**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА**

**ФАКУЛЬТЕТ УПРАВЛІННЯ ФІНАНСАМИ ТА БІЗНЕСУ**

**Кафедра цифрової економіки та бізнес-аналітики**

**КУРСОВА РОБОТА**

з навчальної дисципліни

«Технології проектування та адміністрування БД і СД»

на тему:

**Інформаційна система для роботи з клієнтами туристичного оператора**

**Галузь знань:** 05 «Соціальні та поведінкові науки»

**Спеціальність:** 051 «Економіка»

**Спеціалізація:** «Інформаційні технології в бізнесі»

**Освітній ступінь:**  бакалавр

|  |  |  |
| --- | --- | --- |
| **Науковий керівник:**  к.е.н., Старух А. І.  (науковий ступінь, посада, прізвище, ініціали)  \_\_\_\_\_\_\_\_\_\_\_\_\_“\_\_\_” травня 2021 р.  (підпис) |  | **Виконавець:**  студент(ка) групи УФЕ-31 с  Андріїв Н.Р.  \_\_\_\_\_\_\_\_\_\_\_\_ “\_\_\_”травня 2021 р.  (підпис) |

**Загальна кількість балів** \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис, ПІП членів комісії)

**ЛЬВІВ 2021**

ЗМІСТ

[ВСТУП 3](#_Toc40978589)

[РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 6](#_Toc40978590)

[1.1 Постановка завдання 6](#_Toc40978591)

[1.2 Розробка моделі варіантів використання WEB-сайту 8](#_Toc40978592)

[1.3 Аналіз засобів реалізації (техніко-економічне обгрунтування вибору) 10](#_Toc40978594)

[РОЗДІЛ 2. РОЗРОБКА БАЗИ ДАНИХ 13](#_Toc40978593)

[2.1Опис моделі даних 13](#_Toc40978595)

[2.1.1Етапи створення баз даних 15](#_Toc40978596)

[2.2Нормалізація відношень 17](#_Toc40978597)

[2.3Визначення типів даних 20](#_Toc40978598)

[2.4Проектування структури бази даних 23](#_Toc40978599)

[2.4.1Перелік таблиць та полів бази даних 24](#_Toc40978600)

[2.4.2Функціональні залежності між атрибутами 26](#_Toc40978601)

2.5Реалізація SQL-скрипту 28

[РОЗДІЛ 3. РОЗРОБКАWEB-САЙТУ 30](#_Toc40978604)

[3.1 Структура web-сайту 31](#_Toc40978605)

[3.1.1Варіанти використання сайту 32](#_Toc40978606)

[3.2Макети сторінок web-сайту 33](#_Toc40978607)

[3.3 Програмування серверної частини 36](#_Toc40978608)

[3.5 Програмування клієнтської частини 39](#_Toc40978609)

[ВИСНОВКИ 42](#_Toc40978610)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 43](#_Toc40978611)

**ВСТУП**

Пандемія COVID-19 істотно впливає на стан туристичного бізнесу, проте туризм все ще залишається однією з форм раціонального використання вільного часу, проведення змістовного дозвілля, задоволення пізнавальних інтересів населення. Туризм є важливим засобом вивчення історії, залучення широкого кола людей до пізнання культурної спадщини, яку залишили нам попередні покоління.

На сучасному ринку туристичних послуг функціонує велика кількість суб’єктів, основними з яких є туристичний оператор. Основним завданням діяльності туроператора є створення туру, підкріпленого відповідною програмою обслуговування. Цей продукт і є основним туристським продуктом туроператора, що випускається на ринок і реалізованим через агентську мережу.

Сучасний стан туристського попиту і пропозиції диктує те, що туристський продукт, що формується туроператором і реалізовується турагентом, повинен бути досить диференційованим. Щоб вижити завтра, вже сьогодні потрібно знайти свій, незадоволений сегмент ринку. Це може бути сегмент національний, заснований на специфічних, національні смаки, може бути вікової, а може бути і заснованим на цілі подорожі. Всім зрозуміло, що немає універсальних готелів, тим більше - універсальних турів.

Для сфери туристичних послуг у період коронакризи споживачі послуг надають перевагу щодо: вибору коротших термінів відпочинку, переважання індивідуального пересування (автотуризм) та індивідуального розміщення, вибору оздоровчого, морського та сільського туризму.

Тому розробка веб-сайту, який надасть можливість користувачу  не виходячи з дому замовити або відразу забронювати тур в інтернеті, не варто витрачати час на поїздки по місту. Пошук і бронювання путівки займе кілька хвилин і ви гарантовано отримаєте відмінний відпочинок.

Такий програмний продукт, який буде містити в собі різноманітні функції, набагато пришвидшить та спростить процес підбору турів, і так само відразу перевіряється актуальність обраної вами пропозиції. Метою цієї курсової роботи є: розробка веб-сайту, на якому можна відстежити найцікавіші тури та найактуальніші ціни або заповнивши заявку отримати добірку від оператора в будь-яку країну із запропонованих. Пошук і бронювання путівки займе кілька хвилин і ви гарантовано отримаєте відмінний відпочинок.

Для *досягнення мети* курсової роботи, необхідно вирішити такізавдання:

1. коротко охарактеризувати інформаційну систему для туристичного оператора;
2. проаналізувати предметну область;
3. розробити архітектуру програмної системи та спроектувати структуру

бази даних;

1. зробити вибір мов програмування та технологій для програмної

реалізації описаного продукту;

1. розробити веб-сайт для туристичного оператора для можливості пошуку і бронювання путівки.

*Об’єктом дослідження*виступає база даних, як основний елемент інформаційної системи туристичного оператора.

*Предмет досліджень* – застосування технологій створення веб-орієнтованого програмного забезпечення для розробки веб-сайту.

*Використане програмне забезпечення та структура роботи.*

В процесі розробки використовуватимуться наступні мови програмування

та технології:

Node.js – платформа з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою Javascript. Node.js надає можливість виконувати JavaScript-скрипти на сервері та відправляти користувачеві результат їхнього виконання.

HTML– являє собою мову тегів, якою пишуться гіпертекстові документи для мережіІнтернет.

CSS– спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінкинаписанімовамирозміткиданих.

MySQL – одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

Курсова складається з трьох розділів:

1. «Аналіз предметної області»
2. «Розробка бази даних»
3. «Розробка веб-додатку»

Курсова робота містить також висновки, список використаних джерел.

Загальний обсяг роботи – 44 сторінки

РОЗДІЛ І. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

* 1. Постановка завдання

Основні завдання туроператора – пошук туру, спрямованого на задоволення інтересів подорожуючих, які визначають вид туру.

Аналізуючи діяльність туроператора і враховуючи переваги, які дає перехід на web-обслуговування споживачів, можна зробити висновок, що звернення до можливостей Інтернету дозволяє понизити виробничі витрати, створити сприятливі умови і зручності для покупця, прискорити оформлення турів і резервування місць відпочинку. При цьому вивільняються додаткові засоби, які можна використовувати для введення гнучкішої системи знижок.

Веб-сайт стає ланкою технологічного ланцюжка створення додаткової туристичної цінності для споживача – від пошуку інформації до резервування подорожей, що змінює поведінку споживачів.

*Можна виділити широкий спектр бізнес-операцій, які дозволить виконати веб-сайт туроператора:* обмін інформацією; встановлення контактів; продаж турів; електронну оплату;

Онлайновий туризм стає одним з найефективніших вертикальних ринків, де 64% операцій здійснюються через інтернет, тоді як в решті галузей цей показник складаєвід 30 до 40%.

Головним фактором ефективної роботи туроператорських компаній є кількість і якість продаж. Вплинути на продажі, покращити їх якість, максимально підвищити ефективність роботи можливо шляхом застосування систем бронювання і резервування в діяльності підприємств туристичного бізнесу.

Збільшення числа авіакомпаній, транспортних засобів, а також зростання обсягів авіаперевезень привели до необхідності створення і використання веб-сайтів, як один з видів систем бронювання, які стали основним інструментом для резервування турів.

Інформаційні системи туроператора дозволяють суттєво покращити якість обслуговування клієнтів за рахунок скорочення часу на оформлення квитків, забезпечити їх бронювання, підвищити якість і ефективність роботи.

Більшість учасників туристичного ринку надають переваги технології он-лайн бронювання, яка ґрунтується на моментальному відображенні реальної інформації про наявність місць за поточними тарифами з можливістю їхнього резервування.

Проте практиці ці дві функції часто розмежовуються. Адже моментальне бронювання можливо тільки після надання фінансових гарантій оплати.  
Переваги інформаційної системи туроператора для різних учасників процесу:  
- *для клієнтів:*

1. моментальне отримання матеріального підтвердження бронювання;  
   швидке оформлення путівки;
2. можливість одержати повну інформацію про свою заявку в будь-який момент часу за телефоном або через Інтернет;

- *для туроператора:*

1. Скорочення витрат: зняття необхідності заводити заявки агентів у систему;
2. Всі тури, групові або індивідуальні, йдуть через систему, що дозволяє отримати реальну картину продажів турів;
3. Повну автоматизацію турів;
4. Максимальну автоматизацію індивідуальних турів;
5. Зняття з менеджерів необхідності витрачати час на повідомлення клієнтів про стан їхніх заявок .
6. Збільшення продажів: вивільнення менеджерів напрямків для роботи з індивідуальними турами;
7. Аналітика: отримати актуальну інформацію із продажів на будь-який момент часу; повну прозорість продажів, можливість простежити будь-яку заявку в системі (як оброблену, так і відхилену).
8. Поліпшення якості обслуговування клієнтів: більш швидку обробку заявок клієнтів; наявність додаткових сервісів для клієнтів (різного роду повідомлення, одержання доступу до інформації про свої заявки в будь-який момент часу).

Бронювання агентом послуг у туроператора є центральним у загальній технології обслуговування клієнта. Адже від того, чи зуміє туроператор підтвердити заявку агента в присутності клієнта, залежить все подальше відношення клієнта до агента і послуг, які він надає.

