Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка веб-додатку салону краси «Олена»

Студентки 4 курсу, 7 групи, спеціальності 121 «Інженерія програмного забезпечення» _____ підпис студента — Богданова Каріна Сергіївна

Науковий керівник кандидат педагогічних наук, доцент _____ підпис керівника — Котенко Наталія Олексіївна

Гарант освітньої програми кандидат технічних наук, доцент _____ підпис керівника — Цензура Микола Олександрович

КИЇВ – 2021

Київський національний торговельно-економічний університет

Факультет інженерії програмного забезпечення Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь бакалавр
Спеціальність 121 Інженерія програмного забезпечення
Спеціалізація / освітня програма 121 Інженерія програмного забезпечення

Затверджую
Зав. кафедри_____
_____
"__"_____20__ р.

Завдання
на випускна кваліфікаційна робота студентові

Богданова Каріна Сергіївна
(прізвище, ім'я, по батькові)

1.Тема випускної кваліфікаційної роботи «Розробка веб-додатку салону краси «Олена»

Затверджена наказом ректора від "___" _____ 20__ р. № _____

2. Строк здачі студентом закінченої роботи _____

_____

3. Цільова установка та вихідні дані до роботи

Мета роботи - аналіз діяльності салону краси, та створення повністю робочого веб-додатку салону краси «Олена»

Об'єкт дослідження - діяльність салону краси «Олена»

_____

Предмет дослідження - технології розробки веб-додатку салону краси «Олена»

_____

4. Консультанти роботи із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

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

Вступ

Розділ 1. Теоретичні відомості про веб-додатки у сфері краси

1.1 Аналіз необхідності використання веб-додатків для бізнесу

1.2 Використання веб-додатків у сфері надання послуг

1.2.1 Аналіз підходів та моделей створення веб-додатків

1.2.2 Модель клієнт-сервер

1.3 Метод аналізу великого об’єму даних Data Mining

1.4 Висновки до Розділу 1

Розділ 2. Проектування веб-додатку салону краси

2.1 Постановка завдання

2.2 Аналіз хмарної бази даних Firebase

2.2.1 Структура і безпека Firebase

2.2.2 Аналіз підходу NoSQL до створення баз даних

2.3 Створення моделей взаємодії додатка засобами StarUML

2.4 Висновки до Розділу 2

Розділ 3. Розробка веб-додатку салону краси

3.1 Встановлення та налаштування програмного забезпечення та супутніх технологій

3.2 Опис структури сайту

3.3 Налаштування з’єднання та роботи з базою даних Firebase

3.4 Висновки до Розділу 3

Висновки

Список використаних джерел

Додатки

6. Календарний план виконання роботи

№ пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускної кваліфікаційної роботи*	*21.09.2020*	*21.09.2020*
2.	*Вступ та перелік літературних джерел*	*14.12.2020*	*14.12.2020*
3.	*Розділ 1. «Теоретичні відомості про веб-додатки у сфері краси»*	*19.02.2021*	*19.02.2021*
4.	*Розділ 2. «Проектування веб-додатку салону краси»*	*05.03.2021*	*05.03.2021*
5.	*Розділ 3. «Розробка веб-додатку салону краси»*	*10.04.2021*	*10.04.2021*
6.	*Висновки*	*24.04.2021*	*24.04.2021*
7.	*Здача випускної кваліфікаційної роботи на кафедру (перша перевірка)*	*14.05.2021*	*14.05.2021*
8.	*Підготовка автореферату та презентації доповіді*	*17.05.2021*	*17.05.2021*
9.	*Попередній захист випускної кваліфікаційної роботи*	*18.05.2021 – 21.05.2021*	*21.05.2021*
10.	*Зовнішнє рецензування випускної кваліфікаційної роботи*	*01.06.2020*	
11.	*Здача прошитого випускної кваліфікаційної роботи на кафедру*	*02.06.2020*	
12.	*Публічний захист випускної кваліфікаційної роботи*	*за розкладом роботи ЕК*	

7. Дата видачі завдання «___»_____20__ р.

8. Науковий керівник випускної кваліфікаційної роботи _____
Котенко Н.О
(прізвище, ініціали, підпис)

9. Гарант освітньої програми _____Цензура М.О._____
_____
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Богданова К.С._____
(прізвище, ініціали, підпис)

11. Відгук керівника випускної кваліфікаційної роботи

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

Науковий керівник випускної кваліфікаційної роботи ____________________
(підпис, дата)

Відмітка про попередній захист ____
(ПІБ, підпис, дата)

12. Висновок про випускну кваліфікаційну робота

Випускна кваліфікаційна робота студента Богданова К.С.
(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми Цензура М.О.
(прізвище, ініціали, підпис)

Завідувач кафедри Криворучко О.В.
(підпис, прізвище, ініціали)

«_____»_________________20 ___ р.

АНОТАЦІЯ

Дипломна робота на тему «Розробка веб-додатку салону краси «Олена» містить 44 сторінки, 23 рисунка, 4 додатка та 1 таблицю.

Метаою дослідження: є аналіз діяльності салону краси, та створення повністю робочого веб-додатку салону краси «Олена».

Об'єктом дослідження: є діяльність салону краси «Олена».

Предметом дослідження: технології розробки веб-додатку салону краси «Олена».

У першому розділі проаналізовано існуючі у веб-сайти та веб-додатки у сфері послуг, складено технічне завдання, де визначено мету, призначення та передумови створення веб-додатку, проаналізовано моделі та підходи до створення веб-додатків у бізнесі.

У другому розділі проаналізовано інформацію про салони краси, створено uml-діаграми взаємодії клієнта з веб-додатком, описано процес алгоритм вибору майстра та створено і наповнено базу даних для подальшого її використання у веб-додатку салону краси.

У третьому розділі описано процес розробки веб-додатку салону краси. Тут подано інформацію про зовнішній вигляд сайту, описано основні моменти розробки веб-додатку та наведено фрагменти коду, що використовувалися під час роботи.

ANNOTATION

Thesis on the topic "Development of a web application for beauty salon" Elena "contains 44 pages, 23 figures, 4 applications and 1 table.

The purpose of the study: is to analyze the activities of the beauty salon, and to create a fully working web application of the beauty salon "Elena".

The object of research: is the activity of the beauty salon "Olena".

Subject of research: technologies for developing a web application for beauty salon "Olena".

The first section analyzes the existing websites and web applications in the field of services, the technical task, which defines the purpose, purpose and prerequisites for creating a web application, analyzes the models and approaches to creating web applications in business.

The second section analyzes information about beauty salons, creates uml-diagrams of client interaction with the web application, describes the process of selection algorithm of the master and created and filled a database for further use in the web application of the beauty salon.

The third section describes the process of developing a web application for a beauty salon. It provides information about the appearance of the site, describes the main points of web application development and provides snippets of code used during the work.

ЗМІСТ

КНТЕУ 121 07-21.БР

Зм.	Аркуш	№ докум	Підпис	Дата	*Розробка веб-додатку салону краси «Олена»*	Стадія	Аркуш	Аркушів
Зав. кафедри		Криворучко О.В.				Зміст	2	44
Керівник		Котенко Н.О						
Гарант		Цензура М.О.						
Розроб.		Богданова К.С.			***Зміст***	Факультет інформаційних технологій, 4 курс, 7 група		

ВСТУП

Інтернет, як засіб комунікації, який не має територіальних обмежень, дозволяє обмінюватися різними видами інформації. В останні роки інтернет має величезний вплив на розвиток українських компаній, змінюючи засоби представлення компанії перед потенційними клієнтами, а також обслуговування існуючих клієнтів. Кількість людей, котрі користуються інтернетом як найважливішим засобом для отримання необхідних даних про послуги, що надають компанії, значно виросла в останні роки. Інтернет допомагає у рості бізнесу як крупним учасникам ринку, так і маленьким підприємцям.

Щоб почати надавати свої послуги в мережі інтернет компанії необхідно мати власний веб-сайт. Він може бути як візиткою компанії у мережі інтернет, для отримання клієнтами контактних даних і розуміння, в якій сфері послуг працює ця компанія, так і працювати безпосередньо с діючими клієнтами. Наявність власного веб-сайту позитивно позначається на іміджі компанії, клієнти відносяться с білою довірою. Також веб-сайт допомагає значно збільшити продаж, будучи головним інструментом для вирішення різноманітних маркетингових задач і значно зменшує навантаження на офісних працівників або адміністраторів, відповідаючи клієнтам на рутині запитання, такі як місце положення, список послуг і т.д. Також, значно зменшується документообіг, за рахунок наявності на веб-сайті власного кабінету для діючих користувачів і форм зворотного зв'язку для онлайн заявок.

Із усього описаного вище слідує, що веб-додаток є необхідним атрибутом для сучасної компанії і визначає тему вищої кваліфікаційної роботи – «Розробка веб-додатку салону краси «Олена».

Метою роботи є аналіз діяльності салону краси, та створення повністю робочого веб-додатку салону краси «Олена».

Для досягнення мети необхідно **вирішити наступні задачі**:

- проаналізувати предметну область;
- провести аналіз роботи салону краси;
- проаналізувати сучасні методології та технології для веб-розробки;
- продумати функціонал сайту та взаємодію з користувачем через веб-сайт;
- провести аналіз існуючих рішень;
- створити базу даних на базі готового рішення Firebase;
- розробити дизайн за допомогою інструменту Figma;
- розробити веб-додаток за допомогою HTML,CSS, JavaScript та супутніх бібліотек;

Об’єкт дослідження: діяльність салону краси «Олена».

Предмет дослідження: технології розробки веб-додатку салону краси «Олена».

РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ ПРО ВЕБ-ДОДАТКИ У СФЕРІ КРАСИ

1.1. Аналіз необхідності використання веб-додатків для бізнесу

В епоху інформаційних технологій ні людина, ні компанія не може обійтися без інтернету, який безповоротно увійшов в наше життя. Якщо «обличчям» людини є його сторінка в соціальних мережах, то «обличчям» компанії є її веб-сайт. Інтернет-реклама - це імідж. Від зручності користування, дизайну, структури та контексту залежить бажання потенційного клієнта скористатися послугами компанії. Веб-сайт є необхідним інструментом просування товарів і послуг в умовах сучасного бізнесу. Фахівці стверджують, що традиційні засоби масової інформації (телебачення, радіо, преса) при всіх своїх перевагах, сьогодні і в майбутньому, не здатні забезпечити належний рівень оперативності, необхідний сучасній людині. Тому все більшу популярність і затребуваність набуває Інтернет, гідністю якого є можливість коригування інформації кілька разів на день на веб-сайті.

