­­РЕФЕРАТ

Пояснювальна записка до курсового проекта на тему «Розробка веб-додатку для туристичної фірми Traveler» складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на 34 сторінках друкованого тексту.

Пояснювальна записка має 5 сторінок додатків. Список використаних джерел містить 9 найменувань і займає 1 сторінку. В роботі наведено 13 рисунків. Загальний обсяг роботи – 40 сторінки.

У першому розділі було обґрунтовано створення сайту на тему Розробка інтернет-магазину професійного спорядження для полювання.

У другому розділі проведено проектування і розробка сайту.

У третьому розділі проведено тестування програмного продукту.

Висновок містить в собі результати виконаної роботи при створенні сайту на тему «Розробка веб-додатку для туристичної фірми Traveler».

У додатку представлений лістинг розробленого програмного продукту.

Ключові слова: JAVASCRIPT, HTML, CSS, NODE.JS, MONGODB, EXPRESS, WEB-САЙТ, MVC, АДМІН, АВТОРИЗАЦІЯ, ІНТЕРНЕТ, ДАНІ.

**ЗМІСТ**

[ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 3](#_Toc62517846)

[ВСТУП 4](#_Toc62517847)

[РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ 5](#_Toc62517848)

[1.1 Аналіз задачі, засобів та методів її вирішення 5](#_Toc62517849)

[1.2 Аналіз існуючого програмного забезпечення за тематикою курсового проекту 7](#_Toc62517850)

[1.3 Технічне завдання на курсову роботу 11](#_Toc62517851)

[Висновки до розділу 14](#_Toc62517852)

[РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 15](#_Toc62517853)

[2.1 Проектування загального алгоритму роботи програми 15](#_Toc62517854)

[2.2 Розробка функціональних алгоритмів роботи програми 16](#_Toc62517855)

[2.3 Розробка програмного забезпечення 18](#_Toc62517856)

[Висновки до розділу 22](#_Toc62517857)

[РОЗДІЛ 3 ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ 23](#_Toc62517858)

[3.1 Опис роботи з програмним додатком 23](#_Toc62517859)

[3.2 Тестування веб-додатку 27](#_Toc62517860)

[Висновки до розділу 29](#_Toc62517861)

[ВИСНОВКИ 30](#_Toc62517862)

[ВИКОРИСТАНІ ДЖЕРЕЛА 31](#_Toc62517863)

[ДОДАТКИ 32](#_Toc62517864)

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

1. БД – база даних
2. СУБД – система управління базами даних
3. CMS – (англ. Content Management System) система керування вмістом
4. ПК – персональний комп’ютер
5. HTML – мова розмітки гіпертексту
6. CSS – Таблиця каскадних стилів
7. JS – JavaScript
8. MVC — Model View Controll
9. CRUD — 4 основні функції управління даними «створення, читання, оновлення і вилучення»
10. JWT — JSON Web Token

# ВСТУП

Безумовно, тему створення веб-додатків можна вважати актуальною, адже відсоток використання веб-технологій помітно зростає в усіх сферах людського життя.

Використовуючи веб-ресурси, можна суттєво підвищити ефективність виконання буденних задач. Більш того, це найкраща реклама, інформаційний сторінка для будь-якої організації, це важливий крок компанії до розширення кордонів власного бізнесу та здобуття нової аудиторії.

Метою курсової роботи є дослідження особливостей проектування та реалізації cистеми адміністрування веб-сайтом, а також створення інтерфейсу та бази даних для взаємодії з ним.

Об‘єктом дослідження є методи та засоби розробки CMS за визначеними предметними областями, а також прогнозування поведінки користувача.

Предметом дослідження є використання веб-технологій для забезпечення інформаційних потреб предметної області.

# РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

* 1. Аналіз задачі, засобів та методів її вирішення

Створення веб сайту з системою адміністрування - це комплексна задача, що потребує відповідального ставлення та чіткого визначення задач, їх пріоритетності та порядок виконання, для забезпечення чіткого функціонування сайти, можливість підтримувати його працездатність та управління його контентом.

Для реалізації цього програмного продукту, перед нами стоять задачі:

1. Спроектувати та заповнити базу даних.

Для того, щоб досягти максимальної організованості структури даних, їх надійного збереження та полегшеного доступу для зміни і обробки взаємозалежної інформації великих обсягів, ми створимо БД та заповнимо її тестовими значеннями.