1.2. Розробка моделі варіантів використання WEB-сайту

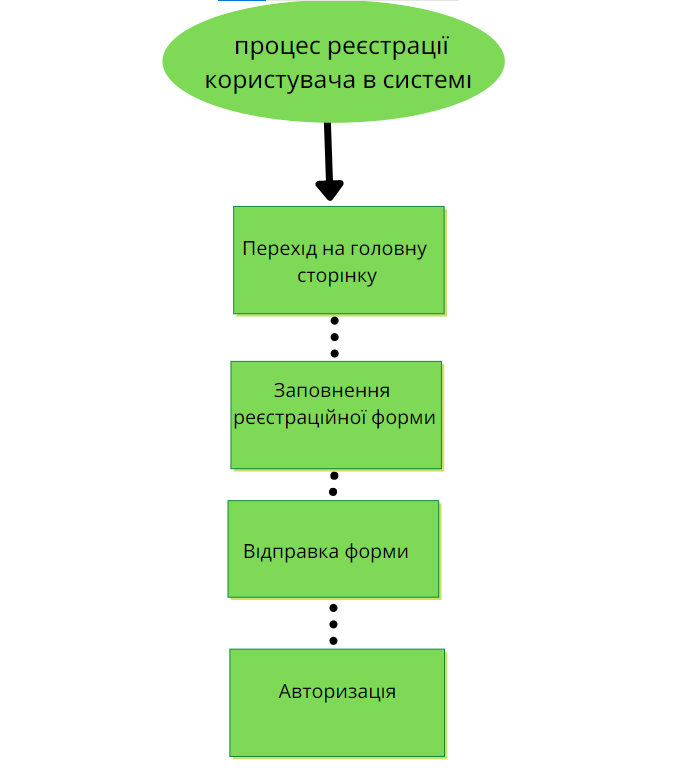
Для представлення роботи веб-сайту туристичного оператора виділено наступні бізнес-процеси (рис. 1.2.2):

* реєстрація користувача в системі;
* робота користувача з web-сайтом;
* резервування турів.

За управління даними в системі відповідатиме оператор, а реєструватися в системі і резервувати тури буде користувач (рис. 1.2.1).



**Рис. 1.2.1. Модель використання web-сайту користувачем**



**Рис. 1.2.2.Модель функцій процесу реєстрації користувача**

Аналіз переваг та недоліків web-сайту туристичного оператора відображено в таблиці 1.

*Таблиця 1*

|  |  |  |
| --- | --- | --- |
| **Функція** | **Недолік** | **Перевага** |
| **Реєстрація в системі** | Туроператор може ввести некоректні дані клієнта | Можна вибрати тур не виходячи з дому |
| **Резервування туру в системі** | Потрібно стояти в черзі для отримання бронювання | моментальне отримання матеріального підтвердження бронювання |
| **Відстежування блогу** | Довгий пошук інформації про тур в різних джерелах | Готова інформація про тур на сайті без використання додаткових джерел |

1.3.Аналіз засобів реалізації (техніко-економічне обґрунтування вибору)

Незаперечна перевага XAMPP в тому, що ця програма працює на будь-яких дистрибутивахLinux, Windows, Solaris, MacOS, є сервери для різних систем.У базову збірку вбудовані всі модулі, які можуть знадобитися, дуже проста інсталяція, може функціонувати в соціальній мережі,захист компонентів вже налаштована.Якщо правильно налаштувати, буде працювати як повноцінний інтернет-сервер.У головному меню програми доступні основні функції. Там можна подивитися статус сервера і встановлені бібліотеки.

Node.js - це програмна платформа, яка робить JavaScript мовою загального призначення. Її також називають середовищем виконання JS. Вона вміє зв'язуватися з зовнішніми бібліотеками, викликати команди з коду і виконувати роль веб-сервера. Якщо пояснювати простіше, цей інструмент додає до повністю фронтендовому мови бекендовую частина, дозволяючи створювати з його допомогою не тільки веб-сайти, а й повноцінні програми, без задіяння браузера.

Як і JavaScript, Node запускається в V8. Це середовище виконання, яка переводить написане веб-розробником в машинний код - набір прямих інструкцій для комп'ютера, які не потребують інтерпретації. Таким чином низькорівневий код стає швидким і зручним для розпізнавання машиною.

Плюси Nodejs можна перераховувати досить довго, але найбільш значимий – неблокуючамодель введення-виведення. Ця система керована подіями і працює асинхронно, вибудовуючи чергу за пріоритетністю. Коли до сервера одночасно підключаються тисячі людей, їй легше впоратися з навантаженням, так як немає потреби створювати окремий потік для кожного підключення. Грамотний розподіл ресурсів допомагає витримати більшу кількість з'єднань.

Також можливість застосовувати одну мову на клієнті і сервері.Якщо програміст прокачався в JavaScript, йому буде легше вивчити "надбудову", ніж кардинально відрізняється технологію.

Загальний код. Коду, використовуваного і на клієнті, і на стороні сервера, трохи, але він присутній. Головне - розуміти, що часто об'єкти з однаковими назвами можуть виконувати абсолютно різні функції в браузері і на бекенде.  
Розроблявся спеціально для web. Вільно взаємодіє з найбільш популярними базами даних, допомагає отримати низькорівневий доступ (http, udp, https, tcp).

І звичайно швидкість.Створення робочого, справляються з навантаженням, прототипу не відніме багато часу. Перший етап, коли програміст формує кістяк майбутнього продукту, проходить дуже швидко.  Велике і бадьорий ком'юніті. Так як відкритий код, веб-розробники можуть писати різні модулі та пакети і ділитися ними. Найчастіше, модулі добре комбінуються.

**MySQL** - це одна з найпопулярніших і найпоширеніших СУБД (система управління базами даних) в інтернеті. Вона не призначена для роботи з великими обсягами інформації, але її застосування ідеально для інтернет сайтів, як невеликих, так і досить великих.

**MySQL** відрізняться хорошою швидкістю роботи, надійністю, гнучкістю. Робота з нею, як правило, не викликає великих труднощів. Підтримка сервера MySQL автоматично включається в поставку PHP.

Важливим фактором є її безкоштовність. MySQL поширюється на умовах загальної ліцензії GNU (GPL, GNU PublicLicense).

Раніше для довготривалого зберігання інформації ми працювали з файлами: поміщали в них деяку кількість рядків, а потім витягували їх для подальшої роботи. Завдання тривалого зберігання інформації дуже часто зустрічається в програмуванні Web-додатків: підрахунок відвідувачів в лічильнику, зберігання повідомлень у форумі, віддалене управління змістом інформації на сайті і т.д.

Тим часом, професійні прийоми роботи з файлами дуже трудомісткі: необхідно піклується про приміщенні в них інформації, про її сортування, добуванні, при цьому не потрібно забувати, що всі ці дії будуть відбуватися на сервері хост-провайдера, де з дуже великою ймовірністю варто один з варіантів Unix - отже, потрібно так само піклується про права доступу до файлів і їх розміщенні. При цьому обсяг коду значно зростає, і зробити помилку в програмі дуже просто.

Всі ці проблеми вирішує використання бази даних. Бази даних самі дбають про безпеку інформації та її сортування і дозволяють отримувати і розміщувати інформацію за допомогою одного рядка. Код з використанням бази даних виходить більш компактним, і налагоджувати його набагато легше. Крім того, не потрібно забувати і про швидкість - вибірка інформації з бази даних відбувається значно швидше, ніж з файлів.

РОЗДІЛ 2. РОЗРОБКА БАЗИ ДАНИХ

Однією з особливостей функціонування туристичної галузі є інформаційна насиченість процесу виробництва, формування та реалізації такого специфічного товару як туристичний продукт. У зв’язку з цим для успішної діяльності будь-якого туристичного підприємства в сучасних умовах зростає роль постійного потоку правдивої та своєчасної інформації. Збір, обробка, застосування та передача інформації мають особливе значення в щоденній роботі підприємства – посередника в туризмі.

На сьогоднішній час до інформації висовуються високі вимоги: надійність, точність, оперативність, адекватність, багатофункціональність, висока швидкість обробки та передачі, гнучкість та інші вимоги, а сукупність цих вимог реалізується в рамках інформаційних систем, що базуються на комп’ютерних інформаційних технологіях.

## **Опис моделі даних**

База даних – це сукупність логічно пов'язаних даних, організованих на машинному носії засобами системи управління даними (СУБД). Система управління базою даних – це комплекс програм, за допомогою якого користувачі можуть визначати, створювати і підтримувати базу даних, а також здійснювати до неї контрольований доступ.

База даних включає дані, які відображають певну логічну модель взаємопов'язаних інформаційних об'єктів, що представляють конкретну предметну область. База даних організується відповідно до моделі і структури даних, які підтримує СУБД.

Щоб користувач легко міг знаходити потрібну інформацію, во­на має бути організована певним чином. Це стосується не лише інформації в комп'ютері, а й будь-якої інформації про об'єкти реаль­ного світу.

Основою бази даних є модель даних – фіксована система понять і правил для представлення даних структури, стану і динаміки проблемної області в базі даних. У різний час послідовне застосування одержували ієрархічна, мережна і реляційна моделі даних.

Вона являє собою логічно структуровану таблицю з полями, що описує дані, їх відносини між собою, операції, вироблені над ними, а головне – правила, які гарантують їх цілісність. Чому модель називається реляційної? В її основі лежать відносини (від лат. Relatio) між даними. Існує безліч визначень цього типу бази даних. Реляційні таблиці з інформацією набагато простіше систематизувати і надати обробці, ніж в мережевий або ієрархічної моделі.

***Реляційна модель БД*** – побудована на взаємовідношеннях між складовими структури: складається з сукупності взаємо­пов'язаних двовимірних таблиць.  
Кількість полів визначається розробником і не може змінюватися користувачем.

1. Будь-яке поле повинне мати унікальне ім’я.
2. Поля можуть мати різний тип:
   * рядок символів (довжиною до 255 символів)
   * дійсне число (з дробовою частиною)
   * ціле число
   * грошова сума
   * дата, час, дата и час
   * логічне поле (істина або лож, так або ні)
   * багаторядковий текст (МЕМО)
   * малюнок, звук або інший об’єкт (об’єкт OLE)
3. Поля можуть бути обов’язковими для заповнення або ні.
4. Таблиця може містити скільки завгодно записів (ця кількість обмежена тільки об’ємом диску); записи можна додавати, видаляти, редагувати, сортувати, шукати.

**2.1.1 Етапи створення баз даних**

***Постановка завдання.***На цьому етапі визначається мета створення бази даних, окреслюється предметна область, для якої буде створюватись база даних,

визначаються потенційні користувачі базою даних.

***Створення описової інформаційної моделі бази даних***, у якій виділяються сутності майбутньої бази даних, описуються зв’язки між ними.