У сучасному інформаційному суспільстві Інтернет можна сміливо назвати невід'ємною частиною бізнесу, що дозволяє компаніям здійснювати ділові комунікації з усіма цільовими групами: клієнтами, торговими посередниками, PR-сферою, постачальниками, конкурентами, діючими та потенційними співробітниками компанії. Слід зазначити, що інтернет можна використовувати не тільки як засіб доведення інформації до певної частки населення, тобто використовувати в якості реклами. Корпоративний веб-сайт пропонує своїм

					КНТЕУ 121 07-21.БР		
Зм.	*Аркуш*	*№ докум*	*Підпис*	*Дата*			
Зав. кафедри		Криворучко О.В.			*Розробка веб-додатку салону краси «Олена»*	*Стадія*	Аркуш	Аркушів
Керівник		Котенко Н.О				Р1	5	44
Гарант		Цензура М.О.				Факультет інформаційних технологій, 4 курс, 7 група		
Розроб.		Богданова К.С.			***Розділ 1***			

партнерам спеціальні розділи, закриті від широкої публіки, які дозволяють оперативно отримувати інформацію, робити замовлення и відслідковувати їх стан, не відриваючи співробітників від роботи довгими телефонними розмовами. У даний час практично будь-який споживач перед покупкою шукає інформацію про продукт(послугу), який його цікавить, перш за все в мережі інтернет, здійснюючи веб-серфінг по різним інтернет-сайтам компаній, що пропонують даний продукт(послугу). Ця обставина диктує необхідність для організацій, що працюють як в B2B, так і в B2C сфері, мати власне веб-представництво у вигляді інтернет-сайту. У всьому світі, і в Україні в тому числі, наявність працюючого WEB-сайту стає ознакою стабільної, професійної роботи компанії. Інтернет давно став не тільки інструментом для спілкування, але і плацдармом для ефективної комерційної діяльності. Практично кожна успішна компанія має в глобальній мережі своє представництво, віртуальний офіс. Сумарний оборот компаній, що ведуть торгівлю в Інтернет, досягає мільярдів доларів. Більшість розсудливих керівників українських компаній вже усвідомили важливість використання даного інструменту для побудови успішного бізнесу, але, на жаль, ще не всі вміють правильно ним користуватися, зокрема і розробляти веб-сайти. Так як в даний час істотно зросли інформаційні потреби населення, зросло значення веб-сайту як ефективного інструменту просування товарів і послуг в умовах сучасного бізнесу.

Виходячи з усього вищевикладеного, можна зробити висновок, що дана тема є актуальною та своєчасною, її вивчення може принести користь в подальшому розвитку даного напрямку.

1.2 Використання веб-додатків у сфері надання послуг

1.2.1 Аналіз підходів та моделей створення веб-додатків

Потреба в створенні інформаційних систем (ІС) може обумовлюватися як необхідністю автоматизації або модернізації існуючих інформаційних процесів, так і необхідністю докорінної реорганізації в діяльності підприємства. Перед

створенням ІС необхідно обміркувати наступні питання: по-перше, для досягнення яких саме цілей необхідна розробка системи; по-друге, до якого часу доцільно здійснити розробку; по-третє, які витрати необхідно здійснити для проектування системи.

Наразі мережа Інтернет є невід'ємною частиною інформаційних систем. Існує чимало способів комерційного підходу до Інтернету. За мережі можна рекламувати стандартні послуги, або продавати товари. Уже зараз інтернет відкриває реальні перспективи електронної комерції. На сучасному етапі розвитку електронних засобів бізнесу можна виділити два основних напрямки використання Інтернет в бізнесі: технології Інтернет для бізнесу і бізнес в Інтернет-просторі.

Перший підхід (Internet to Business) використовується, мало не з самого моменту зародження Інтернет. Будь-якій компанії необхідний інформаційний супровід своїх бізнес-процесів, а також інформаційну взаємодію в режимі On-line з зовнішнім середовищем - філіями в інших містах і країнах, клієнтами, постачальниками - надійну і, бажано, недорогу. Ті компанії, які першими стали використовувати електронну пошту та телеконференції, на деякий час отримали конкурентну перевагу. Компанії стали обзаводитися інформаційними вітринами (сайтами), а багатопрофільні компанії і корпорації - інформаційними порталами, які дуже швидко почали не тільки представляти «обличчя» компанії в бізнесі, а й стали одним з потужних інструментів управління бізнесом.

Другий підхід (Business in the Internet) заснований на розумінні того, що сучасний Інтернет є сформованим інформаційним віртуальним простором, яке доступне будь-якому користувачеві мережі в будь-який час в будь-якій точці Землі. Можливість інтерактивної взаємодії дозволяє користувачам, не виходячи з офісу або будинку, робити покупки в Інтернет-магазинах, оплачувати послуги, грати на біржі, здобувати освіту, підвищувати культурний рівень. Для компаній, що використовують Інтернет-технології, це реальна можливість «просувати» бізнес через Інтернет. У зв'язку з цим сформувалися два поняття: електронний бізнес і електронна комерція.

Електронна комерція (e-Commerce) є найважливішою складовою частиною електронного бізнесу. Це вид бізнесу, при якому взаємодія між учасниками комерційних угод відбувається за допомогою інформаційних технологій, або за допомогою Інтернету. Електронну комерцію в даний час прийнято розділяти на ряд напрямків, основними з яких вважаються: «бізнес - бізнес» (Businessto-Business - B2B), «бізнес - споживач» (Businessto-Consumer, або Business-to-Client - B2C), «споживач - бізнес» (Consumer-to-Business - C2в), «споживач - споживач» (Consumer-to-Consumer- C2C). Електронний бізнес (e-Business) означає здійснення і автоматизацію бізнес-процесів, а також підвищення ефективності діяльності підприємства за рахунок повсюдного застосування досягнень з області Web-технологій. В електронному бізнесі можна виділити чотири шари: Інтернет-інфраструктура (телекомунікаційні компанії і виробники програмного забезпечення комп'ютерного та мережевого обладнання), Інтернет-послуги (надаються Інтернет сервіс- провайдерами, що забезпечують транзакції в мережі, і власниками каналів зв'язку), інформаційні посередники (служби, консультаційні та обслуговуючі компанії, що забезпечують створення web-сторінок і управління їх контентом, пошукові машини).

Для створення салону краси, очевидним вибором є саме підход B2C. Підход B2C - це форма електронної комерції, метою якої є прямі продажі для споживача. B2C дозволяє вести прямі продажі з мінімальним числом посередників. Усунення посередників дає можливість встановлювати конкурентні ціни на місцях і навіть збільшувати їх (виключаючи відсоток посередників), що природно приведе до зростання прибутку.

До веб-систем B2C відносяться:

- web-вітрини (Front Office) торгових компаній для залучення можливих покупців до продуктів і послуг даних підприємств;
- Інтернет-магазини, які займаються тільки продажем товарів і містять необхідну інфраструктуру (Back Office) для виробництва, продажів і управління електронною торгівлею через Інтернет;

- торгові Інтернет-компанії, в яких система електронних продажів (Back Office) повністю інтегрована з усіма бізнес-процесами.

У сучасному бізнесі багато що залежить від самопрезентації компанії, її позиціонуванні на ринку послуг, що надаються і здатності шукати нових клієнтів і ринки збуту. Одним з інструментів, як іміджевих, так і маркетингових, є наявність свого сайту в мережі Інтернет. Для залучення потенційних покупців (нових клієнтів) до послуг даної компаній, автоматизації та спрощення, а отже упорядкування і прискорення, діяльності менеджерів компанії створюється web-сайт, одне з сучасних засобів передачі інформації, комунікативний засіб, і, нарешті, рекламний продукт, що дає великі можливості в області пошуку і залучення клієнтів. Ще одним значущим аргументом на користь створення сайту є те, що сайт - це сучасний і тому актуальний засіб надання інформації. Наявність власного сайту в наш час є правилом хорошого тону і запорукою успіху в розвитку будь-якого бізнесу.

1.2.2 Модель клієнт-сервер

Для роботи сайту салону краси, з можливістю запису, важливо завжди мати можливість клієнтам вносити свої дані у базу. Це відбувається при реєстрації, оформлення запису, запит на фільтр по майстрам, тощо. Простими словами, база даних - це просто набір структурованих даних. Наприклад, коли ви робите селфі: ви натискаєте кнопку і фотографуєте себе. Ваша фотографія - це дані, а галерея вашого телефону - це база даних. База даних - це місце, в якому зберігаються дані. Слово «реляційний» означає, що дані, що зберігаються в наборі даних, організовані у вигляді таблиць. Кожна таблиця пов'язана в деякому роді. Якщо програмне забезпечення не підтримує реляційну модель даних, просто назвіть її СУБД. Комп'ютери, які встановлюють і запускають програмне забезпечення СУРБД, називаються клієнтами. Коли їм потрібно отримати доступ до даних, вони підключаються до сервера СУРБД. Це система «клієнт-сервер».

Для кращого розуміння проведемо аналіз на прикладі СУБД MySQL. MySQL є одним з багатьох варіантів програмного забезпечення СУРБД. Вважається, що СУРБД і MySQL однакові через популярність MySQL. Великі додатки, такі як Facebook, Twitter, YouTube, Google і Yahoo! всі використовують MySQL для зберігання даних. Хоча спочатку він створювався для обмеженого використання, тепер він сумісний з багатьма важливими обчислювальними платформами, такими як Linux, macOS, Microsoft Windows і Ubuntu.

MySQL є однією з найпопулярніших торгових марок програмного забезпечення СУРБД, яка реалізує модель клієнт-сервер (рис.1.1). SQL повідомляє серверу, що робити з даними:

- Запит даних: запит конкретної інформації з існуючої бази даних
- Обробка даних: додавання, видалення, зміна, сортування та інші операції для зміни даних, значень або візуальних елементів
- Ідентифікація даних: визначення типів даних, наприклад, зміна числових даних в цілі числа. Це також включає визначення схеми або взаємозв'язку кожної таблиці в базі даних
- Контроль доступу до даних: забезпечення методів безпеки для захисту даних, у тому числі прийняття рішення про те, хто може переглядати або використовувати будь-яку інформацію, що зберігається в базі даних

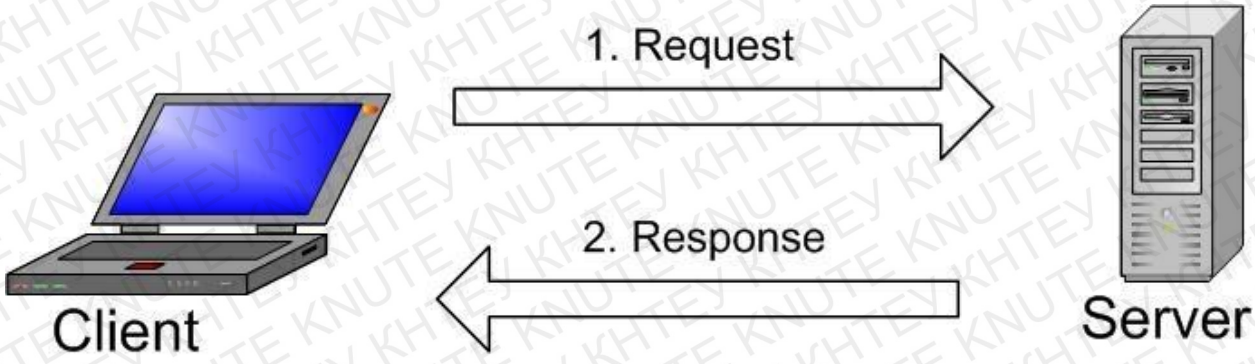

Рис.1.1 Принцип роботи моделі клієнт-сервер

Один або кілька пристроїв (клієнтів) підключаються до сервера через певну мережу. Кожен клієнт може зробити запит з графічного інтерфейсу

користувача (GUI) на своїх екранах, і сервер відобразить бажаний результат, якщо обидва кінці розуміють інструкцію. Основні процеси, що відбуваються в середовищі MySQL, однакові:

- MySQL створює базу даних для зберігання і управління даними, що визначають відносини кожної таблиці.
- Клієнти можуть робити запити, вводячи певні команди SQL на MySQL.
- Додаток сервера відповість запитаної інформацією і з'явиться на стороні клієнта.

Чим легше і зручніше графічний ористувальницький інтерфейс, тим швидше і простіше будуть виконуватися операції з управління даними. Деякими з найбільш популярних графічних інтерфейсів MySQL є MySQL WorkBench, SequelPro, DBVisualizer і Navicat DB Admin Tool.