Для цього оберемо СУБД MongoDB — документо-орієнтована системe керування, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними і зручними у формуванні запитів.

1. Реалізація CRUD операцій для роботи з даними

Для втілення можливості виконання над даними чотирьох основних операцій (створення, читання, оновлення і вилучення), реалізуємо власну CMS систему.

1. Створення інтерфейсу

Завдяки MVS – шаблону (поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування) реалізуємо можливість переглядати коротку інформацію про всі тури, їх детальну сторінку, особистий кабінет користувача з можливістю перегляду всіх коментарів, що він залишив та бронювання, система адміністрування.

Для відділення даних представлення від виконуваного коду, будемо використовувати шаблонізатор PUG.

1. Забезпечення безпеки користувача

Для реалізації максимальної безпеки даних користувача, право редагувати, переглядати та змінювати данні необхідно надати лише адміністратору, передбачити шифрування паролю перед збереженням його у БД, використовувати JWT (стандарт токена доступу на основі JSON, який містить інформацію щодо безпеки сеансу та ідентифікує користувача, групу користувачів та їх привілеї), обмеження кількості запитів від одного і того самого API та інші заходи.

* 1. Аналіз існуючого програмного забезпечення за тематикою курсового проекту

Перед початком роботи над веб-додатком, ми проаналізували та визначили переваги та недоліки інших схожих сервісів. Оцінювання ми проводили за такими критеріями, як: різноманітність даних, інтерфейс, адаптивність, швидкість роботи.

TUI

TUI - це провідна туристична компанія, штаб-квартира якої знаходиться в Німеччині. На сайті туроператора є можливість відвідати 3D-екскурсію по обраному готелю і навіть побачити обраний номер в об'ємному зображенні, що дозволяє туристу отримати максимально точну інформацію і ще до приїзду в готель мати уявлення про те, що його чекає.