***Створення моделі «сутність–зв’язок»***, визначення моделі даних.  
***Створення структури бази даних*** з використанням однієї із систем управління базами даних або однієї з мов програмування

***Введення даних***, встановлення зв'язків.

***Тестування бази даних***, її корекція.

Зв’язок встановлюється між двома загальними полями (стовпцями) двох таблиць. Існують зв’язки з відношенням «один-до-одного», «один-до-багатьох» і «багато-до-багатьох».

Відносини, які можуть існувати між записами двох таблиць:

**один** – **до** – **одного** – кожному запису з однієї таблиці відповідає один запис у іншій таблиці;

**один** – **до** – **багатьох** – кожному запису з однієї таблиці відповідає кілька записів у іншій таблиці;

**багато** – **до** – **одного** – безлічі записів з однієї таблиці відповідає один запис у іншій таблиці;

**багато** – **до** – **багатьох** – безлічі записів з однієї таблиці відповідає кілька записів в іншій таблиці.

Відношення «один-до-багатьох» створюється в тому випадку, коли тільки одне з полів є полем первинного ключа або унікального індексу.

Відношення «один-до-одного» створюється в тому випадку, коли обидва поля є ключовими або мають унікальні індекси.

Відношення «багато-до-багатьох» фактично є двома відносинами «один-до-багатьох» з третьої таблицею, первинний ключ якої складається з полів зовнішнього ключа двох інших таблиць.

**Ключ** – це стовпець (може бути декілька стовпців), що додається до таблиці і дозволяє встановити зв’язок із записами в іншій таблиці.

Існують ключі двох типів: первинні і вторинні (зовнішні).

**Первинний ключ** – це одне або кілька полів (стовпців), комбінація значень яких однозначно визначає кожний запис у таблиці. Первинний ключ не допускає значень Null і завжди повинен мати унікальний індекс. Первинний ключ використовується для зв’язування таблиці з зовнішніми ключами в інших таблицях.

**Зовнішній (вторинний) ключ** – це одне або кілька полів (стовпців) у таблиці, що містять посилання на поле або поля первинного ключа в іншій таблиці. Зовнішній ключ визначає спосіб об’єднання таблиць.

З двох логічно пов’язаних таблиць одну називають таблицею первинного ключа або головною таблицею, а іншу таблицею вторинного (зовнішнього) ключа або підпорядкованою таблицею. СУБД дозволяють зіставити споріднені записи з обох таблиць і спільно вивести їх у формі, звіті або запиті.

Існує три типи первинних ключів: ключові поля лічильника (лічильник), простий ключ і складовий ключ.

Поле лічильника (Тип даних «Счетчик»). Для кожного запису цього поля таблиці автоматично заноситься унікальне числове значення.

**Простий ключ.** Якщо поле містить унікальні значення, такі як коди чи інвентарні номери, то це поле можна визначити як первинний ключ. В якості ключа можна визначити всі поля, що містить дані, якщо це поле не містить повторювані значення або значення Null.

**Складові ключі.** У випадках, коли неможливо гарантувати унікальність значень кожного поля, існує можливість створити ключ, що складається з декількох полів. Найчастіше така ситуація виникає для таблиці, використовуваної для зв’язування двох таблиць відношенням «багато – до – багатьох».

Необхідно ще раз відзначити, що в полі первинного ключа повинні бути тільки унікальні значення в кожному рядку таблиці, тобто збіг не допускається, а в полі вторинного або зовнішнього ключа збіг значень у рядках таблиці допускається.

Якщо виникають труднощі з вибором потрібного типу первинного ключа, то в якості ключа доцільно вибрати поле лічильника.  
Програми, які призначені для структурування інформації, розміщення її в таблицях і маніпулювання даними, називаються системами управління базами даних (СУБД).

* 1. **Нормалізація відношень**

Під нормалізацією розуміється процес приведення відношення до однієї з нормальних форм (НФ). Але перед тим як розглядати нормальні форми, треба розповісти, для чого потрібна нормалізація.

При проектуванні баз даних в першу чергу робиться наголос на вірогідність та несуперечливість збережених даних, до того ж, ці властивості неповині втрачатися в процесі роботи з даними, тобто після багаточисельних змін, знищень та доповнень даних по відношенню до початкового стану бази даних.

Для підтримки бази даних в стійкому стані використовується ряд механізмів, які отримали узагальнену назву засобів підтримки цілісності. Ці механізми використовуються статично (на етапі проектування бази даних), так й динамічно (в процесі роботи з базою даних).

База даних повинна володіти обмеженнями, які будуть задовільняти її в процесі створення, незалежно від її наповнення даними. Доведення структури бази даних до відповідності цим обмеженням – це нормалізація. В цілому суть цих обмежень дуже проста: кожен факт, який зберігається в базі даних, повинен зберігатися один єдиний раз, так як дублювання може призвести до незгоди між копіями однакової інформації. Треба уникати невизначеностей, а також надмірності зберігаючої інформації. Отже, наша мета – доведення відношень до НФ. Треба відмітити, що в процесі нормалізації постійно зустрічається ситуація, коли відношення необхідно розкласти на декілька інших відношень. Тому коректніше було б казати про нормалізацію не окремих відношень, а всієї їх сукупності в базі даних.

Нормалізація може не викликати проблему, якщо база даних проектується одразу ж за визначеними правилами. Іншими словами, можна спочатку створити базу даних як-небудь, а потім нормалізувати її, або ж з самого початку будувати її за правилами для того, щоб в подальшому не треба було б її переробляти.

Нормалізація відношень забезпечує ефективність структур даних в реляційній БД. Цей процес зменшує надмірність даних (зберігання однакових даних в декількох місцях). В результаті більш раціонально використовується зовнішня пам’ять, зменшується ймовірність порушення узгоджуваності даних.

Нормалізація - це послідовне перетворення початкової бази даних до НФ, при цьому кожна наступна НФ обов’язково містить у собі попередню.

В реляційній теорії нараховують 6 НФ:

1 НФ;

2 НФ;

3 НФ;

НФ Бойса-Кодда (НФБК);

4 НФ;

5 НФ.

На практиці обмежуються 3 НФ, її достатньо для створення надійної схеми бази даних. НФ більш високих порядків уявляють академічну зацікавленість із-за надмірної складності. Крім того, при реалізації абстрактної схеми бази даних у вигляді реальної бази іноді розробники змушені зробити крок назад – провести денормалізацію з метою підвищення ефективності, так як ідеальна з точки зору теорії структура може статися занадто накладною на практиці.

Основні властивості нормальних форм:

• кожна наступна нормальна форма покращує властивості попередньої нормальної форми;

• при переході до наступної нормальної форми властивості попередніх нормальних форм зберігаються.

*Перша нормальна форма (1НФ)*. Відношення знаходиться в 1 НФ, якщо на перетині кожного стовпця з кожним рядком міститься лише атомарне (нерозподілене) значення. Іншими словами, кожен атрибут відношення повинен зберігати одне єдине значення та не бути ні списком, ні множиною значень. Атрибут, який є атомарним в одному додатку, може виявитися складовим в іншому.

*Друга нормальна форма (2НФ)*. Друга нормальна форма (2НФ) додатково стосується концепції видалення дублюючих даних:

Відповідати всім вимогам першої нормальної форми.

Видаліть підмножини даних, які застосовуються до декількох рядків таблиці, і помістіть їх в окремі таблиці.

Створіть зв’язки між цими новими таблицями та їх попередниками за допомогою використання зовнішніх ключів.

*Третя нормальна форма (3НФ)*. Третя нормальна форма (3НФ) йде на один великий крок далі:

Відповідати всім вимогам другої нормальної форми.

Видаліть стовпці, які не залежать від первинного ключа.

Нормальна форма Бойса-Кодда (БКНФ або 3.5НФ)

Нормальна форма Бойса-Кодда, яку також називають "третьою та половиною (3, 5) нормальною формою", додає ще одну вимогу:

Відповідати всім вимогам третьої нормальної форми.

Кожна детермінанта повинна бути ключовим кандидатом.

*Четверта нормальна форма (4НФ)*. Нарешті, четверта нормальна форма (4НФ) має ще одну додаткову вимогу:

Відповідати всім вимогам третьої нормальної форми.

Відношення є в 4НФ, якщо воно не має багатозначних залежностей.

Пам'ятайте, ці норми щодо нормалізації є сукупними. Щоб база даних знаходилася в 2НФ, вона повинна спочатку виконати всі критерії бази 1НФ.

Тобто, в нормалізованій базі даних для виконання транзакцій або запитів найбільш інтенсивно використовується центральний процесор, вимагається більше пам’яті та більша кількість операцій введення-виведення, ніж в ненормалізованій базі даних. В нормалізованій базі даних треба знаходити відповідні таблиці та пов’язувати дані для того, щоб вилучати потрібну інформацію та обробити її.

**2.3 Визначення типів даних**

При створенні таблиці для всіх її стовпців необхідно вказати певний тип даних. Тип даних визначає, які значення можуть зберігатися в стовпці, скільки вони будуть займати місця в пам'яті.

Залежно від характеру значень все їх можна розділити на групи:

*Числові типи даних:*

BIT : зберігає значення 0 або 1. Фактично є аналогом булевого типу в мовах програмування. Займає 1 байт.

TINYINT : зберігає числа від 0 до 255. Займає 1 байт. Добре підходить для зберігання невеликих чисел.

SMALLINT : зберігає числа від -32 768 до 32 767. Займає 2 байта

INT : зберігає числа від -2 147 483 648 до 2 147 483 647. Займає 4 байта. Найбільш використовуваний тип для зберігання чисел.

BIGINT : зберігає дуже великі числа від -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807, які займають в пам'яті 8 байт.

DECIMAL : зберігає числа c фіксованою точністю. Займає від 5 до 17 байт в залежності від кількості чисел після коми.

Даний тип може приймати два параметри precision і scale: DECIMAL(precision, scale).

Параметр precision представляє максимальну кількість цифр, які може зберігати число. Це значення повинно знаходитися в діапазоні від 1 до 38. За замовчуванням воно дорівнює 18.

Параметр scale представляє максимальну кількість цифр, які може містити число після коми. Це значення повинно знаходитися в діапазоні від 0 до значення параметра precision. За замовчуванням воно дорівнює 0.

NUMERIC : даний тип аналогічний типу DECIMAL.