1.3 Метод аналізу великого об’єму даних Data Mining

Зазвичай, коли говорять про серйозну аналітичну обробку, особливо якщо використовують термін Data Mining, мають на увазі, що даних величезна кількість. У загальному випадку це не так. Досить часто доводиться обробляти невеликі набори даних, і знаходити в них закономірності нітрохи не простіше, ніж в сотнях мільйонів записів. Хоча немає сумнівів, що необхідність пошуку закономірностей у великих базах даних ускладнює і без того нетривіальну задачу аналізу.

Така ситуація особливо характерна для бізнесу, пов'язаного з роздрібною торгівлею, телекомунікаціями, банками, інтернетом. В їх базах даних акумулюється величезна кількість інформації, пов'язаної з транзакціями: чеки, платежі, дзвінки, логи і т.п.

Термін Data Mining (видобуток даних, інтелектуальний аналіз даних) введений Григорієм Пятецким-Шапіро у 1989 році. З його визначенням, Data

Mining — це процес виявлення в сирих даних раніше невідомих, нетривіальних, практично корисних і доступних інтерпретації знань4, необхідних для прийняття рішень у різних сферах людської діяльності. Формально, Data Mining — це побудова моделі даних. На сьогоднішній день існує декілька підходів до побудови моделей даних, а саме:

- статистичний (Statistical Modelling): базується на теорії та зосереджується на перевірці гіпотез;
- на основі машинного навчання (Machine Learning): евристичний, концентрується на поліпшенні роботи агентів;
- обчислювальний (по суті — інтелектуальний аналіз даних): інтеграція теорії, сконцентрований на єдиному процесі аналізу даних, включає евристику даних, навчання, інтеграцію та візуалізацію результатів.

Суть і мету інтелектуального аналізу даних можна охарактеризувати таким чином: це технологія, призначена для пошуку у великих обсягах даних (Big Data) неочевидних, об'єктивних і корисних на практиці закономірностей. Неочевидних — тобто таких, що не виявляються стандартними методами обробки інформації або експертним шляхом. Об'єктивних — тобто таких, будуть повністю відповідати дійсності (на відміну від суб'єктивної експертної думки). Корисних на практиці — тобто таких, яким можна знайти практичне застосування. Нерідко Data Mining ототожнюють з виявленням знань у базах даних (Knowledge Discovery in Databases), хоча більш правильно вважати Data Mining одним із кроків цього процесу.

Інтелектуальний аналіз даних — потужний засіб у рекламі, у бізнесі, у економіці тощо. Це не просто область науки, це область життєдіяльності, що є як науковою, оскільки включає в себе наукові дисципліни, так і прикладною, оскільки спеціалісту у сфері інтелектуального аналізу даних необхідні навички програмування. Крім цього, це в якомусь сенсі культура та мистецтво, оскільки

алгоритми Data Mining вимагають постійно шукати нові шляхи вирішення проблем.

1.4 Висновки до Розділу 1

Бази даних є одним з різновидів інформаційних технологій, а також формою зберігання даних. Метою створення баз даних є побудова такої системи, що була б незалежною від програмного забезпечення та застосовуваних технічних засобів. Побудова такої системи має забезпечувати несуперечливу і цілісну інформацію. При проектуванні бази даних передбачається багатоцільове її використання. У найпростішому випадку база даних представляються у вигляді системи двовимірних таблиць.

Data Mining — це побудова моделі даних. Дані — це необроблений матеріал, що надається постачальниками даних і використовується споживачами для формування інформації на основі даних. Табличні дані складаються із записів, кожен з яких складається з фіксованого набору атрибутів.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ САЛОНУ КРАСИ

2.1 Постановка завдання

Отже, спроектована система має бути здатна зберігати в собі потрібну інформацію в структурованому вигляді та надавати швидкий доступ до неї. Використання бази даних в умовах салону краси означає, що база даних повинна мати швидкий відклик та могла зберігати велику кількість інформації.

Процес розробки складається з: проекування та самої розробки програмного продукту.

Етап проектування містить у собі:

- постановку задачі;
- вивчення особливостей та принципу роботи веб-систем у сфері надання послуг;
- аналіз поставленої задачі;
- аналіз уже існуючих рішень поставленої задачі;
- проектування програмного забезпечення;
- затвердження архітектури;

На етапі розробки мають бути виконані такі пункти:

- вибір мови програмування, фреймворків, застосунку для роботи с базою даних;
- структуризацію етапів розробки;
- розробка структури бази даних та схеми зв’язків;
- створення запитів;

- розробка клієнтського застосунку для взаємодії с базою даних;
- розробка звітів, екранних форм, запитів, макросів і програм;
- налагодження і тестування програм з ІС та навчання персоналу;

2.2 Аналіз хмарної бази даних Firebase

Firebase - це хмарна база даних, яка дозволяє користувачам зберігати і отримувати збережену інформацію, а також має зручні засоби і методи взаємодії з нею (рис.2.1). Firebase зберігає текстові дані в JSON форматі і надає зручні методи для читання, оновлення та вилучення даних. Також, Firebase може допомогти з реєстрацією та авторизацією користувачів, зберіганням сесій (авторизовані користувачі), медіафайлів до яких з легкістю надає доступ завдяки Cloud Storage.

Звичайно, що Firebase не може бути повністю безкоштовною. Але найосновніші і гаряче затребувані функції реєстрації, авторизації та зберігання тексту доступні всім після реєстрації в системі.

База даних Firebase Realtime дозволяє створювати багатофункціональні програми для спільної роботи, забезпечуючи безпечний доступ до бази даних безпосередньо з коду на стороні клієнта.

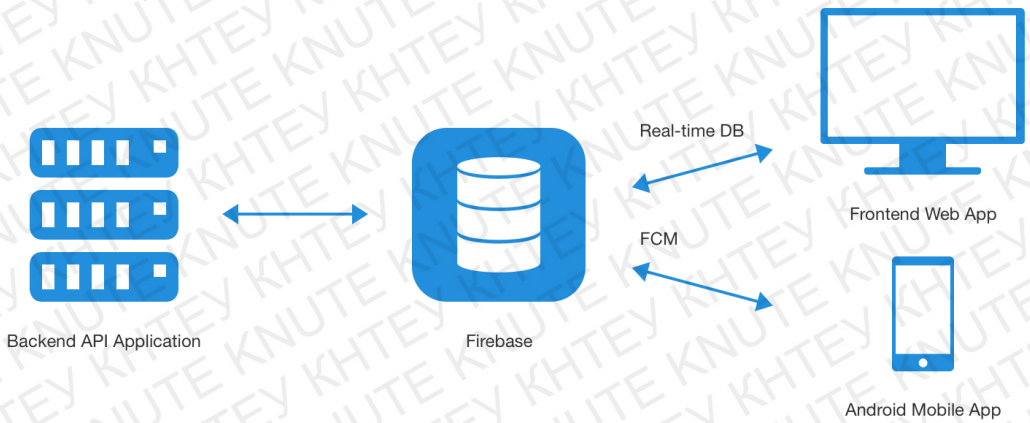

Рис.2.1 Модель взаємодії бази даних Firebase

Дані зберігаються локально, і навіть в автономному режимі події в реальному часі продовжують спрацьовувати, надаючи користувачу повноцінний відгук. Коли пристрій відновлює з'єднання, база даних реального часу синхронізує локальні зміни даних з віддаленими оновленнями, які відбулися, коли клієнт знаходився в автономному режимі, автоматично об'єднуючи будь-які зміни.

Вона надає гнучку мову правил на основі виразів, званий правилами безпеки бази даних в реальному часі Firebase, для визначення того, як ваші дані повинні бути структуровані і коли дані можуть зчитуватися або записуватися. При інтеграції з аутентифікації Firebase розробники можуть визначити, хто має доступ до яких даних і як вони можуть отримати до них доступ. База даних реального часу є базою даних NoSQL і тому має інші види оптимізації і функціональність у порівнянні з реляційною базою даних. API-інтерфейс бази даних реального часу дозволяє виконувати тільки ті операції, які можуть бути виконані швидко. Це дозволяє вам проводити обробку даних в реальному часі, обслуговуючи мільйони користувачів без шкоди для швидкості відгуку. У зв'язку з цим важливо продумати те, як користувачі повинні отримувати доступ до ваших даних, і потім відповідним чином структурувати їх.

2.2.1 Структура і безпека Firebase

Хоча база даних використовує дерево JSON, дані, що зберігаються в базі даних, можуть бути представлені у вигляді певних власних типів, які відповідають доступним типам JSON, що забезпечує можливість написати більш підтримуваний код. Наприклад, розглянемо додаток чату, який дозволяє користувачам зберігати базовий профіль і список контактів. Профіль користувача знаходиться, наприклад, в /users /$uid. У користувача alovelace може бути запис в базі даних, який виглядає приблизно так як зображено на малюнку (рис. 2.2).

```
{ "users" : { "alovelace" : { "name" : "Ada Lovelace" , "contacts" : { "ghopper" : true }, }, "ghopper" : { ... }, "eclarke" : { . .. } } }
```

2.2 Приклад запису в базі даних

База даних Firebase Realtime дозволяє вкладати дані глибиною до 32 рівнів, однак не варто думати, що так структура є за замовчуванням. Таким чином, коли ви вибираєте дані в будь-якому місці в вашій базі даних, ви також отримуєте все його дочірні вузли. Крім того, коли ви надаєте комусь доступ для читання або запису в вузлі вашої бази даних, ви також надаєте їм доступ до всіх даних в цьому вузлі. Тому на практиці краще зберігати структуру даних як можна більш плоскою.

Правила бази даних Firebase Realtime визначають, хто має права на читання і запис в вашу базу даних, як структуровані ваші дані і які індекси існують. Ці правила розташовуються на серверах Firebase і застосовуються автоматично в будь-який час. Кожен запит на читання і запис буде виконаний, тільки задовольняючи правила. За замовчуванням правила не дозволяють будь-кому доступ до вашої базі даних. Це необхідно для захисту бази даних від зловживань до тих пір, поки у вас не буде часу налаштувати свої правила або налаштувати аутентифікацію.

Правила бази даних реального часу мають JavaScript-подібний синтаксис і бувають чотирьох типів:

Таблиця 2.1

Види правил бази даних реального часу

.read	Описує, чи дозволено читання даних користувачам.
.write	Описує можливість і час запису даних.
.validate	Визначає, як буде виглядати правильно відформатоване значення, чи має воно дочірні атрибути і тип даних.
.indexOn	Визначає дочірній елемент для індексації для підтримки впорядкування і запитів.

База даних Firebase Realtime надає повний набір інструментів для управління безпекою вашого застосування. Ці інструменти полегшують аутентифікацію ваших користувачів, забезпечення прав користувачів і перевірку вхідних даних.

2.2.2 Аналіз підходу NoSQL до створення баз даних

NoSQL - це підхід до реалізації масштабування сховища (бази) інформації з гнучкою моделлю даних, що відрізняється від класичних реляційних СУБД. У нереляційних базах проблеми масштабованості і доступності, важливі для Big Data ,вирішуються за рахунок атомарности і узгодженості даних.

NoSQL-бази оптимізовані для додатків, які повинні швидко, з низькою часовою затримкою обробляти великий обсяг даних з різною структурою. Таким чином, нереляційні сховища безпосередньо орієнтовані на Big Data. Однак, ідея баз даних такого типу зародилася набагато раніше терміну «великі дані», ще в 80-і роки минулого століття , за часів перших комп'ютерів, і використовувалася для ієрархічних служб каталогів. Сучасне розуміння NoSQL-СУБД виникло на початку 2000-х років , в рамках створення паралельно розподілених систем для високомасштабованих інтернет-додатків, таких як онлайн-пошук.