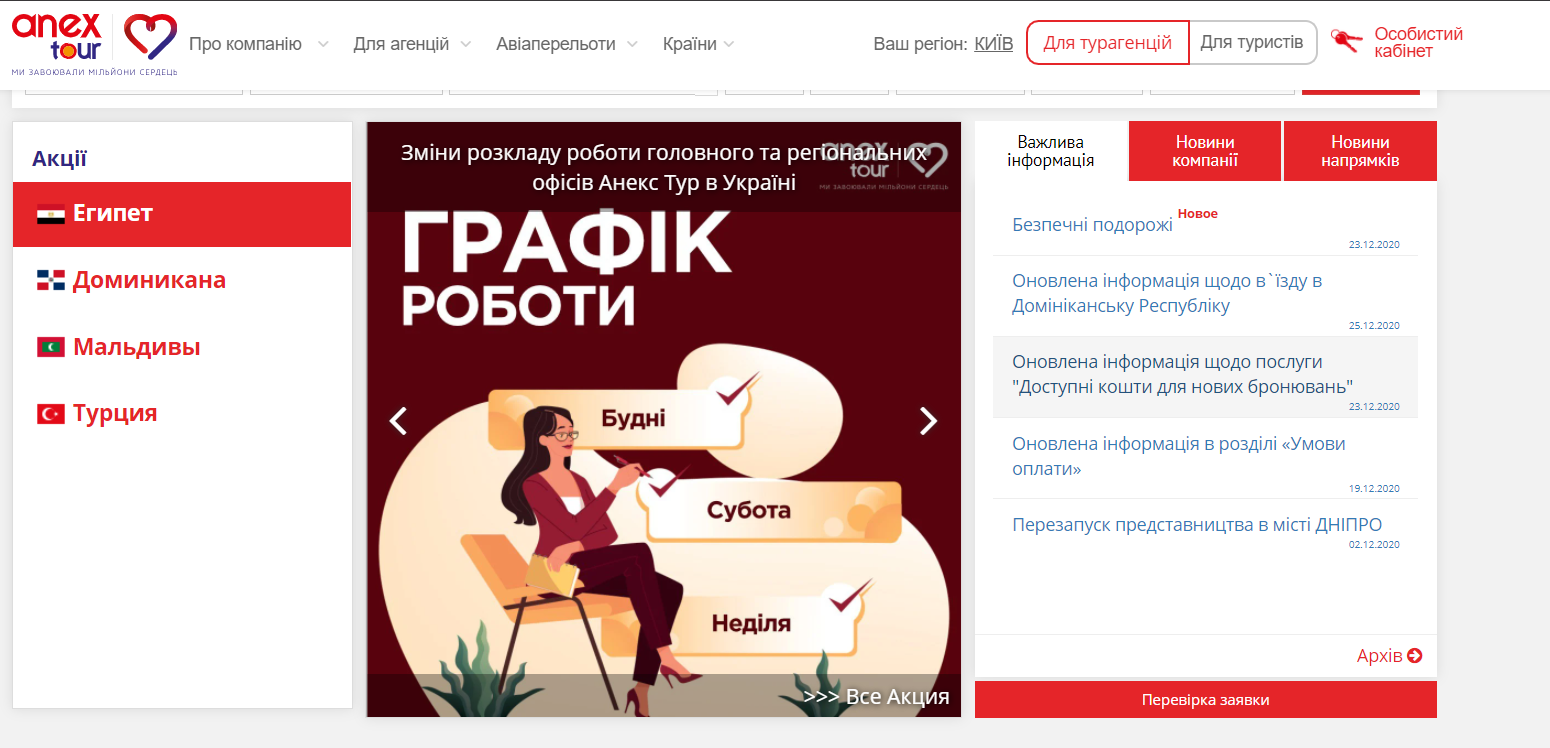
Рис.1.1. Веб-сторінка компанії TUI

До безумовних переваг даного сервісу є великий вибір туристичних напрямків в сфері пляжного відпочинку, функція 3D-екскурсії та, незважаючи на велику кількість інформації про тур та завдяки високому рівні адаптивності, мобільна версія сайту виглядає значно зручніше, ніж ПК-версія.

Недоліками є невелика кількість пропозицій для піших екскурсій та екстремальних подорожей, перенавантаження даними про подорожі, довге завантаження сторінок.

Anex tour

Дана тур агенція організує відпочинки від економ до бізнес класу в усіх напрямках. Їх асортимент готельної бази та видів відпочинку постійно оновлюється. Стверджують, що їх сайт це справжній помічник для тур агентів, адже у ньому впроваджена можливість моніторингу за кількістю вільних місць на рейсах, опис готелів, новинки, акції тощо.

Рис.1.2. Веб-сторінка компанії Anex tour

Плюсами даного веб сайту є широкий вибір можливих функцій, швидка робота сервісу приємна кольорова гамма сайту, цілодобова підтримка туристів по «гарячій лінії».

До мінусів цього сервісу є відсутність актуальної інформації, погана адаптивність сайту для мобільний пристроїв, не дружелюбний для користувача інтерфейс, незручна навігація, погана локалізація, вузько направлений вибір відпочинків.

Join UP!

Join UP! — багатопрофільний український туроператор з головним офісом у Києві. Станом на початок 2018 року компанія має: 40 напрямків; офіси в 6-ти містах; є одним із найбільших замовників чартерних рейсів. Компанія є найбільшим туроператором України та станом на 2018 рік займає 30 % ринку виїзного туризму держави.