SMALLMONEY : зберігає дробові значення від -214 748.3648 до 214 748.3647. Призначено для зберігання грошових величин. Займає 4 байти. Еквівалентний типу DECIMAL(10,4).

MONEY : зберігає дробові значення від -922 337 203 685 477.5808 до 922 337 203 685 477.5807. Являє грошові величини і займає 8 байт. Еквівалентний типу DECIMAL(19,4).

FLOAT : зберігає числа від -1.79E + 308 до 1.79E + 308. Займає від 4 до 8 байт в залежності від дробової частини.

*Типи даних, що представляють дату і час:*

DATE : зберігає дати від 0001-01-01 (1 січня 0001 року) до 9999-12-31 (31 грудня 9999 року). Займає 3 байта.

TIME : зберігає час в діапазоні від 00: 00: 00.0000000 до 23: 59: 59.9999999. Займає від 3 до 5 байт. Може мати форму TIME(n), де n представляє кількість цифр від 0 до 7 в дробової частини секунд.

DATETIME : зберігає дати і час від 01/01/1753 до 31/12/9999. Займає 8 байт.

DATETIME2 : зберігає дати і час в діапазоні від 01/01/0001 00: 00: 00.0000000 до 31/12/9999 23: 59: 59.9999999. Займає від 6 до 8 байт в залежності від точності часу. Може мати форму DATETIME2(n), де n представляє кількість цифр від 0 до 7 в дробової частини секунд.

SMALLDATETIME : зберігає дати і час в діапазоні від 01/01/1900 до 06/06/2079, тобто найближчі дати. Займає від 4 байта.

DATETIMEOFFSET : зберігає дати і час в діапазоні від 0001-01-01 до 9999-12-31. Зберігає детальну інформацію про час з точністю до 100 наносекунд. Займає 10 байт.

*Строкові типи даних:*

CHAR : зберігає рядок довжиною від 1 до 8 000 символів. На кожен символ виділяє по 1 байту. Не підходить для багатьох мов, так як зберігає символи не в кодуванні Unicode.

Кількість символів, яке може зберігати стовпець, передається в дужках. Наприклад, для стовпця з типом CHAR(10)буде виділено 10 байт. І якщо ми збережемо в стовпці рядок менше 10 символів, то вона буде доповнена пробілами.

VARCHAR : зберігає рядок. На кожен символ виділяється 1 байт. Можна вказати конкретну довжину для стовпця - від 1 до 8 000 символів, наприклад, VARCHAR(10). Якщо рядок повинен мати більше 8000 символів, то задається розмір MAX, а на зберігання рядка може виділятися до 2 Гб: VARCHAR(MAX).

Не підходить для багатьох мов, так як зберігає символи не в кодуванні Unicode.

На відміну від типу CHAR якщо в стовпець з типом VARCHAR(10)буде збережена рядок в 5 символів, то в Столц буде збережено саме п'ять символів.

NCHAR : зберігає рядок в кодуванні Unicode завдовжки від 1 до 4 000 символів. На кожен символ виділяється 2 байти. наприклад,NCHAR(15)

NVARCHAR : зберігає рядок в кодуванні Unicode. На кожен символ виділяється 2 байта.Можно задати конкретний розмір від 1 до 4 000 символів: . Якщо рядок повинен мати більше 4000 символів, то задається розмір MAX, а на зберігання рядка може виділятися до 2 Гб.

**2.4 Проектування структури бази даних**

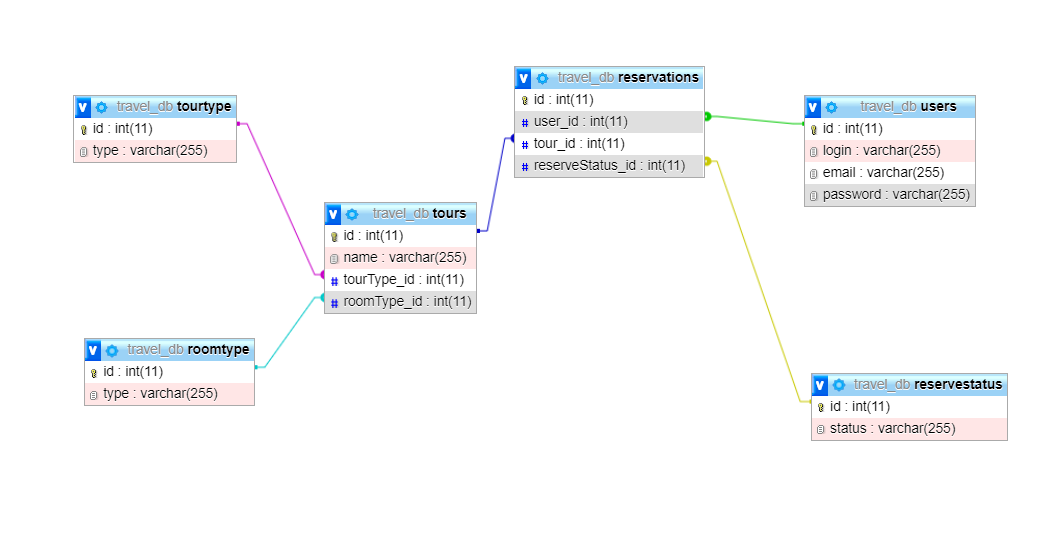
Для розробки моделі бази даних обрана реляційна модель даних. Вона найкраще підходить для вирішення цієї задачі, адже вона має ряд наступних переваг:

- простота розробки та моделювання інформаційного ресурсу як плата за деякі обмеження та уніфікацію на рівні реалізації операцій над даними;

- незалежність програм від даних. Ідея використання баз даних та систем управління базами даних передбачає використання додаткового рівня між прикладними програмами та власне даними, завдяки чому прикладні програмісти можуть абстрагуватися від реалізації самої бази даних, а зосередити свою увагу на логіці обробки даних;

- наявність умов керування даними за допомогою операцій над множинами.

В процесі проектування структури бази даних потрібно створити діаграму концептуальної моделі даних. На основі визначених елементів і зв’язків створити ER – діаграму. Діаграма концептуальної моделі даних представлена на рисунку 2.4.1



**Рис.2.4.1 Діаграма концептуальної моделі даних**

**2.4.1 Перелік таблиць та полів бази даних**

У процесі розробки проекту у базі даних «travel\_db» створено таблиці:

«users»– таблиця з користувачами;

«reserveStatus»– таблиця з статусом резервації;

«tourType»– таблиця з типом туру;

«roomType»– таблиця типом кімнати;

«tours»– таблиця з назвою туру;

«reservations»–таблиця з назвою резервації;

Перелік полів таблиць бази даних «travel\_db»:

«Login» –логін

«Email» –емейл

«Password»–пароль

«Name»–ім’я

«tourType\_id»–тип туру

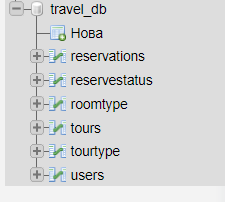
«roomType\_id»–тип кімнати

«user\_id»–користувач

«tour\_id»–тур

«reserveStatus\_id»–статус резервації

«Status»–статус



**Рис. 2.4.2 Таблиці в базі даних «travel\_db»**

CтруктураMySQLтрирівнева: бази даних – таблиці – записи. Бази даних і таблиці MySQ Lфізично представляються файлами з розширеннями frm, MYD, MYI. Ім'я бази даних MySQL унікальна в межах системи, а таблиці - в межах бази даних, поля - в межах таблиці. Таблиці складаються із записів, а записи, у свою чергу, складаються з полів. Поле має два атрибути - ім'я і тип даних.

Основні типии даних, які використовуються в моїй базі даних MySQL:

- VARCHAR –може зберігати не більше 255 символів;

- INT – діапазон від -2 147 483 648 до 2 147 483 647.

*Таблиця 2*

**Склад таблиці«tours»**

|  |  |  |  |
| --- | --- | --- | --- |
| **№п/п** | **Найменуванняатрибутів** | **Типполів** | **Розмір полів** |
| 1 | id | INT | 11 |
| 2 | name | VARCHAR | 255 |
| 3 | tourType\_id | INT | 11 |
| 4 | RoomType\_id | INT | 11 |

*Таблиця 3*

**Склад таблиці «reservestatus»**

|  |  |  |  |
| --- | --- | --- | --- |
| **№п/п** | **Найменуванняатрибутів** | **Типполів** | **Розмір полів** |
| 1 | id | INT | 11 |
| 2 | status | VARCHAR | 255 |

*Таблиця 4*

**Склад таблиці «users»**

|  |  |  |  |
| --- | --- | --- | --- |
| **№п/п** | **Найменуванняатрибутів** | **Типполів** | **Розмір полів** |
| 1 | id | INT | 11 |
| 2 | login | VARCHAR | 255 |
| 3 | email | VARCHAR | 255 |
| 4 | password | VARCHAR | 255 |

*Таблиця 5*

**Склад таблиці «reservations»**

|  |  |  |  |
| --- | --- | --- | --- |
| **№п/п** | **Найменуванняатрибутів** | **Типполів** | **Розмір полів** |
| 1 | id | INT | 11 |
| 2 | user\_id | VARCHAR | 255 |
| 3 | tour\_id | INT | 11 |
| 4 | ReserveStatus\_id | INT | 11 |

*Таблиця 6*

**Склад таблиці «tourtype»**

|  |  |  |  |
| --- | --- | --- | --- |
| **№п/п** | **Найменуванняатрибутів** | **Типполів** | **Розмір полів** |
| 1 | id | INT | 11 |
| 2 | type | VARCHAR | 255 |

*Таблиця 7*

**Склад таблиці «roomtype»**

|  |  |  |  |
| --- | --- | --- | --- |
| **№п/п** | **Найменуванняатрибутів** | **Типполів** | **Розмір полів** |
| 1 | id | INT | 11 |
| 2 | type | VARCHAR | 255 |

Проектування виконується на основі інфологічної схеми і полягає у виконанні таких пунктів:

* виділення таблиць (R-відношень);
* визначення фізичних форматів атрибутів на основі типів СУБД;
* визначення складу індексованих полів;
* визначення зв'язків навігації і логічної цілісності;
* проектування представлень;
* проведення нормалізації реляційних схем (усунення надлишковості, багатозначності).

Результат даталогічного проектування – реляційна схема бази даних.