Взагалі термін NoSQL означає «не тільки SQL» (Not Only SQL), характеризуючи відгалуження від традиційного підходу до проектування баз даних. Спочатку так називалася опенсорсна база даних, створена Карло Строззі, яка зберігала всі дані як ASCII-файли, а замість SQL-запитів доступу до даних використовувала шелловські скрипти. На початку 2000-х років Google побудував свою пошукову систему і додатки (GMail, Maps, Earth і інші сервіси), вирішивши проблеми масштабованості і паралельної обробки великих обсягів даних. Так були створені розподілена, файлова і координована системи, а також колончате сховище (column family store), засноване на обчислювальній моделі MapReduce. Після того, як корпорація Google опублікувала опис цих технологій, вони стали дуже популярні у розробників відкритого програмного забезпечення. В результаті цього був створений Apache Hadoop і запущені основні пов'язані з ним проекти. Наприклад, в 2007 році інший IT-гігант, Amazon.com, опублікувавши статті про свою високодоступну базу даних Amazon DynamoDB. Далі в цю гонку NoSQL-технологій для управління великими даними включилося безліч корпорацій: IBM,

Facebook, Netflix, eBay, Hulu, Yahoo! та інші IT-компаній зі своїми відкритими рішеннями (рис. 2.3).

2.3 Різноманіття NoSQL СУБД

Всі NoSQL рішення прийнято ділити на 4 типи:

1. **Ключ-значення** (Key-value) - найбільш простий варіант сховища даних, що використовує ключ для доступу до значення в рамках великої хеш-таблиці. Такі СУБД застосовуються для зберігання зображень, створення спеціалізованих файлових систем, як кешей для об'єктів, а також в масштабованих Big Data системах, включаючи ігрові та рекламні програми. Найбільш відомими представниками нереляційних СУБД типу key-value вважаються Oracle NoSQL Database, Berkeley DB, MemcacheDB, Redis, Riak, Amazon DynamoDB, які підтримують високе роз'єднання, забезпечуючи безпрецедентне горизонтальне масштабування, недосяжне при використанні інших типів БД.

2. **Документно-орієнтоване** сховище , в якому дані, представлені парами ключ-значення, стискаються в вигляді полуструктурированного документа з тегованих елементів, подібно JSON, XML, BSON і іншим подібним форматам. Така модель добре підходить для каталогів, призначених для користувача профілів і систем управління контентом, де кожен документ унікальний і змінюється з часом. Тому найчастіше документні NoSQL-СУБД використовуються в CMS-системах, видавничій справі та документальному пошуку. Найяскравіші приклади

документно-орієнтованих нереляційних баз даних - це CouchDB, Couc hbase , MongoDB, eXist, Berkeley DB XML.

3. **Колоночне сховище** , яке зберігає інформацію у вигляді розрідженої матриці, рядки і стовпці якої використовуються як ключі. У світі Big Data до колоночного сховища належать бази типу «сімейство стовпців» (Column Family). У таких системах самі значення зберігаються в стовпцях (колонках), представлених в окремих файлах. Завдяки такій моделі даних можна зберігати велику кількість атрибутів в стислому вигляді, що прискорює виконання запитів до бази, особливо операції пошуку і агрегації даних. Наявність тимчасових міток (timestamp) дозволяє використовувати такі СУБД для організації лічильників, реєстрації та обробки подій, пов'язаних з часом: системи біржової аналітики, IoT / IIoT-додатки, систему керування вмістом і т.д. Найвідомішою колоночною базою даних є Google Big Table, а також засновані на ній Apache HBase і Cassandra . Також до цього типу належать менш популярні ScyllaDB, Apache Accumulo і Hypertable.

4. **Графове сховище** – це мережева база, яка використовує вузли і ребра для відображення та зберігання даних. Оскільки ребра графа є збереженими, його обхід не вимагає додаткових обчислень (як з'єднання в SQL). При цьому для знаходження початкової вершини обходу необхідні індекси. Зазвичай графові СУБД підтримують ACID-вимоги і спеціалізовані мови запитів (Gremlin, Cypher, SPARQL, GraphQL і т.д.). Такі СУБД використовуються в задачах, орієнтованих на зв'язку: соціальні мережі, виявлення шахрайства, маршрути громадського транспорту, дорожні карти, мережеві топології. Приклади графових баз: InfoGrid, Neo4j, Amazon Neptune, OrientDB, AllegroGraph, Blazegraph, InfiniteGraph, FlockDB, Titan, ArangoDB.

У порівнянні з класичними SQL-базами, нереляційні СУБД мають **наступні переваги**:

- лінійна масштабованість - додавання нових вузлів в кластер збільшує загальну продуктивність системи;

- гнучкість, що дозволяє оперувати напівструктуровані дані, реалізуючи, у тому числі повнотекстовий пошук по базі;
- можливість працювати з різними уявленнями інформації, у тому числі без завдання схеми даних;
- висока доступність за рахунок реплікації даних і інших механізмів відмовостійкості, зокрема, шаринга;
- продуктивність за рахунок оптимізації для конкретних видів моделей даних і шаблонів доступу;
- широкі функціональні можливості - власні SQL-подібні мови запитів, RESTful-інтерфейси, API і складні типи даних, наприклад, map, list і struct, що дозволяють обробляти відразу безліч значень.

Зворотною стороною вищевказаних переваг є **наступні недоліки**:

- обмежена ємність вбудованого мови запитів
- складності в підтримці всіх ACID-вимог до транзакцій (атомарность, консистентность, ізоляція, довговічність) через те, що NoSQL-СУБД замість CAP - моделі (узгодженість, доступність, стійкість до поділу) скоріше відповідають моделі BASE (базова доступність, гнучке стан і підсумкова узгодженість)
- сильна прив'язка додатки до конкретної СУБД через специфіку внутрішнього мови запитів і гнучкою моделі даних, орієнтованої на конкретний випадок
- недолік фахівців з NoSQL-баз в порівнянні з реляційними аналогами

2.3 Створення моделей взаємодії додатка засобами StarUML

StarUML - це програмний інструмент візуального моделювання з відкритим вихідним кодом, який підтримує стандартизовану мову графічного опису UML (Unified Modeling Language) для моделювання систем і програмного забезпечення.

Уніфікована мова моделювання (UML) була розроблена з метою створення єдиної візуальні мови з багатою семантикою і розгорнутим синтаксисом для проектування і впровадження програмних систем зі складною структурою і комплексною поведінкою. Варто відзначити, що UML застосовується не тільки в розробці програм, але і в інших сферах, наприклад, в схематизації потоків виробничих процесів. UML призначена для спрощення спілкування і взаємодії участників проекта, скорочення часу на пояснення і засвоювання інформації, полегшення документування.

Діаграма прецедентів (Use-case diagram) (рис.2.4) використовує 2 основних елементи:

1) Actor (учасник) - безліч логічно пов'язаних ролей, виконуваних при взаємодії з прецедентами або сутностями (система, підсистема або клас). Учасником може бути людина, роль людини в системі або інша система, підсистема або клас, які представляють щось поза суті.
2) Use case (прецедент) - опис окремого аспекту поведінки системи з точки зору користувача. Прецедент не показує, "як" досягається певний результат, а тільки "що" саме виконується.

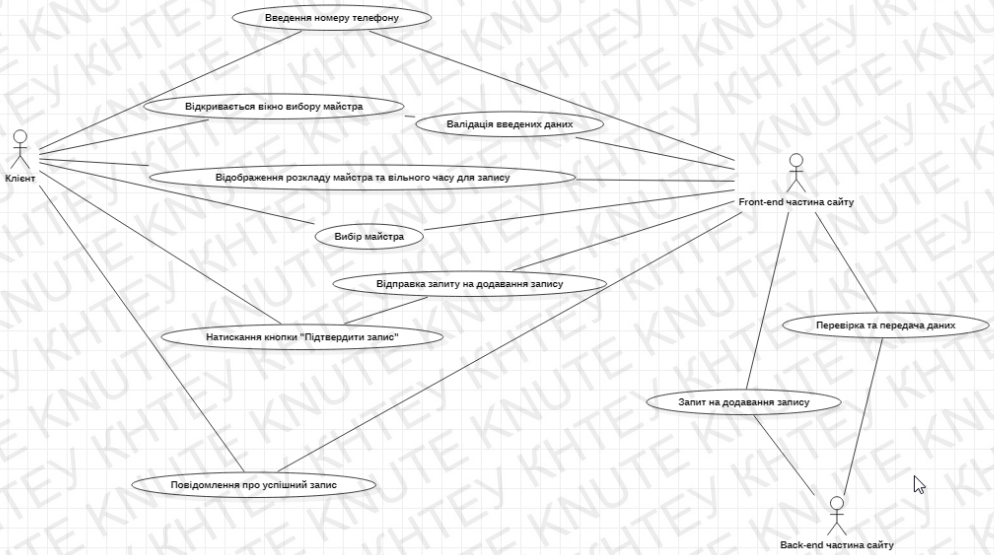

Рис. 2.4 Діаграма прецедентів

Діаграма послідовності (Sequence Diagram) (рис.2.5) використовується для уточнення діаграми прецедентів - описує поведінкові аспекти системи. Діаграма послідовності відображає взаємодію об'єктів в динаміці, в часі. При цьому інформація набуває вигляду повідомлень, а взаємодія об'єктів передбачає обмін цими повідомленнями в рамках сценарію.

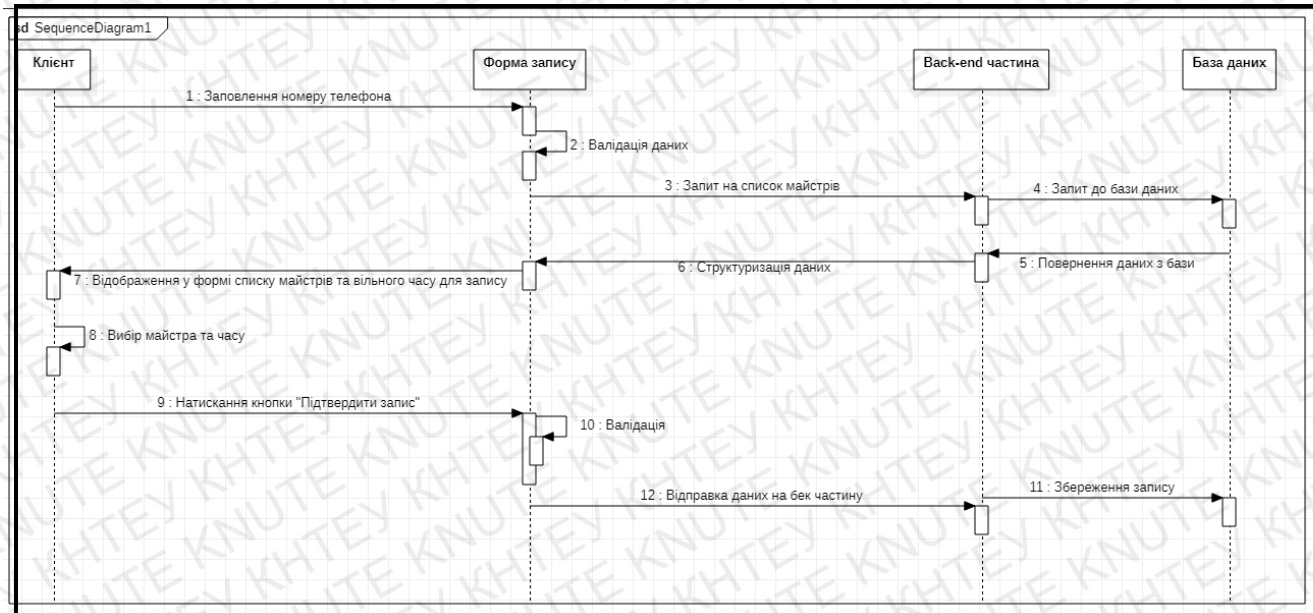

Рис. 2.5 Діаграма послідовності

Клас (class) - категорія речей, які мають загальні атрибути та операції. Сама **діаграма класів** (Class diagram) (рис.2.6) являє собою набір статичних, декларативних елементів моделі. Вона дає нам найбільш повне і розгорнуте уявлення про зв'язки в програмному коді, функціональності та інформації про окремих класах. Додатки генеруються часто саме з діаграми класів.