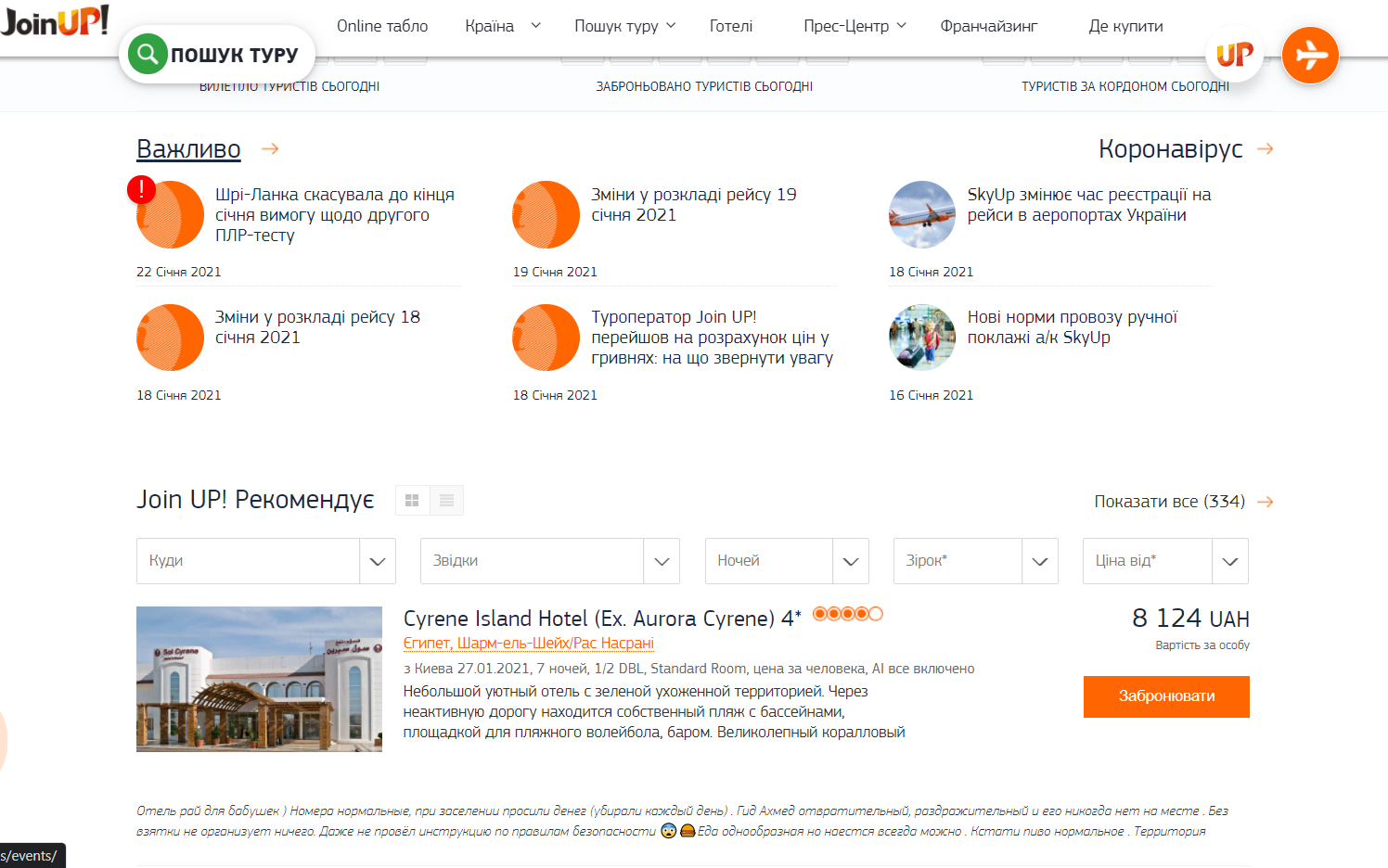


Рис.1.3. Веб-сторінка компанії Anex tour

Даний веб сайт має приємний дизайн, завдяки асинхронному підвантаженню даних, довге завантаження сторінки не таке помітне, широкий вибір послуг наявність повноцінного веб застосунку.

Негативними сторонами цього тур агентства є не дружелюбна адаптація сервісу, довге завантаження деяких елементів сайту, не актуальні данні.

Обравши три різних сайтів туристичних компаній, ми склали таблицю переваг і недоліків кожного з них (Таб. 1.1), відщтовхуючись від загальних критеріїв оцінювання.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | інтерфейс | різноманітність даних | адаптивність | швидкість роботи |
| TUI | - | - | + | - |
| Anex tour | + | - | - | + |
| Join UP! | + | + | - | - |

Таб.1.1. Порвіняння переваг та недоліків

Отже, проаналізувавши декілька схожих веб-додатків, можемо дійти висновку, що найбільш актуальним напрямком розробки буде створення продукту, що поєднуватиме у собі зручний та зрозумілий інтерфейс, буде здатний працювати з великою кількістю різноманітних даних, матиме широкий функціонал, відображатиметься коректо на всіх пристроях та буде швидкий у роботі.

1.3 Технічне завдання на курсову роботу

1. Загальне положення

1.1. Найменування програмного засобу

Повне найменування програмної системи: «Розробка веб-додатку для туристичної фірми Traveler» (надалі "веб-додаток"). Коротка назва програмної системи - "Туристичної фірма".

1.2. Призначення розробки та область застосування

Веб-сайт "Туристичної фірма" призначений для розміщення каталогу популярних коміксів, їх пошуку, та продажу.

Також існує система перегляду найпопулярніших коміксів.

Система може бути впроваджена у роботу інтернет-магазину.

Веб-сайт "Туристична фірма Traveler " дозволить швидко переглядати інформацію про тури, та забезпечить керівництву підприємства своєчасною та повною інформацією про актуальний стан користувачів, їх бронювання та відгуків, а також надасть користувачам можливість вибору різноманітних турів в залежності від їх смаку та думки інших користувачів.