**2.4.2 Функціональні залежності між атрибутами**

Після того, як таблиця приведена до першої нормальної форми 1НФ, потрібно визначити функціональну залежність між атрибутами (полями, стовпцями) таблиці. Це необхідно для забезпечення максимально-можливої раціональності в побудові таблиць бази даних.

Функціональна залежність – це зв’язок, що може виникнути між сутностями, що зберігаються в базі даних.

*Таблиця 8*

**Функціональна залежність«tourtype»**

|  |  |
| --- | --- |
| **Найменуванняатрибутів** | **Функціональнізалежності** |
| іd  type |  |

*Таблиця 9*

**Функціональна залежність «tours»**

|  |  |
| --- | --- |
| **Найменуванняатрибутів** | **Функціональнізалежності** |
| id  name  tourType\_id  roomType\_id |  |

*Таблиця 10*

**Функціональна залежність «roomtype»**

|  |  |
| --- | --- |
| **Найменуванняатрибутів** | **Функціональнізалежності** |
| id  email |  |

*Таблиця 11*

**Функціональна залежність «reservations»**

|  |  |
| --- | --- |
| **Найменуванняатрибутів** | **Функціональнізалежності** |
| іd  user\_id  tour\_id  reserveStatus\_id |  |

*Таблиця 12*

**Функціональна залежність «users»**

|  |  |
| --- | --- |
| **Найменуванняатрибутів** | **Функціональнізалежності** |
| іd  login  email  password |  |

*Таблиця 13*

**Функціональна залежність «reservestaus»**

|  |  |
| --- | --- |
| **Найменуванняатрибутів** | **Функціональнізалежності** |
| id  status |  |

**2.5. Реалізація SQL- скрипту**

SQL – декларативна [мова програмування](https://uk.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) для взаємодії користувача з [базами даних](https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), що застосовується для формування запитів, оновлення і керування [реляційними БД](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85), створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних.

*createtableifnotexistsusers(*

*idintprimarykeyauto\_increment,*

*loginvarchar(255) notnull,*

*emailvarchar(255) notnull,*

*passwordvarchar(255) notnull)*

*createtableifnotexistsreserveStatus(*

*idintprimarykeyauto\_increment,*

*statusvarchar(255) notnull)*

*select \* fromreserveStatus*

*insertintoreserveStatus(status) values ?*

*createtableifnotexiststourType(*

*idintprimarykeyauto\_increment,*

*typevarchar(255) notnull)*

*select \* fromtourType*

*insertintotourType(type) values*

*createtableifnotexistsroomType(*

*idintprimarykeyauto\_increment,*

*typevarchar(255) notnull)*

*select \* fromroomType*

*insertintoroomType(type) values ?*

*createtableifnotexiststours(*

*idintprimarykeyauto\_increment,*

*namevarchar(255) notnull,*

*tourType\_idint,*

*roomType\_idint,*

*FOREIGN KEY(tourType\_id) REFERENCES tourType (id),*

*FOREIGN KEY(roomType\_id) REFERENCES roomType (id)*

*select \* fromtours*

*INSERT INTO tours(name, tourType\_id, roomType\_id) VALUES ?*

*createtableifnotexistsreservations(*

*idintprimarykeyauto\_increment,*

*user\_idint,*

*tour\_idint,*

*reserveStatus\_idint,*

*FOREIGN KEY(user\_id) REFERENCES users (id),*

*FOREIGN KEY(tour\_id) REFERENCES tours (id),*

*FOREIGN KEY(reserveStatus\_id) REFERENCES reserveStatus (id)*

*INSERT INTO reservations(user\_id, tour\_id, reserveStatus\_id) VALUES ?*

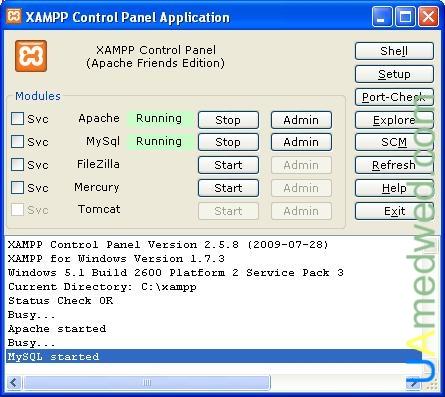
*SELECT \* FROM users WHERE email=? AND password=?*

*SELECT \* FROM users WHERE email=?*

*INSERT INTO users(login, email, password) VALUES(?, ?, ?)*

**РОЗДІЛ 3. РОЗРОБКА WEB-САЙТУ**

Я обрала **XAMPP як локальний сервер.** XAMPP зручний у використанні, і гарна підтримка, тобто розробники XAMPP завжди стежать за тим що б при появі нової версії тієї чи іншої бібліотеки, комплект Web-сервера оновлювався. XAMPP не створює ніяких додаткових дисків, і містить дуже широкий вибір бібліотек, які можна додати.



**Рис. 3.1 Модулі, підключені за допомогою XAMPP**

Програмна реалізація веб-сервісу почалася з верстки шаблону сайту.

Верстка виконувалася мовою розмітки гіпертексту HTML. Для наданняр озмітці стилів використовувалися каскадні таблиці стилів CSS. Для адміністрування системи управління базами даних MySQL використовується веб-застосунок з графічним веб-інтерфейсом, розроблений на мові програмування PHP – phpMyAdmin.

Післязавершення верстки сайту, потрібно підключити форми до бази даних.

**3.1. Структура web-сайту**

Структура сайту ділиться на зовнішню та внутрішню.

Основними елементами Зовнішньої структури є:

- **Шапка сайту**, яка містить логотип і назву сайту. Шапка сайту може  вирішувати декілька завдань. По-перше, це візитна картка інтернет-ресурсу.  По-друге, шапка сайту дуже часто використовується для розміщення в ній  основного меню.

- **Меню** – це засіб для переходу до основних розділів сайту. Наприклад,  таких, як каталог файлів, форум, зворотній зв’язок та ін.

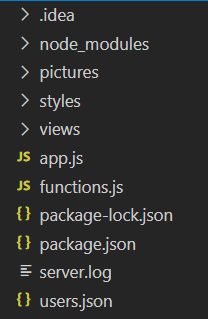
- **Контент** (англ. content – зміст) – змістове наповнення веб-сторінки,  доступне користувачу: тексти, зображення, відео, звукові дані та інше. Перед нами головна сторінка шкільного сайту, основним контентом  якої є новини.

- **Підвал сайту** або **футер**, є менш функціональною частиною. В ньому  містяться контактні дані та повідомлення про авторські права. Окрім цього, на сайті можуть бути присутні посилання на інші сайти,  різноманітні інформери (наприклад, погода у Львові, святковий календар) та  інші елементи.

Внутрішня структура сайту - логічне розташування всіх елементів на сторінках, їх взаємопов'язаність по тематиці і значенням, а також впорядкованість сторінок ресурсу. Неправильна компоновка розділів призводить до того, що частина інформації залишається «за кадром» як для пошукових систем, так і для користувачів.

Грамотна побудова ресурсу необхідно для досягнення наступних завдань.

1. **Простота розробки.**Якщо чітко визначити контури проекту і заздалегідь встановити зв'язки між об'єктами, часу на написання коду витрачається менше, а готовий веб-продукт виходить більш якісним.
2. **Можливість модернізації.** При наявності оптимально побудованої схеми ресурсу нарощування числа сторінок і розділів чи не позначиться на якості просування.
3. **Юзабіліті.**Зовнішня організація сторінок повинна бути такою, щоб користувачу було зручно дивитися матеріали, а на пошук потрібної інформації витрачалося мінімум часу.

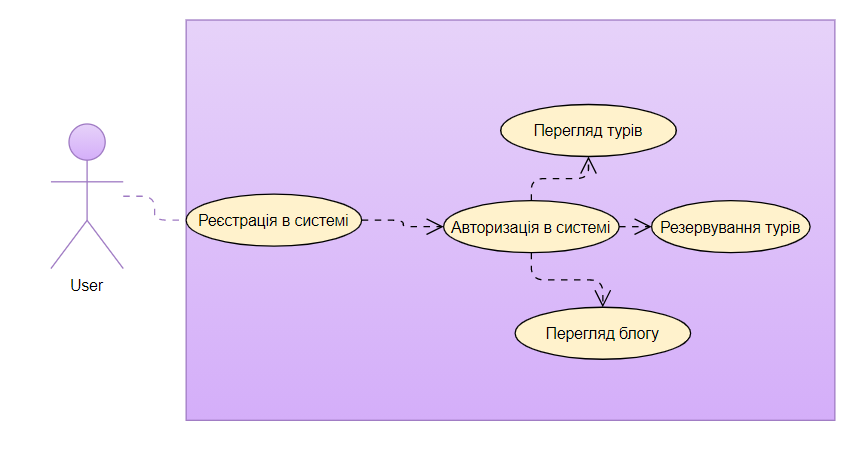


**Рис. 3.1.1. Структура папок, з яких складається проект**

**3.1.1. Варіанти використання сайту**

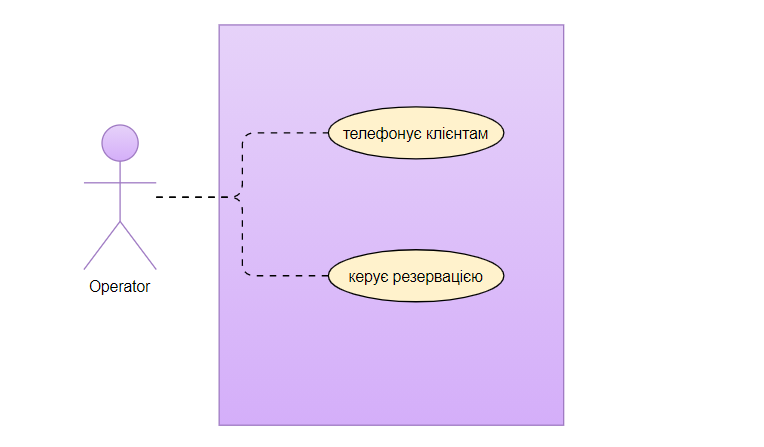
Доступ до сайту відбувається через адресу:<http://localhost:3000/>

Вигляд діаграми варіантів використання для користувачі продукту на рисунку 3.1.