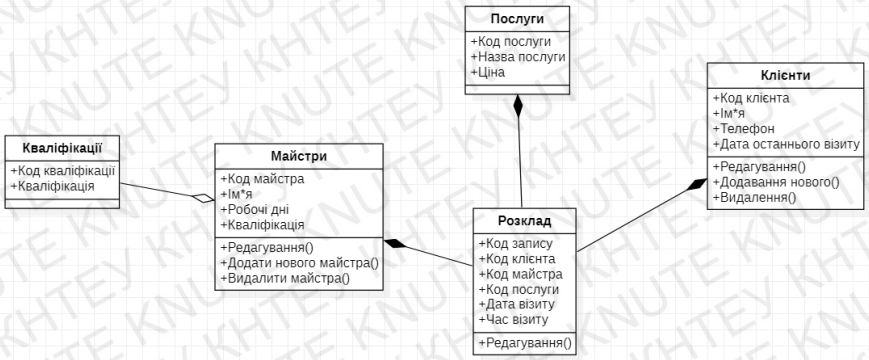

Рис. 2.6 Діаграма класів

Діаграма діяльності (діаграма активності) (рис.2.7) дозволяє моделювати послідовності бізнес-процесів або дій, реалізованих методами класів. Зазначені

послідовності можуть являти собою альтернативні галузей процесу обробки даних або галузям, які можуть виконуватися паралельно. Діаграми діяльності є аналогом блок-схеми будь-якого алгоритму. Вони, як і діаграми станів та переходів, відображаються у вигляді орієнтованого графу, вершинами якого є дії, а ребрами – переходи між діями.

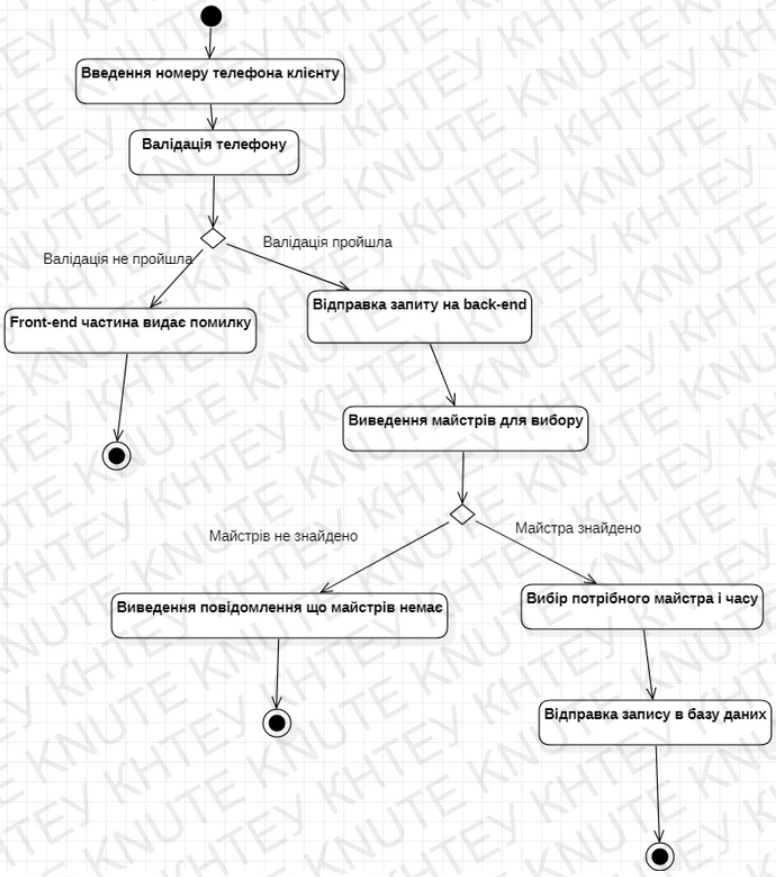

Рис. 2.7 Діаграма діяльності

Діаграма станів (рис.2.8) - це граф спеціального виду, який представляє певний автомат. Вершинами графа є можливі стани автомата, зображувані відповідними графічними символами, а дуги позначають його переходи зі стану в стан. Діаграми станів можуть бути вкладені одна в одну для більш детального представлення окремих елементів моделі.

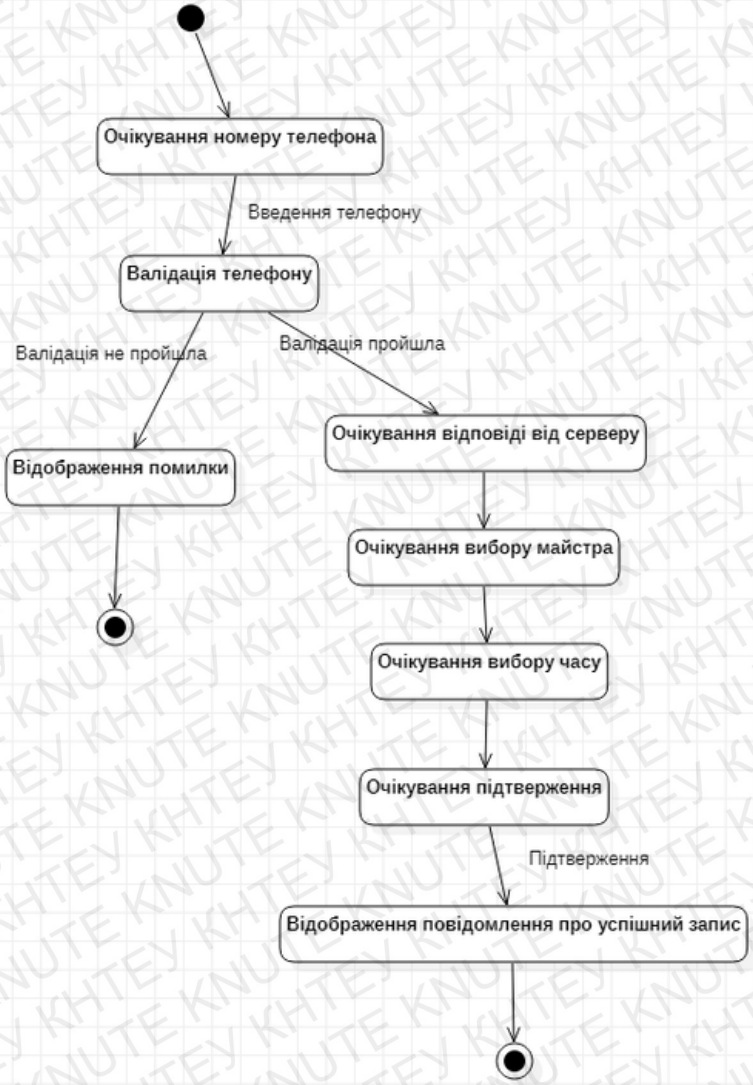

Рис. 2.8 Діаграма станів

2.4 Висновки до Розділу 2

Підводячи підсумок опису основних аспектів нереляційних СУБД, варто відзначити деяку некоректність запиту «NoSQL vs SQL» в зв'язку з різними архітектурними підходами і прикладними завданнями, на які орієнтовані ці IT-засоби. Традиційні SQL-бази відмінно справляються з обробкою строго

типизированной інформації невеликої кількості. Наприклад, локальна ERP-система або хмарна CRM. Однак, в разі обробки великого обсягу напівструктурованих і неструктурованих даних, тобто Big Data, в розподіленої системі слід вибирати з безлічі NoSQL-сховищ, з огляду на специфіку самого завдання.

Отже, для створення бази саме для веб-додатку салону краси «Олена» підійде готове рішення FireBase на основі NoSQL моделі. Таке рішення обумовлене тим, що в ній буде зберігатися не багато інформації, та помітно зросте швидкість обробки запитів на сайті

РОЗДІЛ 3. РОЗРОБКА ВЕБ-ДОДАТКУ САЛОНУ КРАСИ

3.1 Встановлення та налаштування програмного забезпечення та супутніх технологій

Розробка веб-додатку для салону краси ділиться на два етапи : розробка інтерфейсу сайту(front-end), та розробка серверної частини(back-end). Front-end і back-end - терміни в програмній інженерії, які розрізняють відповідно до принципу поділу відповідальності між представницьким рівнем і рівнем доступу до даних відповідно. Front-end - інтерфейс взаємодії між користувачем і основною програмно-апаратною частиною (back-end). Front-end і back-end можуть бути розподілені між однією або декількома системами. Для роботи над проектом будемо використовувати Visual Studio Code. Для створення структури проекту необхідно завантажити додаток, та створити в ньому папки для зберігання файлів проекту. Також дуже важливо зберігати структуру проекту. Створимо основні папки проекту.

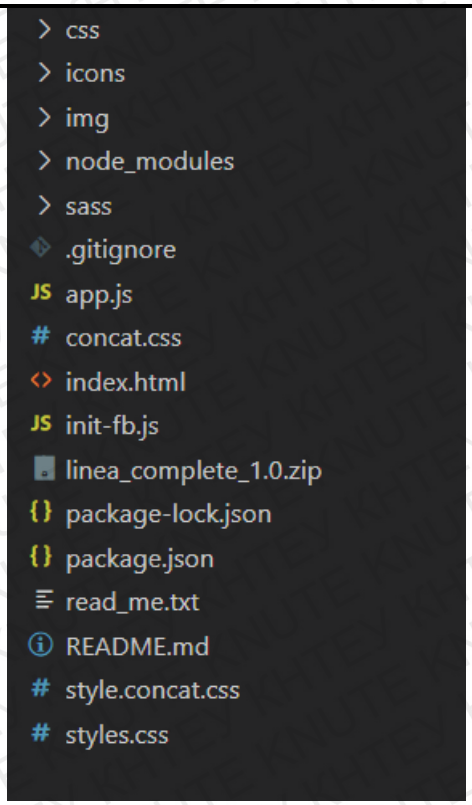

Рис.3.1 Структура проекту

На рисунку (рис.3.1) зображено основні папки, що будуть містити в собі файли для розробки сайту. Папка «node_modules» містить у собі основні модулі для роботи платформи Node.js, в неї будуть завантажені додаткові пакети для роботи. Також, проект містить такі папки:

- «css» - в ній знахотяться файли розширення .css для налаштування дизайну сайту;
- «sass» - в ній знаходяться файли, для роботи з препроцесором розширення .sass
- «img» - для зберігання картинок;

Для розробки інтерфейсу сайту будуть використовуватися HTML, CSS та JavaScript. Також, для спрощення роботи с CSS буде використовуватися фреймворк Sass. Sass - це метамова на основі CSS, призначена для збільшення рівня абстракції CSS-коду і спрощення файлів каскадних таблиць стилів.

Для серверної частини буде використовуватися Node.js. Node.js - програмна платформа, що перетворює JavaScript з вузькоспеціалізованої мови в

мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виведення через свій API (написаний на C ++), підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи виклики до них з JavaScript-коду. Також дає змогу працювати с базами даних за допомогою пакету npm.