1.4. Найменування розробника та замовника.

Розробник даного продукту - студент групи ІПЗ-19-1 Маковська Ольга-Марія Юріївна (надалі "розробник").

Замовник програмного продукту – кафедра інженерія програмного забезпечення Житомирського державного технологічного університету в межах виконання курсової з дисципліни «Інтернет-програмування» Морозов Андрій Васильович.

2. Підстава для розробки

2.1. Документ на підставі якого ведеться розробка

Робота ведеться на підставі навчального плану за напрямом 121 «Інженерія програмного забезпечення».

3. Вимоги до програми

3.1. Вимоги до функціональних характеристик.

3.1.1. Загальні вимоги

Веб-додаток має забезпечувати:

* можливість дистанційної роботи з робочих станцій локальної та глобальної мережі підприємства;
* постійний доступ користувачів веб-додатку;
* оформлення замовлення;
* організацію управління сайтом;
* можливість доступ до бази даних;

3.1.1. Склад виконуваних функцій

Розробити сайт для турагенції, що підтримує виконання наступних операцій:

1. Відображення повної та короткої інформації про тур

2. Зміна інформації про тур, додавання та видалення туру

3. Вхід до профілю користувача, його зміна, реєстрація нового

4. Можливість забронювати тур та залишити відгук

5. Перегляд користувачем його замовлень та коментарів

3.1.2. Організація вхідних і вихідних даних

Вхідними даними є інформація про тур (назва, ціна, складність, обсяг групи, рейтинг, точка відправки, всі локації, гіди).

Організація вхідних і вихідних даних повинна відповідати інформаційній структурі виконуваних з операцій, вхідним та вихідним паперовим документами.

Введення оперативних даних повинно виконуватися з використанням діалогових екранних форм, побудованих на основі візуальних компонентів. Введення даних виконується на основі затверджених форм документів: анкета, заява, інформаційна довідка.

3.1.3. Часові характеристики і розмір пам'яті, необхідної для роботи

програми.

Час реакції програми на дії користувача (маніпуляції з пристроями введення даних) не повинен перевищувати 0,25 с.

Час виконання команд меню не більше 1 с.

Відображення масивів даних за запитами не більше 3 хвилин.

Доступність БД – 90% цілодобово.

Операції з’єднання з БД не більше 1 хвилини.

Обсяг оперативної пам'яті, необхідний для роботи програми не менше 1Гб.

Дисковій простір, необхідний для збереження програми і файлів даних не більше 300 Мбат для робочої станції та 20 ГБайт..

Інсталяційний пакет програми, що містить у складі БД не повинні перевищувати 100 Мбайт.

3.2. Вимоги до методів рішення і мов програмування

Вибір методів рішення здійснюється розробникам без узгодження з замовником.

3.2.1. Вимоги до системи програмних засобів:

Node.JS, Express, MongoDB, PUG, HTML, CSS, JS.

Вимоги до програмного забезпечення робочої станції:

VS Code

Висновки до розділу

При виконанні даного розділу курсової роботи було проаналізовано поставлене завдання. На основі первинного розгляду поставленої перед нами задачі ми визначили напрямок подальшої розробки веб-застосунку.

Також ми виконали порівняльний аналіз програмних веб-додатків схожого призначення та визначили їх переваги та недоліки, обрали найбільш необхідні функції для подальшої роботи.

Результатом проведеного аналізу є розробка технічного завдання – переліку основних функцій та інтерфейсних особливостей вебсайту, визначені проблеми та знайдені методи для їх вирішення. Вивчено нові теоретичні відомості, необхідні для виконання поставлених перед нами задач.

# РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Проектування загального алгоритму роботи програми

Після вивчення деякої інформації про побудову сайтів, визначення як працює MVC-шаблон, дослідження галузі туризму та визначення необхідного для повноціної роботи веб-застосунку туристичної фірми функціоналу та аналізу схожих програмних додатків, ми почали проектувати основний функціонал нашого веб-забезпечення.

Ми визначили загальний алгоритм роботи користувача з нашим сайтом на основі роботи будь-якого динамічного сайту (Рис. 2.1).