**Рис. 3.1.1.1.Діаграма варіантів використання для користувача**

Вигляд діаграми варіантів використання для оператора продукту на рисунку 3.2



**Рис. 3.1.1.2 – Діаграма варіантів використання для оператора**

Клієнтська частина має такі головні сторінки:

**Home** – домашня сторінка, дозволяє повернутись на головну сторінку сайту з будь-якої іншої сторінки;

**Blog**– сторінка, за допомогою якої користувач може відслідковувати блог про подорожі;

**Contact**–сторінка, яка містить в собі контактну інформацію;

**SignIn**–сторінка, яка містить форму авторизації за допомогою якої користувач може ввійти в базу даних;

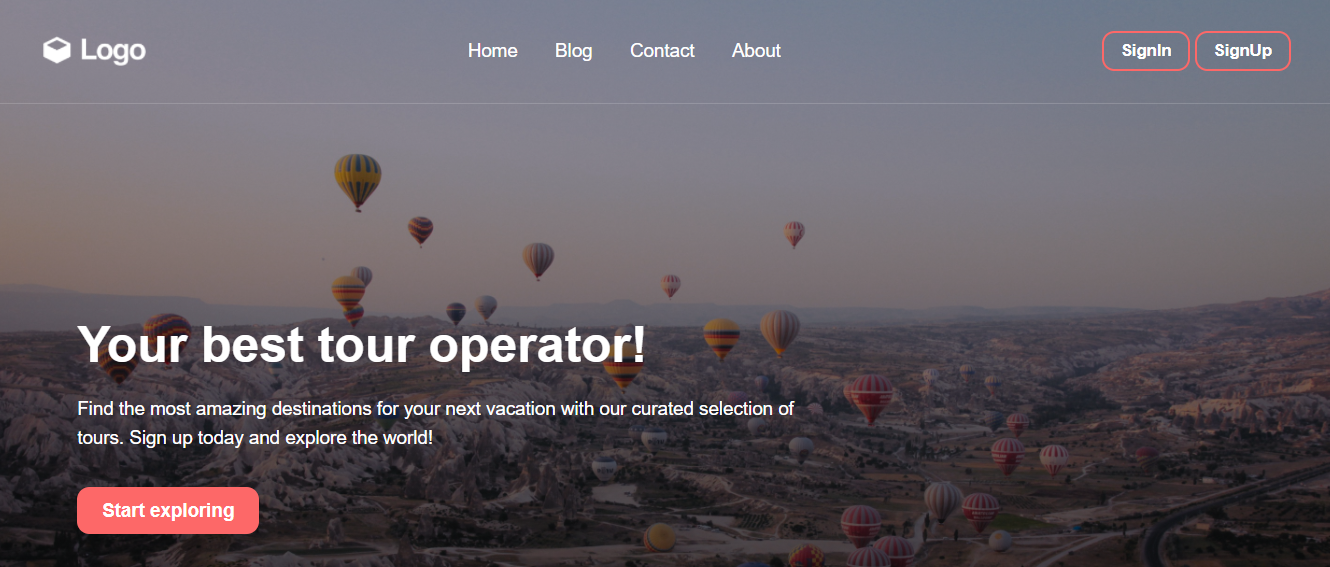
**SignUP**–сторінка, яка містить форму реєстрації, за допомогою якої користувач може зареєструватись в базу даних;

**3.2. Макети сторінок web-сайту**

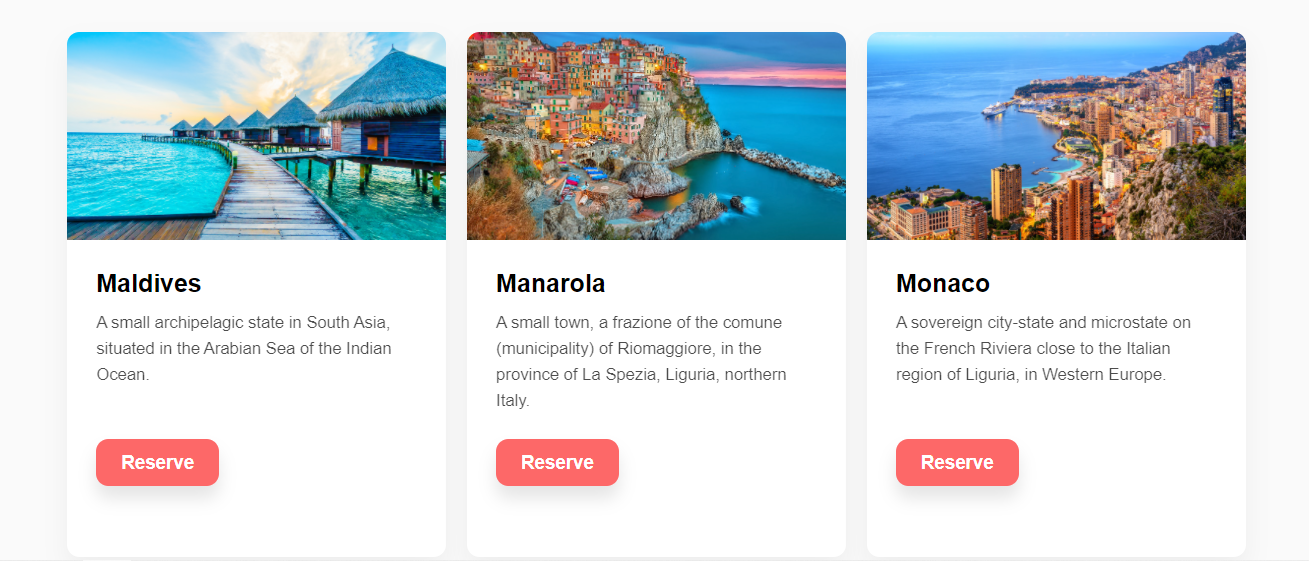
**Дизайн-макет сайту**  - це візуальний образ майбутнього сайту, розроблений з урахуванням технічних можливостей HTML верстки. Такий макет є демонстрацією того, як візуально буде виглядати ваш сайт після верстки та наповнення.

Макет представляється у вигляді картинки, яка буде відображена в інтернет браузері, без активних кнопок і інших динамічних елементів.

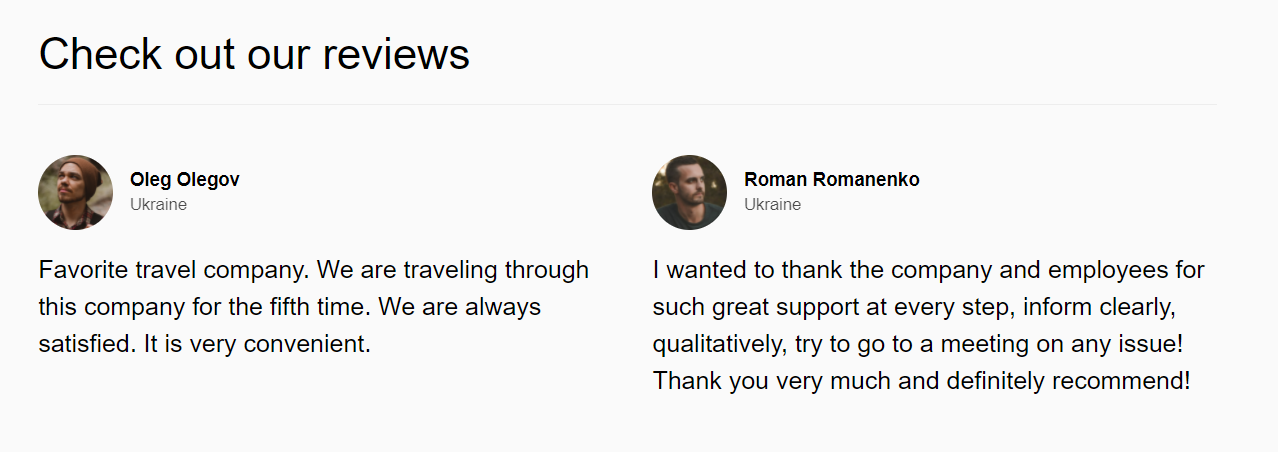
Специфіка розробки графічного дизайн-макету стосовно сайту вдає із себе поєднання технічних і візуальних параметрів майбутнього сайту. Це опрацювання розташування і розміру елементів сайту з точки зору зручності пошуку і використання інформації на сайті.



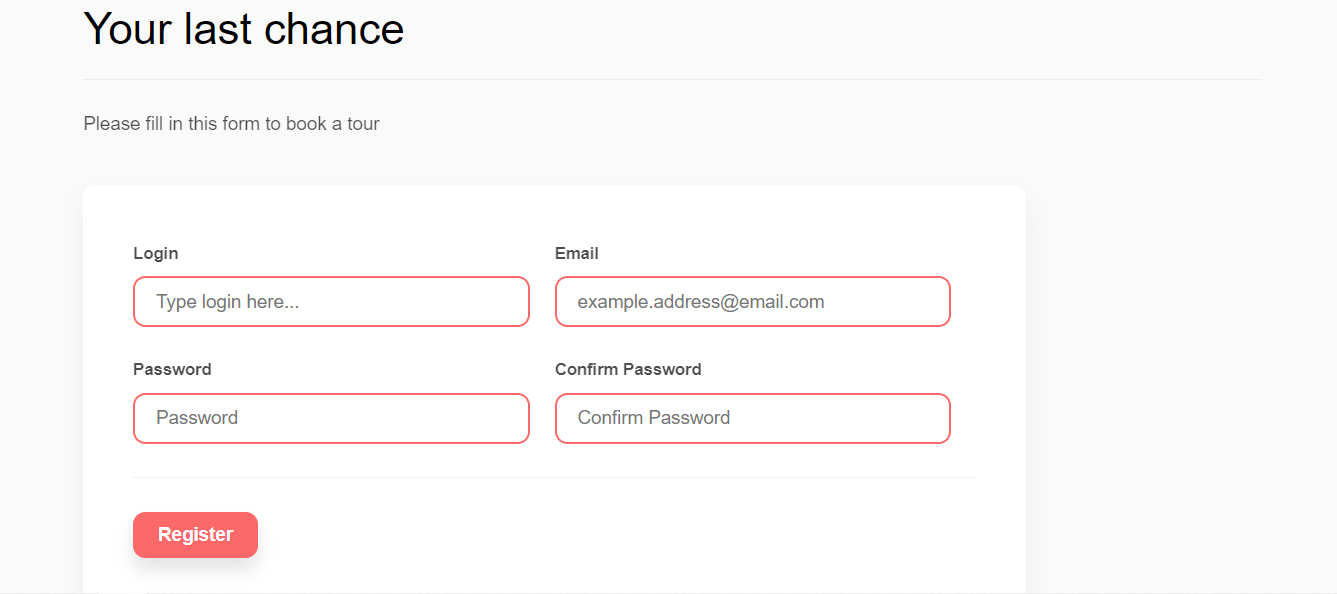
**Рис. 3.2.1. Header та intro сторінки web-сайту**



