрій

Retouch Service - 2021
Фільтри - не застосовані

Рис. 3.22. Таблиця О1

Таблиця О2.

Мета – створити для бізнесу таблицю, яка відображатиме кількість замовлень та загальну суму замовлень по місяцям, також відображає найбільш активного клієнта за місяць.

Отримані дані – таблиця представлятиме набір даних таких сповпців як рік, місяць, кількість замовлень, сума, середня сума та клієнт №1.

Необхідні параметри для генерації. Присутня можливість не вказувати параметри пошуку, в такому випадку не будуть застосовуватись фільтри. Присутня можливість сортування вибірки за рік.

Таблиця О2 - Сума замовлень за місяць					
Рік	Місяць	Кількість замовлень	Сума	Середня сума	Клієнт №1
2020	Січень	96	22639	235.8229167	Петренко Іван
	Лютий	96	72710	757.3958333	Малюк Євгеній
	Березень	109	51745	474.7247706	Кібиш Ольга
	Квітень	210	65709	312.9	Маршал Метрс
	Травень	121	38184	315.5702479	Благов Павло
	Червень	69	98675	1430.072464	Андрущенко Олена
	Липень	45	89655	1992.333333	Томілін Дмитро
	Серпень	60	35528	592.1333333	Вігор Владислав
	Вересень	103	54315	527.3300971	Максимчук Вадим
	Жовтень	68	77097	1133.779412	Вакуленко Вікторія
	Листопад	178	48837	274.3651685	Швець Олексій
	Грудень	204	28551	139.9558824	Гончар Владислав
Retouch Service - 2021 *Фільтри - не застосовані*					

Рис. 3.23. Таблиця О2

Таблиця О3.

Мета – створити для бізнесу таблицю, яка відображатиме кількість замовлень та загальну суму замовлень по дням в місяці, також відображає найбільш активного клієнта за день.

Отримані дані – таблиця представлятиме набір даних таких сповпців як рік, місяць, кількість замовлень, сума, середня сума та клієнт №1.

Необхідні параметри для генерації. Присутня можливість не вказувати параметри пошуку, в такому випадку не будуть застосовуватись фільтри. Присутня можливість сортування вибірки за рік.

Таблиця О3 - Сума замовлень за день

Місяць	День	Кількість замовлень	Сума	Середня сума	Клієнт №1
Вересень	1	9	3646	405.1111111	Кібиш Ольга
	2	6	4914	819	Курач Віталій
	3	6	5780	963.3333333	Малюк Євген
	4	5	494	98.8	Томілін Дмитро
	5	8	670	83.75	Кібиш Ольга
	6	6	2207	367.8333333	Кравченко Іван
	7	5	3352	670.4	Вакуленко Вікторія
	8	6	3633	605.5	Слободяник Катерина
	9	9	442	49.11111111	Слободяник Катерина
	10	9	2896	321.7777778	Курач Віталій
	11	10	3062	306.2	Карпов Андрій
	12	9	2869	318.7777778	Блажченко лілія
	13	9	4758	528.6666667	Кібиш Ольга
	14	8	2384	298	Петренко Іван
	15	8	4051	506.375	Лаврова Юлія
	16	9	3790	421.1111111	Томілін Дмитро
	17	8	5373	671.625	Швець Олексій
	18	10	4784	478.4	Андрущенко Олена
	19	9	1387	154.1111111	Вігор Владислав
	20	7	2236	319.4285714	Курач Віталій
	21	5	3825	765	Карпов Андрій
	22	6	5574	929	Томілін Дмитро
	23	5	617	123.4	Кібиш Ольга
	24	9	1975	219.4444444	Марченко Олексій
	25	9	2720	302.2222222	Андрущенко Лідія
	26	10	3054	305.4	Бондаренко Руслан
	27	9	504	56	Лаврова Юлія
	28	9	5545	616.1111111	Курач Віталій
	29	10	1725	172.5	Малюк Євген
	30	5	5395	1079	Кібиш Ольга

Retouch Service - 2021
Фільтри - не застосовані

Рис. 3.24. Таблиця О3

Висновок до розділу 3

В даному розділі описані основні призначення HTML, CSS, JS для веб-розробки. Надано короткі теоретичні відомості та приклади. Основної метою даного розділу є практична частина – розробка та представлення інтерфейсу системи, а також інтеграція модуля репортингу та приклади генерації звітностей.