Npm - це стандартний менеджер пакетів, автоматично встановлюється разом з Node.js. Він використовується для завантаження пакетів з хмарного сервера npm, або для завантаження пакетів на ці сервера. Усі залежні модулі проекту прописані у створеному файлі «package.json». При завантаженні додаткових модулів вони автоматично прописуються у цьому файлі.

Основні функції сайту (підключення додаткових модулів, налаштування підключення до бази даних, тощо) прописані у файлі «app.js». Також завдяки Node.js буде завантажено локальний сервер. Функція якого прописана у файлі «app.js». Завдяки командному рядку, або теміналу додатку Visual Studio Code, через файл «app.js» буде запущено локальний сервер. Запускати локально сайт необхідно через команду «npm run start» у терміналі додатку Visual Studio Code (рис. 3.2).

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

PS C:\Natours-Project> npm run start

> starter@1.0.0 start C:\Natours-Project

> starter@1.0.0 start C:\Natours-Project

> starter@1.0.0 start C:\Natours-Project
> npm-run-all --parallel serve watch:sass


> starter@1.0.0 watch:sass C:\Natours-Project
> node-sass sass/main.scss css/style.css -w


> starter@1.0.0 serve C:\Natours-Project
> live-server

Serving "C:\Natours-Project" at http://127.0.0.1:8080
Ready for changes
```

Рис. 3.2 Запуск проекту локально

Як видно на малюнку сервер запустився, тому тепер можна переходити до підключення бази даних. Для цього будемо використовувати окремий js-файл init-fb.js (рис.3.3) У цьому файлі буде лише виключно все те, що стосується підключення до бази для зберігання структури проекту.

```
JS init-fb.js > ...
1   // Your web app's Firebase configuration
2   var firebaseConfig = {
3     apiKey: "AIzaSyBck12Il6RNxw0rbjV9rTTWKJW8umbpqMs",
4     databaseURL: "https://salon-d83e1-default-rtdb.europe-west1.firebasedatabase.app",
5     authDomain: "salon-d83e1.firebaseapp.com",
6     projectId: "salon-d83e1",
7     storageBucket: "salon-d83e1.appspot.com",
8     messagingSenderId: "387311788022",
9     appId: "1:387311788022:web:7b0a086aa7b74f6c5871d7"
10  };
11  // Initialize Firebase
12  firebase.initializeApp(firebaseConfig);
```

Рис.3.3 Функція підключення до бази даних Firebase

Усі ці налаштування автоматично створюються базою даних Firebase після того як ви її створили. Розробники бази даних не рекомендують якимось чином редагувати параметри, тому залишаємо як є.

Після усіх приготувань та налаштувань можна починати роботу с сайтом та описувати структуру сайту.

3.2 Опис структури сайту

Веб-додаток салону краси «Олена» ділиться на дві частини: клієнтську і база даних. У клієнтській частині сайту представлена інформація про салон, послуги, що надаються, є можливість подачі заявки, відправляється адміністратору фірми. В базі даних Firebase є можливість додавання, редагування майстрів та записів, розташованих в клієнтській частині, видів і вартості послуг, редагування інформації про співробітників салону.

Робота з веб-додатком починається з запуску браузера і переходу на хост web-сервера. Після переходу відкривається головна сторінка сайту (рис.3.4).

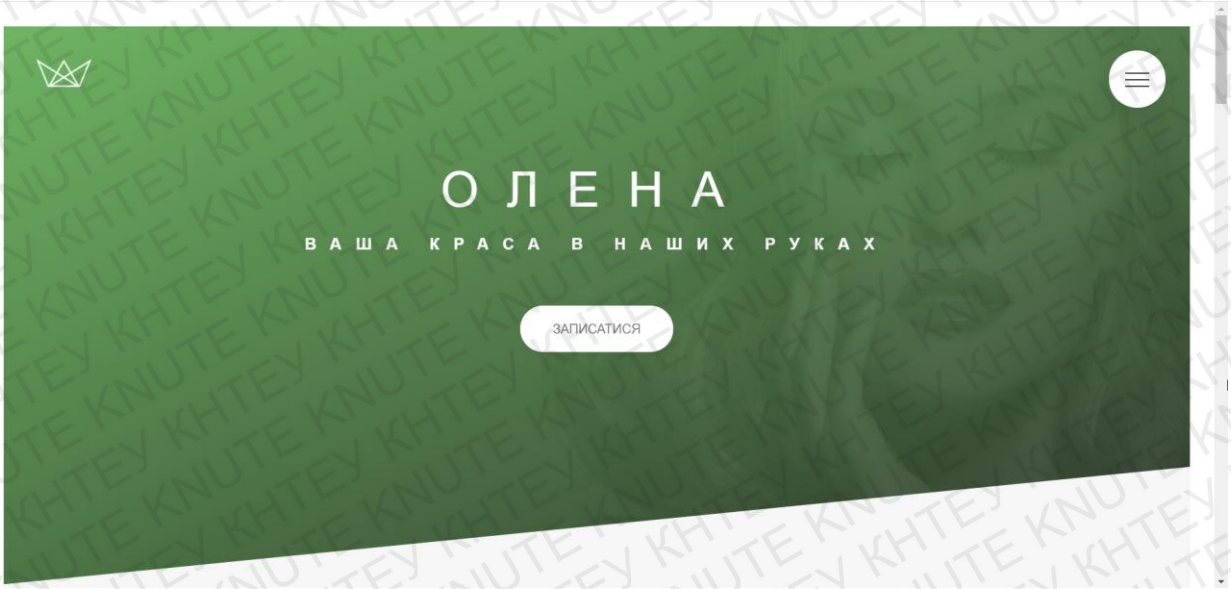

Рис. 3.4 Головна сторінка сайту

Перше що побачить користувач, це головна сторінка сайту. Головна сторінка виконана у вигляді лендіну, та розбита на декілька блоків. На головній сторінці відображається назва салону і кнопки для навігації по сайту для

відвідувачів. Так само на головній сторінці розташовується логотип салону краси «Олена», який так само є посиланням на головну сторінку сайту.

Справа в «шапці» знаходиться бургер-меню сайту - посилання для навігації по сайту (рис.3.5).

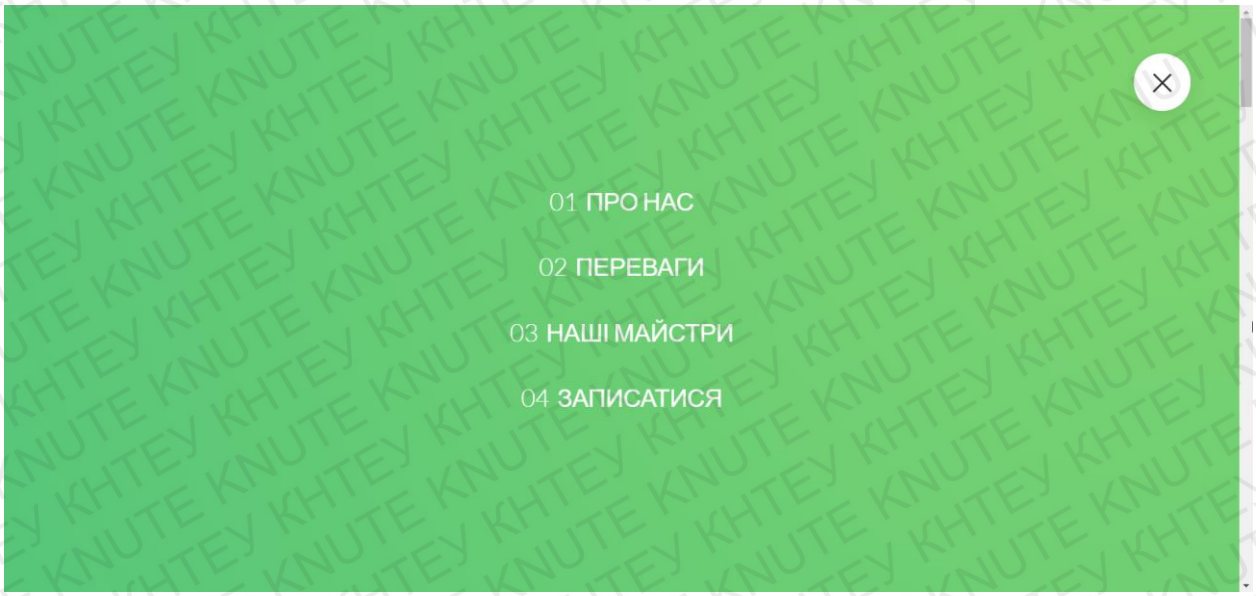

Рис. 3.5 Меню навігації сайту

При переході по кожному із посилань користувач потрапляє на блок з відповідною назві інформацією. При натисканні на логотип фірми, розташований в «шапці» сайту користувач повертається на головну сторінку сайту.

При натисканні на кнопку меню сайту «Про нас», або прокрутивши вниз до наступного блоку, користувач потрапляє на блок, в якій розміщена інформація про салон краси «Олена» (рис. 3.6).

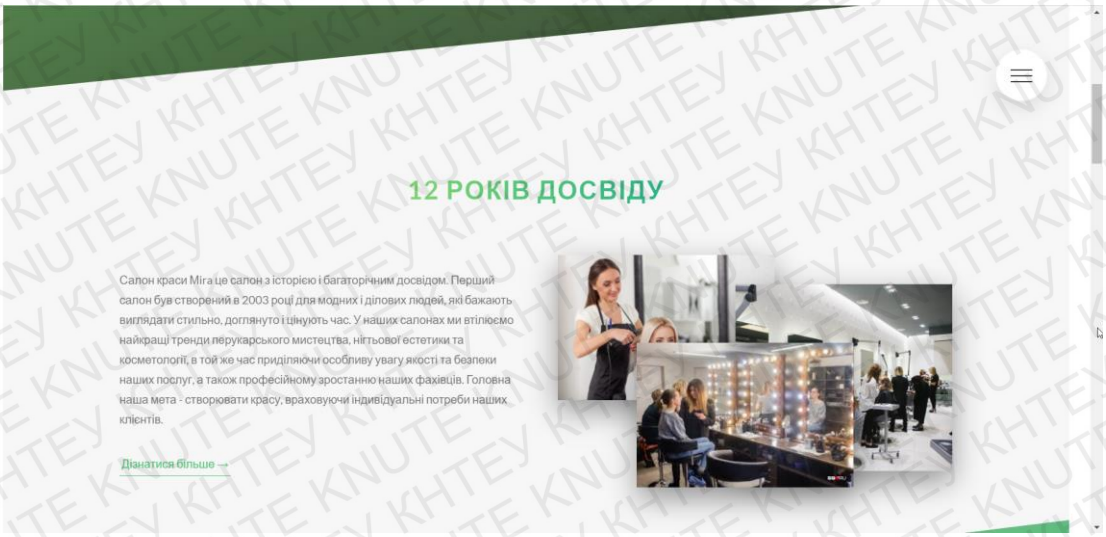

Рис. 3.6 Блок «Про нас»

Весь сайт наповнений великою кількістю анімацій. В блоці «Про нас» анімація виконана за допомогою псевдо класу ::hover наступним чином (рис. 3.7).