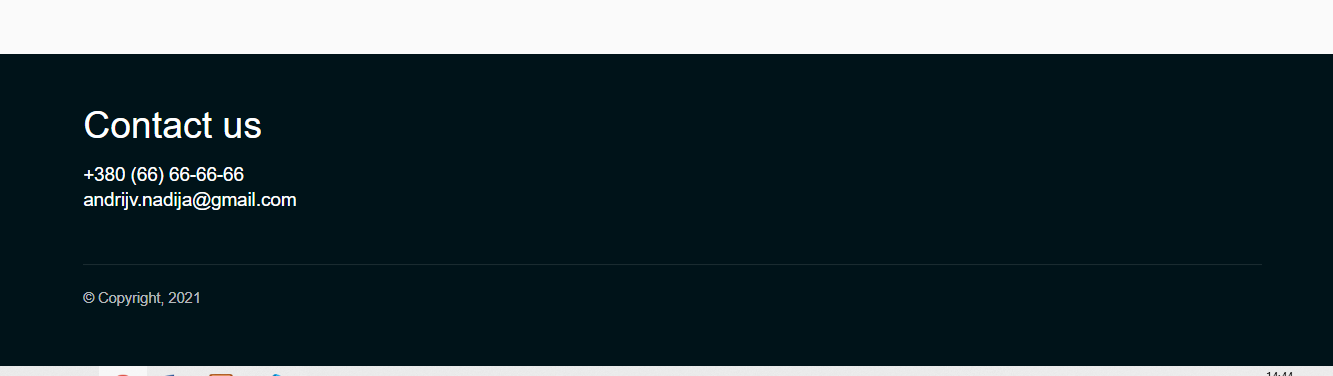
**Рис. 3.2.2. Секція «Special offers» сторінки web-сайту**



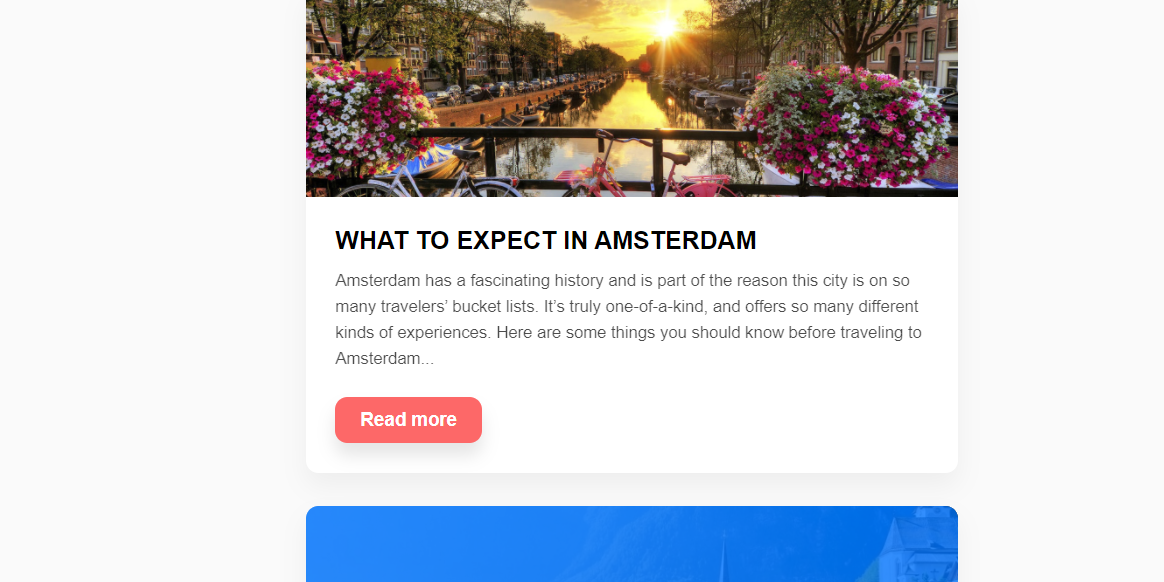
**Рис. 3.2.3. Секція «Reviews» сторінки web-сайту**

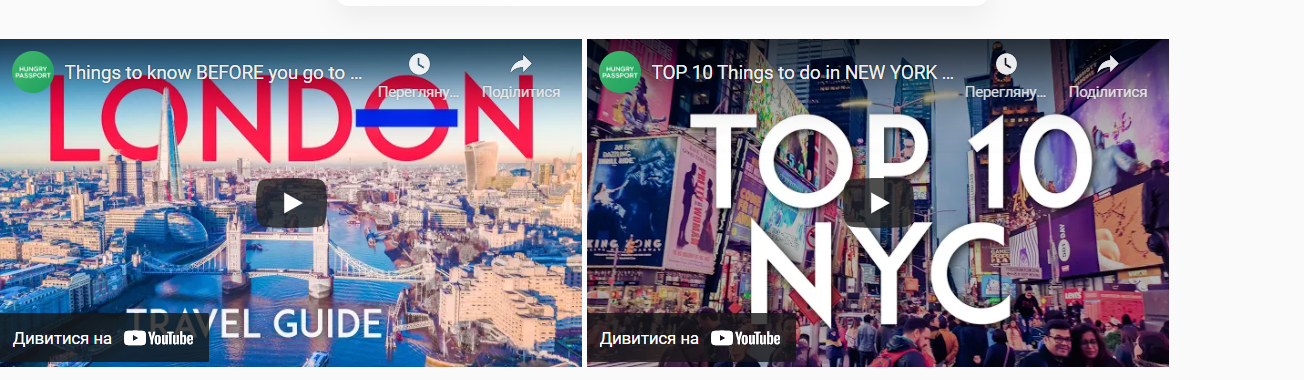


**Рис. 3.2.4. Секція «Registration» сторінки web-сайту**



**Рис. 3.2.5. Footer сторінки web-сайту**





**Рис. 3.2.6. Blog web-сайту**

**3.3. Програмування серверної частини**

За допомогою скриптівnodejs організовано такі операції для функціонування сайту:

1. Реєстрація
2. Авторизація
3. Деавторизація
4. Бронювання

Для роботи з сервером і протоколом http в Node.js використовується модуль http. Щоб створити сервер, слід викликати метод http.createServer()

Node.js використовує модульну систему, тобто вся вбудована функціональність розбита на окремі пакети або модулі. Модуль представляє блок коду, який може використовуватися повторно в інших модулях.

Для завантаження модулів застосовується функція require(), в яку передається назва модуля. Наприклад, в першому додатку з попередньої теми для отримання і обробки запиту був необхідний модуль http :

consthttp = require("http"); Після отримання модуля ми зможемо використовувати весь визначений у ній функціонал:

1. *Рестрація користувача:*

let user = [req.body.login, [req.body.email](http://req.body.email/), req.body.password];  
constsql\_insert = "INSERT INTO users(login, email, password) VALUES(?, ?, ?)";  
  
connection.query(sql\_insert, user, function (error, data) {  
if (error) console.log(error);  
else console.log("User addedtoDatabase");  
  
console.log('SetSession');  
  
req.session.user = data.insertId;  
req.session.flag = true;  
  
status = 'SUCCESS';  
res.redirect('/home');  
});

1. *Валідація даних*

Початок форми

Кінець форми

module.exports.validate = functionvalidate(req) { varerrors = []; if(req.body.login == '') errors.push('Loginisrequired!');

if(req.body.email == '' || !req.body.email.includes('@')) errors.push('Emailisrequired!');

if(req.body.password == '') errors.push('Passwordisrequired!');

elseif(req.body.confirmPassword != req.body.password) errors.push('Passwordmismatch!');

returnerrors;

}

Для більш зручного управління конфігурацією і пакетами додатки в npm застосовується файл конфігурації package.json .

Файл package.json грає велику роль і може полегшити роботу з пакетами в різних ситуаціях. Наприклад, ми плануємо використовувати безліч пакетів. Але вводити для установки кожного пакета в консолі відповідну команду не дуже зручно. У цьому випадку ми можемо визначити всі пакети в файлі package.json і потім однією командою їх встановити.

*Особливості nodemon*:

Автоматичний перезапуск програми. Виявляєрозширення файлу за замовчуванням для моніторингу. Підтримкавузла за замовчуванням, але легко запускати будь-якийвиконуваний файл, такий як python, ruby, makeтощо. Ігноруванняпевнихфайлівабокаталогів. Переглядайтеконкретні каталоги. Працює з сервернимипрограмамиабоодноразовимиутилітами та REPL. Для скриптів через вузолпотрібніоператори. Відкритий код і доступний наgithub.

ШаблонізаторHandlebars для JavaScript надає умовний оператор if (так само можливе використання і else ), але оператор if може обробляти тільки єдине значення (не вираження).

Як правило, додатки Express для визначення візуального інтерфейсу використовує не стандартні файли html, а спеціальні суті - уявлення, з яких потім створюються html-файли. Перевагою уявлень є те, що ми можемо визначати в них деякі шаблони, замість яких потім вставляється якесь динамічний вміст за допомогою коду javascript.

Управляє уявленнями спеціальний компонент - движок уявлень (viewengine), який також називають движок шаблонів (templateengine). Взагалі движків уявлень в Express досить багато: Pug, Jade, Dust, Nunjucks, EJS, Handlebars і інші. Питання вибору движка уявлень - в основному питання переваг, всі вони надають схожу функціональність, розрізняючи лише в якихось деталях.

Для роботи з двигунами уявлень в Express визначено ряд загальних параметрів, які ми можемо встановити. Перш за все це настройка viewengine , яка встановлює використовуваний движок предствлений, і views , яка встановлює шлях до папки з уявленнями всередині проекту (якщо цей параметр не встановлено, то за замовчуванням використовується папка з ім'ям views ).

Для початку розглянемо роботу з уявленнями на основі движка Handlebars або скорочено hbs.

Одним з переваг шаблонів є те, що ми можемо передавати в уявлення на місце шаблонів моделі уявлення - спеціальні об'єкти, дані які використовує движок уявлень для рендеринга.