Опис інтерфейсу складється із зображень основних сторінок та їх елементів, форм внесення даних та їх характеристик. Показані відмінності між доступним функціоналом адміністратора та ретушера.

Для репортингу надано основні теоретичні дані з використання SSRS. Наведені переваги та недоліки та види звітів. З точки зору бізнесу, детально окреслені основні види звітності в системі, їх призначення, необхідні параметри та вихідний результат. Візуально продемонстровано згенеровані основні види звітностей для моделей клієнта, замовлення та ретушера у вигляді таблиць із згенерованими даними.

ВИСНОВКИ

Під час виконання дипломної роботи був розроблений веб-сервіс ретуш центру. Модуль має два вида ролей – адміністратора та ретушера. Адміністратор має функціонал перегляду, редагування клієнтів, створення, редагування замовлень та підпунктів, можливість генерації звітності. Ретушер має можливість перегляду своїх замовлень, а також зміни статусу підпункту.

Для реалізації даного проєкту використовувались самі передові технології веб-розробки із клієнтської та серверної частин. Головною частиною завдання було реалізація підмодуля звітності, який буде легкий у майбутній підтримці та розширенні новими типами звітності. Для роботи із звітами було обрано SSRS як потужний інструмент, який задовольняє основним бізнес потребам.

Даний програмний модуль спроектований для легкої розробки нового функціоналу в майбутнього. Введено велику кількість абстракцій, максимально зменшено кількість залежностей. Для максимальної незалежності застосовано багатошарову архітектура, що довзоляє поділити систему на підмодулі, які не залежать один від одного і в майбутньому можуть легко розширюватись незалежно один від одного різними командами розробників. Використовуючи патерн Репозиторій, модуль може легко взаємодіяти з різноманітними провайдерами БД.

При необхідності розширення ролей, це буде легко реалізовано за допомогою підключеної бібліотеки .NET Identity та правильного розподілу прав на певні меню, секції на клієнті.

В подальшому, при потребі розробки десктопної або мобільної версії , це можна буде легко реалізувати, так як система має уніфікований Web API інтерфейс, що надає можливість взаємодіяти різноманітним типам клієнтів із сервером.

Даний продукт є досить універсальним для сфери ретуш замовлень, тому може бути гнучко підлаштований під різноманітні бізнес потреби. Звітність також може розширюватись індивідуально під потреби конкретного клієнта. Саме тому в залежності

від потреб та побажань клієнтів, дана система буде розширюватись новим функціоналом, оновленим дизайном та оптимізацією роботи.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ

1. Поняття ретуші. – режим доступу : https://www.imaginated.com/glossary/what-is-photo-retouching/
2. Переваги та недоліки веб-сервісів. – режим доступу : https://www.hitechwhizz.com/2021/04/5-advantages-and-disadvantages-drawbacks-benefits-of-web-application.html
3. Авторизація та автентифікація. – режим доступу : https://securityboulevard.com/2020/06/authentication-vs-authorization-defined-whats-the-difference-infographic/
4. Типи і класифікація СУБД. – режим доступу : https://whatisdbms.com/types-and-classification-of-database-management-system
5. MS SQL Server. – режим доступу : https://www.infotectraining.com/blog/what-is-microsoft-sql-server-and-what-is-it-used-for
6. ER – діаграма. – режим доступу : https://www.lucidchart.com/pages/er-diagrams
7. SSRS Reports. – режим доступу : https://go.christiansteven.com/ssrs/sql-reporting-services-ssrs-design-generate-deploy-business-reports