```
42          &:hover {
43              outline: 1.5rem solid $color-primary;
44              transform: scale(1.05) translateY(-.5rem);
45              box-shadow: 0 2.5rem 4rem rgba($color-black, .5);
46              z-index: 101; // padding: 10px;
47              // border: 2px solid $color-primary;
48          }
49      }
50      &:hover &__photo:not(:hover) {
51          transform: scale(.95);
52      }
53  }
```

Рис. 3.7 Створення анімації фотографій

Далі слідує блок з переліком послуг, які надаються в салоні, та коротким описом їх (рис.3.8). На сайті представлений не весь перелік, тут відображені лише головні категорії, по яким салон може надавати послуги. Це зроблено задля того, щоб не перевантажувати користувача великим об’ємом інформації. Для отримання більш детальної інформації користувач може звернутися за телефоном до адміністратора.

Рис.3.8 Блок переліку послуг

У цьому блоці також використовуються певні анімації (рис. 3.9).

```
310    .feature-box:hover {
311      transform: scale(1.03) translateY(-1.5rem); }
```

Рис.3.9 Анімація блоку з послугами

Наступним блоком іде декілька відгуків від клієнтів с фотографіями виконаної роботи (рис.3.10). Коли користувач зайде на сайт, скоріш за все йому захочеться переглянути відгуки інших клієнтів та подивитися на роботу майстрів.

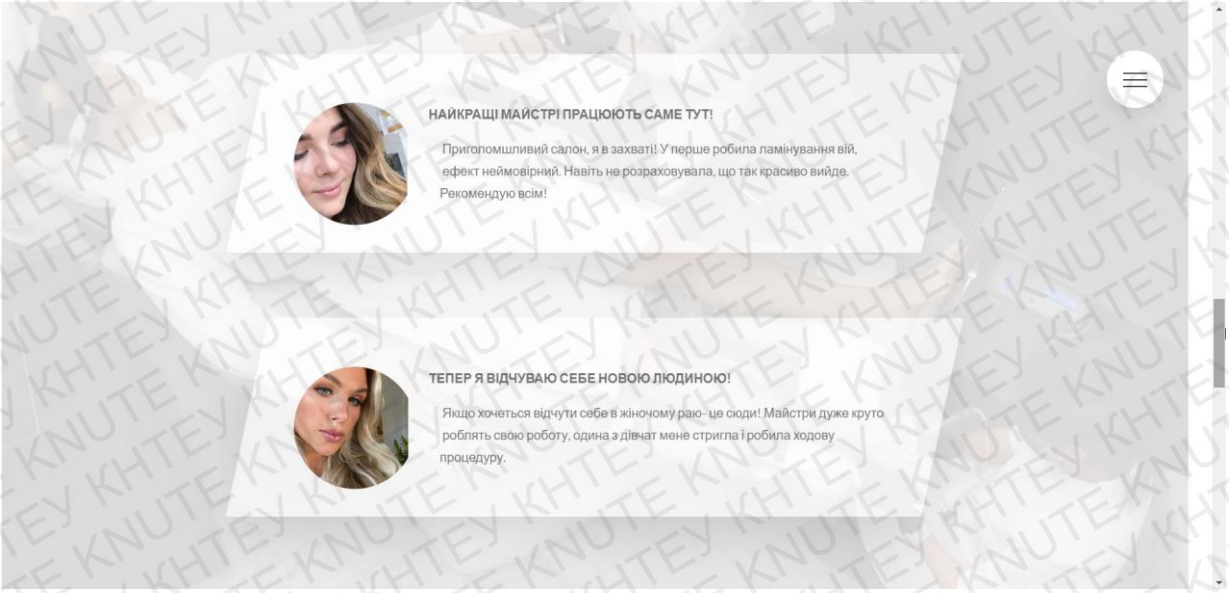

Рис. 3.10 Блок з відгуками

Наступний блок самий головний, адже саме тут відбувається з’єднання з базою даних. Усі функції для роботи з нею описані в окремих файлах app.js та init-fb.js (Додаток А). Блок з записом до майстра представлений у вигляді форми з двома dropdown-меню, одним datepicker та input для вводу електронної пошти (рис. 3.11).

Рис. 3.11 Форма запису до майстра

3.3 Налаштування з’єднання та роботи з базою даних Firebase

Як тільки клієнт відкриває сторінку сайту, вона автоматично з’єднується з базою даних Firebase. Відбувається це за допомогою функції, яку ми розглянули в минулому розділі (рис. 3.3). Далі, кожен раз при натисканні на поля у формі запису, де повинні з’явитися дані з бази даних, спрацьовує код, який відправляє запит (рис.3.12)

```
  getOrders() {
    return new Promise((resolve, reject) => {
      this.db.ref('orders').once('value', (snap) => {
        resolve(Object.values(snap.val()))
      })
    })
  }
  getMasters() {
    return new Promise((resolve, reject) => {
      this.db.ref('masters').once('value', (snap) => {
        resolve(Object.values(snap.val()))
      })
    })
  }
  getServices() {
    return new Promise((resolve, reject) => {
      this.db.ref('services').once('value', (snap) => {
        resolve(Object.values(snap.val()))
      })
    })
  }
  getWorkingHours() {
    return new Promise((resolve, reject) => {
      this.db.ref('workingHours').once('value', (snap) => {
        resolve(Object.values(snap.val()))
      })
    })
  }
}
```

Рис. 3.12 Запити в базу даних

У запитах використовуються об’єкти Promise, які використовують для відкладених і асинхронних обчислень. У запиті відправляється назва таблиці, з якої необхідно взяти дані.

В залежності від того, якого майстра користувач обрав, та які у нього робочі години, на формі буде відображатися тільки його вільні години. Після того як користувач обрав майстра, на формі з’являються radiobutton з вільним часом. Відбувається це за допомогою js коду (рис. 3.13).

```
fillWorkingHours(workingHours) {
  const caption = $('.time-caption');
  const select = $('#times')
  if (workingHours.length) {
    caption.innerText = 'Оберіть час'
    select.innerHTML = `
      ${workingHours.map(el =>
        `<div class="form__radio-group">
          <input id="time-${el.id}" type="radio" name="time" value="${el.id}" class="form__radio-input">
          <label for="time-${el.id}" class="form__radio-label">
            <span class="form__radio-button"></span>
            ${el.value}.00
          </label>
        </div>`
      )}
    `
  } else {
    caption.innerText = 'Оберіть час'
    select.innerHTML = '<p class="sorry">Нажаль в цей день у майстер зайнятий, будь ласка оберіть іншу дату</p>'
  }

}
}
```

Рис. 3.13 Функція для вибору годин

У разі якщо у майстра увесь час на обраний день зайнятий, то виводиться повідомлення «Нажаль в цей день у майстер зайнятий, будь ласка оберіть іншу дату». Рахування вільного часу у майстра відбувається наступним чином (рис. 3.14).

```
generateFreeWorkingHours() {
  const masterId = $('#masters').value
  const selectedDate = $('#datepicker').value
  const usedWorkingHours = this.orders
    .filter(order => {
      return order.date === selectedDate && +order.master_id === +masterId
    })
    .map(order => order.time)

  const workingHoursToShow = this.workingHours.filter(el => !usedWorkingHours.includes(+el.value) )
  this.ui.fillWorkingHours(workingHoursToShow)

}
```

Рис. 3.14 Розрахунок вільного часу у майстра

Нарешті, коли усі дані були обрані та заповнені клієнт натискає на кнопку «Записатися» тим самим відправляючи свої дані до бази даних, де вони зберігаються (рис. 3.15).

```
73    initSubmitBtn() {
74      $('#submit').addEventListener('click', e => {
75        e.preventDefault();
76        if (this.form.validate()) {
77          const order = this.form.prepareOrder()
78          this.db.saveNewOrder(order)
79            .then(() => {
80              this.resetItems();
81              alert('Замовлення успішно додано. Дякуємо');
82            })
83            .catch(err => {
84              alert('Вибачте, щось пішло не так. Перевірте ваше інтернет з*єднання')
85            })
86        }
87      })
88    }
```

Рис.3.15 Зберігання запису

3.4 Висновки до Розділу 3

Було розроблено веб-додаток для салону краси «Олена». Автор використала здобуті навички у володінні HTML та CSS, роботі з базою даних Firebase, роботі з мовою JavaScript та платформою Node.js. Також було використано препроцесор для CSS – SASS.

Було створено сайт салону краси, який виконаний у мінімалістичному стилі в форматі лендінгу та має блоки «Про нас», блок з категоріями послуг, блок з відгуками та форму запису. Було створено зручний інтерфейс та дизайн. Було оптимізовано запити на сервер. Було створено зручну структуру проекту. Розмітка сторінок була розділена на декілька файлів для зручної навігації в коді та підтримки. Було розроблено адаптивний дизайн. У разі необхідності змінити сторінку потрібно буде працювати лише з файлом, у якому прописана розмітка потрібного елементу.

ВИСНОВКИ

В результаті виконання роботи, було створено веб-додаток салону краси, який створений у форматі лендінгу. Сайт виконаний у стилі мінімалізму та заповнений цікавими анімаціями. Сайт доповнений формою запису до майстра, яка дозволяє відправляти запис до бази та зберігати його там.

Були спроектовані такі діаграми: USECASE, послідовності, класів, діяльності, станів. Були створені таблиці в базі даних в середовищі Firebase. На думку автора, була досягнута мета курсової роботи, а саме :

- було проаналізовано поставлену задачу та обрано оптимальне рішення;
- спроектовано сайт, завдяки якому користувач може ознайомитись з інформацією про салон краси;
- інформація була структурована для зручного доступу до неї;
- програмний код було максимально оптимізовано та структуровано для зручності в подальших допрацювань та змін;
- було правильно сформовано запити до бази даних, щоб отримувати потрібну інформацію;
- у разі зміни даних в основних таблицях, чи залежних по ключових полях, дані будуть змінюватися коректно;
- у разі доповнення бази даних новими даними, вони будуть правильно відображені на формі запису;

					КНТЕУ 121 07-21.БР		
Зм.	*Аркуш*	*№ докум*	*Підпис*	*Дата*			
Зав. кафедри		Криворучко О.В.			*Розробка веб-додатку салону краси «Олена»*	*Стадія*	Аркуш	Аркушів
Керівник		Котенко Н.О..				В	40	44
Гарант		Цензура М.О.				Факультет інформаційних технологій, 4 курс, 7 група		
Розроб.		Богданова К.С.			***Висновки***			

- сайт забезпечує зручний доступ до форми запису с будь-якого девайсу, який може підключатися до мережі інтернет. Для власників салону було спрощено додавання нової інформації до бази;

Безумовно проект має потенціал подальшої розробки. Наприклад, розробка панелі адміністратора, через яку буде можливість додавати майстрів, запису, або послуг до бази даних. Також планується покращення дизайну сайту, додавання слайдерів для перегляду робіт та сторінка з переліком майстрів. Можлива розробка реєстрації користувачів для входу до особистого кабінету, де буде зберігатися інформація про клієнта та його історію відвідувань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. David Flanagan. JavaScript: The Definitive Guide, Seventh Edition- O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.
2. Michael McMillan. Data Structures and Algorithms with JavaScript -O'Reilly Media, Inc.,2014
3. Ben Henick. HTML & CSS: The Good Parts: The Good Parts – February 25th 2010 by O'Reilly Media
4. Роббінс Д.Н. HTML5 Кишеньковий довідник: швидкий, всебічний, незамінний / Д.Н. Роббінс, 2013. – 192 с.
5. Дронов.В. HTML.5.CSS.3.и.Web 2.0.Розробка сучасних Web-додатків - БХВ-2011
6. Мейер Е.А. CSS: Повний довідник / Е.А. Мейер, Е. Уейл, 2019. – 1019 с.
7. Кирил Сухов. Node.js. Путівник по технології – М.:ДМК Прес, 2015 – 416с.:ил
8. Robin Nixon. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 (Learning PHP, MYSQL, Javascript, CSS & HTML5)
9. Офіційна документація JavaScript – режим доступу : https://developer.mozilla.org/ru/docs/Web/JavaScript
10. Офіційна документація Node.js – режим доступу : https://developer.mozilla.org/ru/docs/Web/JavaScript
11. Стандарти HTML5 – режим доступу: https://htmlweb.ru/html/doctype.php
12. Офіційна документація Firebase – режим доступу : https://firebase.google.com/docs

13. S. Franklin Planning Your Web-Site with UML [Електронний ресурс]. – URL: https://people.apache.org/~jim/NewArchitect/webrevu/2001/05_18/developers/index01.html
14. Git: Documentation [Електронний ресурс]. – URL: https://git-scm.com/doc

ДОДАТКИ

Додаток А

Лістинг файлу app.js