**3.4. Програмування клієнтської частини**

Клієнтська **частина** це Frontend розробка, до неї можна віднести CSS верстку, створення макета і шаблонів для **сайту**, а також призначеного для користувача інтерфейсу і спеціальних скриптів, що відповідають за візуалізацію і анімацію.

Дизайн мого сайту базувався на використовувані технології flexbox. Це нова технологія, яка вже має досить широку підтримку браузерів.

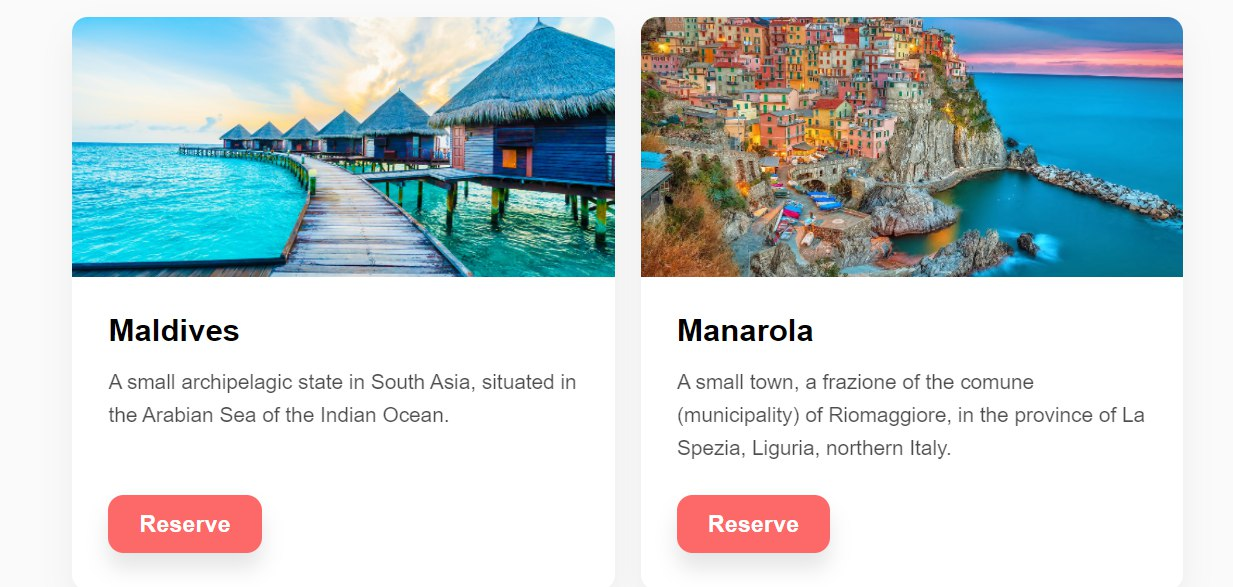
 Flexbox надає інструменти для швидкого створення складних, гнучких макетів, і функції, які були складні в традиційних методах CSS.

Довгий час єдиними надійними інструментами CSS верстки були такі способи як Float (обтікання) і позиціонування.З їх допомогою складно або неможливо досягти наступних простих вимог до макету.

Flexbox розмітка в CSS дає один з найбільш ефективних способів розстановки, вирівнювання і розподілу місця між елементами всередині контейнера, навіть якщо їх розмір невідомий або динамічний (власне, з цього його і називають flex, від слова flexible, що по-англійськи має подвійне значення – гнучкий і поступливий, що дуже поєднується з моделлю поведінки flexbox).

Основною метою flexbox є надання можливості зміни своїх елементів по ширині і висоті, для того, щоб вони максимально ефективно вміщалися в доступному місці батьківської контейнера, зокрема - це зручно в тих випадках, коли потрібно відповідати всім типам дисплеїв пристроїв і розмірами екранів. Flex контейнер розширює вкладені елементи для того, щоб заповнити обсяг пам'яті, доступний або ж урізує їх, щоб уникнути переповнення.

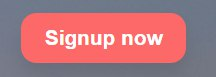
Дуже важливий момент, flexbox незалежний від напрямку, на відміну від звичайних шаблонів, блокова модель заснована на вертикальному розміщенні елементів, а in-line на горизонтальній.



**Рис. 3.4.1. Демонстрація використання flexbox**

Особливість мого дизайну у використанні псевдокласів, які визначають динамічний стан елементів, які змінюється за допомогою дій користувача, а також положення в основі документа. Прикладом такого стану служить текстове посилання, яке змінює свій колір при наведенні на неї курсора миші. При використанні псевдокласів браузер не перевантажує поточний документ, тому за допомогою псевдокласів можна отримати різні динамічні ефекти на сторінці.

CSStransitions можуть застосовуватися до всіх елементів і навіть до псевдоелементів. Використовуються для пожвавлення нашої верстки. Що в свою чергу призводить до поліпшення взаємодії з користувачем, як правило шляхом приємною анімаційною реакцією на його дії. Наприклад, наведення і натиснення на кнопку «signup now».



**Рис. 3.4.2. Демонстрація анімаційної реакції за допомогою transitions**

**CSS transitions**  надають спосіб контролювати швидкість анімації, при зміні CSS властивостей. Замість того, щоб властивість застосувати відразу, ви можете зробити це дію, що відбувається протягом якогось моменту часу.

CSS переходи дозволяють вам вирішити які властивості потрібно анімувати (перерахуванням їх явно), коли анімація почнеться (установкою затримки), як довго перехід буде виконуватися (установкою тривалості), а також як вона буде запущена (ухвалою функції за часом, наприклад лінійно або швидко в початку, повільно в кінці).

**ВИСНОВКИ**

В pозділі 1 було опрацьовано предметну область системи, визначено переваги та недоліки розробки web-сайту, визначено характеристики бізнес-процесу.

У даному розділі було проаналізовано предметну область для розроблюваного продукту. Досліджено структуру та напрями діяльності об’єкту управління. Подано характеристику процесу реєстрації користувача і характеристику використання web-сайту користувачем.

В pозділі 2 було спроектовано та описано структуру БД, її таблиці та записи. Визначено типи даних та функціональні залежності. Розроблено запити та визначено атрибути полів. Для кращого розуміння архітектури продукту було розроблено діаграми, які показують взаємодію користувачів та їх функцій між собою та у системі.

Оскільки дана система оперує даними, було спроектовано архітектуру бази даних. База даних є реляційною, була створена діаграма корпоративної моделі даних, яка показує об’єкти системи та зв’язки між ними(ER-діаграма). Вона показує взаємодію між відношеннями і дозволяє приступити до програмування бази даних.

В розділ 3 представлено практичне завдання. В ході практичного завдання було розроблено web-сайт використовуючи nodejs – платформу з відкритим кодом для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript. Програмна реалізація веб-сайту почалася з верстки шаблону сайту.

Верстка виконувалася мовою розмітки гіпертексту HTML. Для надання розмітці стилів використовувалися каскадні таблиці стилів CSS. Та за допомогою модуля flexbox вирішила такі завдання як: розташування елементів, порядок відображення елементів, автоматичне визначення розділу елемента, створення колонок однакової висоти.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Буч Г. Язык UML: руководствопользователя / Г. Буч, Д. Рамбо, И. Якобсон // [пер. с англ. Н. Мухина]. – Москва. – 2019. – 493 с.
2. Дивак М.П. Системний аналіз та проектування КІС /М.П.Дивак// Навчальний посібник – Т.: Економічна думка. – 2018.
3. Калбертсон Роберт, Браун Крис, КоббГэриБыстроетестирование. — М.: «Вильямс», 2017. — 374 с.
4. Костарев А. Ф. PHP 5. — Спб.: «БХВ-Петербург», 2016. — С. 1104.
5. Кузнецов Максим, СимдяновИгорь. PHP. Практика созданияWebсайтов. — 2-е изд. перераб. и доп. — Спб.: «БХВ-Петербург», 2018. — С. 1264.
6. ЛайзаКриспин, ДжанетГрегориГибкоетестирование: практическоеруководство для тестировщиков ПО и гибких команд = AgileTesting: A PracticalGuideforTestersandAgileTeams. — М.: «Вильямс», 2016. — 464 с
7. МэттЗандстра. PHP: объекты, шаблоны и методики программирования, 3-е издание = PHP Objects, PatternsandPractice, ThirdEdition. — М.: «Вильямс», 2015. — С. 560
8. Пасічник В. В. Організація баз даних та знань / В.В. Пасічник, В.А. Резніченко. – К.: Видавничагрупа BHV, 2017. – 384 с.
9. Синицын С. В., Налютин Н. Ю. Верификацияпрограммногообеспечения. — М.: БИНОМ, 2016. — 368 с.
10. Том 1. Основы, 7-е изд.» : підручник / К. С. Хорстманн, Г. Корнелл : Издательскийдом "Вильямс",2007. – 896 с.
11. Юр'єв В.К., Куценко Г.І. Громадське здоров'я та охорона здоров'я. СП, 2016. - С. 240-283.
12. CSSstyles[Електронний ресурс]. - Режим доступу:<http://htmlbook.ru/css>
13. Dash, Python [Електронний ресурс]. – Режим доступу: <https://dash.plot.ly>.
14. GoslingTheJavaLanguageSpecificationJava SE 8 Edition [Електронний ресурс]. – Режим доступу: http://docs.oracle.com/javase/specs/ jls/se8/jls8.pdf
15. PHP [Електронний ресурс]. https://en.wikipedia.org/wiki/PHP
16. PHPDeveloper[Електронний ресурс]. - Режим доступу:<https://dev.to/full_stackgeek/5-articles-being-a-php-developer-you-should-read-21j9>
17. PHP documentation [Електронний ресурс]. - Режим доступу: <http://php.net/docs.php/>
18. Webdeveloperinformationwebsite [Електронний ресурс]. - Режим доступу: <http://www.w3schools.com/>
19. Webdeveloperinformationwebsite [Електронний ресурс]. - Режим доступу: <https://uk.wikipedia.org/wiki/PHP>
20. Вікіпедія. [Електронний ресурс]. - Режим доступу: <https://uk.wikipedia.org>
21. Сpp-reference. [Електронний ресурс]. – Режим доступу : <http://cpp-reference.ru/patterns/behavioral-patterns/command/>
22. Статті PHP[Електронний ресурс]. - Режим доступу:<https://tutorialzine.com/tag/php>