```
// entry point
document.addEventListener('DOMContentLoaded', run)
const $ = document.querySelector.bind(document)
const $$ = document.querySelectorAll.bind(document)

async function run() {
 const app = new App()
 app.initListeners()
 await app.getData()
 app.fillUI()
}

class App {
 constructor() {
  this.db = new DB()
  this.ui = new UI()
  this.form = new Form({ orders: [], masters: [], services: [], workingHours: []})
  this.flatpick = this.initFlatpick();
  this.orders = []
  this.masters = []
  this.services = []
  this.workingHours = []
 }
 initFlatpick() {
  const today = new Date();
```

```
    const date = `${today.getDate()}-${today.getMonth() + 1}-
${today.getFullYear()}`
    return flatpickr("#datepicker", {
     minDate: date,
     defaultDate: [date],
     locale: {
      firstDayOfWeek: 1,
     },
     dateFormat: "d-m-Y",
     disable: [
      function(date) {
       return date.getDay() === 0 || date.getDay() === 6
      }
     ]
    });
   }
   async getData() {
    this.orders = await this.db.getOrders()
    this.masters = await this.db.getMasters()
    this.services = await this.db.getServices()
    this.workingHours = await this.db.getWorkingHours()
    this.form = new Form({ order: this.orders, masters: this.masters, services:
this.services, workingHours: this.workingHours })
    console.log({ orders: this.orders, masters: this.masters, services: this.services,
workingHours: this.workingHours })
   }
   initListeners() {
    this.initDatepicker()
    this.initSubmitBtn()
    this.initMastersChange()
   }
```

```
  initMastersChange() {
    const masters = $('#masters')
    masters.addEventListener('change', e => {
      this.generateFreeWorkingHours()
    })
  }
  generateFreeWorkingHours() {
    const masterId = $('#masters').value
    const selectedDate = $('#datepicker').value
    const usedWorkingHours = this.orders
      .filter(order => {
        return order.date === selectedDate && +order.master_id === +masterId
      })
      .map(order => order.time)

    const workingHoursToShow = this.workingHours.filter(el =>
!usedWorkingHours.includes(+el.value) )
    this.ui.fillWorkingHours(workingHoursToShow)

  }
  initSubmitBtn() {
    $('#submit').addEventListener('click', e => {
      e.preventDefault();
      if (this.form.validate()) {
        const order = this.form.prepareOrder()
        this.db.saveNewOrder(order)
          .then(() => {
            this.resetItems();
            alert('Замовлення успішно додано. Дякуємо');
          })
```

```
        .catch(err => {
          alert('Вибачте, щось пішло не так. Перевірте ваше інтернет з*єднання')
        })
      }
    })
  }
  async resetItems() {
    this.orders = await this.db.getOrders()
    this.generateFreeWorkingHours()
  }
  initDatepicker() {
    this.flatpick.config.onChange.push((selectedDates, dateStr, instance) => {
      console.log({ selectedDates, dateStr, instance })
      this.selectedDate = dateStr;
      this.generateFreeWorkingHours()
    });
  }
  fillUI() {
    this.ui.fillMasters(this.masters)
    this.ui.fillServices(this.services)
    // this.ui.fillWorkingHours(this.workingHours)
  }
}
class Form {
  constructor({ orders, masters, services, workingHours }) {
    this.orders = orders;
    this.masters = masters;
    this.services = services;
    this.workingHours = workingHours
    this.db = firebase.database()
```

```
}
valid(v) {
 return !v || v !== 'none'
}
validate() {
 try {
  const service_id = $('#services').value
  const master_id = $('#masters').value
  const time = $('input[name="time"]:checked')
  const time_id = time.value;
  if (!this.valid(service_id) || !this.valid(master_id) || !this.valid(time_id)) {
   alert('Будь ласка заповніть всі поля')
   return false;
  }
  return true
 } catch(err) {
  alert('Будь ласка перевірте всі поля')
  return false
 }
}
prepareOrder() {
 const date = $('#datepicker').value
 const service_id = $('#services').value
 const master_id = $('#masters').value
 const time_id = $('input[name="time"]:checked').value;

 const service = this.services.find(el => el.id === +service_id)
 const master = this.masters.find(el => el.id === +master_id)
 const time = this.workingHours.find(el => el.id === +time_id)
 console.log(this)
```

```
    const order = {
      date,
      service_id: service_id,
      service_name: service.name,
      master_id: master.id,
      master_name: master.name,
      time: time.value
    }
    console.log(order)
    return order
  }
}
class UI {
  constuctor() {

  }
  fillMasters(masters) {
    const select = $('#masters')
    select.innerHTML += `
      ${masters.map(el => `
        <option class="option" value=${el.id} data-id=${el.id}>
          ${el.name}
        </option>
      `)}
    `
  }
  fillServices(services) {
    const select = $('#services')
    select.innerHTML += `
      ${services.map(el => `
```

```
      <option class="option" value=${el.id} data-price=${el.price} data-id=${el.id}>
        ${el.name} ${el.price} грн
      </option>
    `)}
  `
 }
 fillWorkingHours(workingHours) {
  const caption = $('.time-caption');
  const select = $('#times')
  if (workingHours.length) {
   caption.innerText = 'Оберіть час'
   select.innerHTML = `
    ${workingHours.map(el =>
     `<div class="form__radio-group">
      <input id="time-${el.id}" type="radio" name="time" value="${el.id}" class="form__radio-input">
      <label for="time-${el.id}" class="form__radio-label">
       <span class="form__radio-button"></span>
       ${el.value}.00
      </label>
     </div>`
    )}
  `
  } else {
   caption.innerText = 'Оберіть час'
   select.innerHTML = '<p class="sorry">Нажаль в цей день у майстер зайнятий, будь ласка оберіть іншу дату</p>'
  }

 }
```

```
}
class DB {
 constructor() {
  this.db = firebase.database()
  this.firebase = firebase
 }
 saveNewOrder(order) {
  const ref = this.db.ref('orders');
  var newOrderRef = ref.push();
  return newOrderRef.set(order)
 }
 getOrders() {
  return new Promise((resolve, reject) => {
   this.db.ref('orders').once('value', (snap) => {
    resolve(Object.values(snap.val()))
   })
  })
 }
 getMasters() {
  return new Promise((resolve, reject) => {
   this.db.ref('masters').once('value', (snap) => {
    resolve(Object.values(snap.val()))
   })
  })
 }
 getServices() {
  return new Promise((resolve, reject) => {
   this.db.ref('services').once('value', (snap) => {
    resolve(Object.values(snap.val()))
   })
```

```
    })
  }
  getWorkingHours() {
    return new Promise((resolve, reject) => {
      this.db.ref('workingHours').once('value', (snap) => {
        resolve(Object.values(snap.val()))
      })
    })
  }
}
```

Додаток Б

Лістинг файлу init-fb.js

```
// Your web app's Firebase configuration
var firebaseConfig = {
  apiKey: "AIzaSyBck12Il6RNxw0rbjV9rTTWKJW8umbpqMs",
  databaseURL: "https://salon-d83e1-default-rtdb.europe-
west1.firebasedatabase.app",
  authDomain: "salon-d83e1.firebaseapp.com",
  projectId: "salon-d83e1",
  storageBucket: "salon-d83e1.appspot.com",
  messagingSenderId: "387311788022",
  appId: "1:387311788022:web:7b0a086aa7b74f6c5871d7"
};
// Initialize Firebase
firebase.initializeApp(firebaseConfig);
```

Додаток В

Лістинг html коду форми запису

```
    <section class="section-book">
        <div class="row">
            <div class="book">
```

```
<div class="book__form">
  <form action="#" class="form">
    <div class=' u-margin-bottom-medium'>
      <h2 class="heading-secondary">
        Записатися
      </h2>
    </div>
    <div class="form__group">
     <select class="form__input" id="masters">
      <option value="none" hidden="">Оберіть майстра</option>
     </select>
      <label for="masters" class="form__label">Оберіть майстра</label>
    </div>
    <div class="form__group">
     <select class="form__input" id="services">
      <option value="none" hidden="">Оберіть Послугу</option>
     </select>
      <label for="services" class="form__label">Оберіть Послугу</label>
    </div>

    <div class="form__group">
      <input placeholder="Оберіть дату" class="form__input" id="datepicker" placeholder="Оберіть дату" required type="text">
      <label for="datepicker" class="form__label">Оберіть день</label>
    </div>
    <div class="form__group">
      <input id="email" type="text" class="form__input" placeholder="Email" required>
```

```
                    <label for="email" class="form__label">Email адресу</label>
                </div>
                <h3 class="time-caption heading-tertiary u-margin-bottom-small
u-margin-top-small">Оберіть майстра</h3>

                <div class="form__group u-margin-bottom-medium"
id="times">
                    <!-- <div class="form__radio-group">
                        <input name="size" type="radio" class="form__radio-input"
id="small">
                        <label for="small" class="form__radio-label">
                            <span class="form__radio-button"></span>
                            Small tour group
                        </label>
                    </div>
                    <div class="form__radio-group">
                        <input name="size" type="radio" class="form__radio-input"
id="large">
                        <label for="large" class="form__radio-label">
                            <span class="form__radio-button"></span>
                            Large tour group
                        </label>
                    </div> -->
                </div>
                <div class="form__group">
                    <button id="submit" class="btn btn--green">Забронювати
&rarr;</button>
                </div>
            </form>
        </div>
    </div>
```

```
    </div>
  </section>
```

Додаток Г

Лістинг коду основних анімацій

```
.composition {
  position: relative;
  &__photo {
    width: 55%;
    box-shadow: 0 1.5rem 4rem rgba($color-black, .4);
    position: absolute;
    z-index: 10;
    transition: all .2s;
    outline-offset: 2rem;
    @include respond(tab-port) {
      float: left;
      position: relative;
      width: 33.3333%;
      box-shadow: 0 1.5rem 3rem rgba($color-black, .2);
    }
    &--p1 {
      left: 0;
      top: -2rem;
      @include respond(tab-port) {
        top: 0;
        transform: scale(1.2);
      }
    }
    &--p2 {
      right: 0;
```

```
      top: 2rem;
      @include respond(tab-port) {
        top: -1rem;
        transform: scale(1.3);
        z-index: 100;
      }
    }
    &--p3 {
      left: 20%;
      top: 10rem;
      @include respond(tab-port) {
        top: 1rem;
        left: 0;
        transform: scale(1.1);
      }
    }
    &:hover {
      outline: 1.5rem solid $color-primary;
      transform: scale(1.05) translateY(-.5rem);
      box-shadow: 0 2.5rem 4rem rgba($color-black, .5);
      z-index: 101; // padding: 10px;
      // border: 2px solid $color-primary;
    }
  }
  &:hover &__photo:not(:hover) {
    transform: scale(.95);
  }
}
```