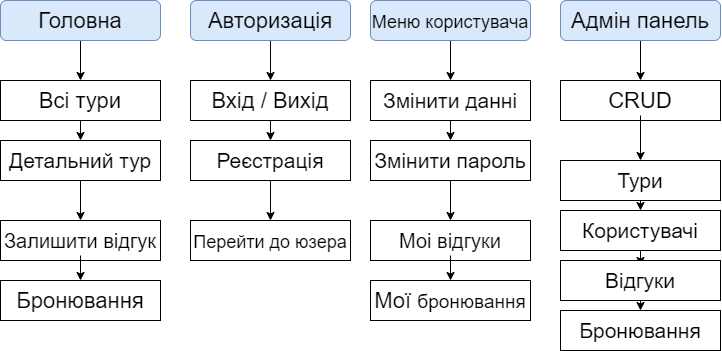


Рис.2.1. Основний функціонал

2.2 Розробка функціональних алгоритмів роботи програми

Після проектування загального алгоритму програми ми розбили завдання на підпункти та розробили функціональні алгоритми роботи. За допомогою чого, ми можемо сформувати точне бачення необхідного функціоналу та досягти поставлених задач.

Починаємо створювати динамічний веб-сайт, у якого сторінки формуються на стороні сервера динамічно, за запитом, зі сторінок-шаблонів та окремо зберігаємого вмісту (інформація, скрипти та ін.). Як правило, для відображення будь-якої кількості різних сторінок використовується одна сторінка-шаблон, у яку підгружається відповідний вміст, ще дозволяє виводити різні данні, редагуючи лише один шаблон.

Розглянемо алгоритм детальніше (Додаток А). Сервер отримує запрошення від користувача і запускає обробку файлу-скрипта інтерпретатором. На цій стадії відбувається вибір необхідної інформації з бази даних та доповнення кожної сторінки, після чого готова сторінка передається користувачу.

На клієнтській стороні використовується JavaScript, який може бути використаний як для мінімальних операцій (наприклад, перевірка правильності задання паролів при реєстрації на сайті), так і глобальних послідовностей.

Для реалізації роботи з даними ми будемо використовувати MVC схему, таким чином ми зможемо досягти того, що модифікація кожного компонента може здійснюватися незалежно. Розглянемо детальніше алгоритм роботи (Додаток Б).

Контролер забезпечує «зв'язок» між користувачем та системою, слідкує та направляє дані від користувача до системи та навпаки. Використовує модель і представлення для реалізації необхідних дій.

Представлення відповідає за отримання необхідних даних із моделей та надсилає їх користувачу, не обробляє введені дані.

Модель надає дані та методи роботи з ними: запрошення в базу даних, перевірка коректності. Модель побудована таким чином, щоб відповісти на запрошення, змінивши свій стан.

Додаток використовує JWT для перевірки аутентифікації користувача в такий спосіб:

Спершу користувач заходить на сервер аутентифікації за допомогою аутентифікаційного ключа. Потім сервер створює JWT і відправляє його юзеру. Коли користувач робить запит до API додатка, він додає до нього отриманий раніше JWT.

Коли користувач робить API запит, додаток може перевірити за поданою із запитом JWT чи є користувач тим, за кого себе видає. У цій схемі сервер додатки налаштований так, що зможе перевірити, чи є вхідний JWT саме тим, що був створений сервером аутентифікації.

Перед початком проектування сайту туристичної фірми, ми зібрали та проаналізували всі данні, які необхідні для повноцінної компанії. Далі за допомогою онлайн сервісів ми змоделювали нашу структуру даних (Додаток В), створили основні таблиці та визначили зв’язки між ними.

2.3 Розробка програмного забезпечення

Було створено нове рішення (Додаток Г), яке складається з 8 файлів контролерів для аутентифікації, обробки помилок, загального алгоритму по створенню, зміні, видаленні даних, бронювання, відгуків, турів, користувачів. Також у рішенні містяться моделі, маршрутизатори, додаткові утиліти, файлі стилів, PUG відображення, та інше.

Далі з MongoDB Atlas створили кластер MongoDB. Використовуючи призначений для користувача інтерфейс на основі браузера Atlas, можемо налаштувати його і контролювати продуктивність. Також необхідно забезпечити єдину точку підключення до БД при розгортанні сервера та піключення до порту.

Лістинг:

const DB = process.env.DATABASE.replace(

  '<PASSWORD>',

  process.env.DATABASE\_PASSWORD

);

mongoose.set('useUnifiedTopology', true);

mongoose

  .connect(DB, {

    useNewUrlParser: true,

    useCreateIndex: true,

    useFindAndModify: false,

  })

  .then(() => console.log('Успішне підключення до БД!'));

const port = process.env.PORT || 3000;

const server = app.listen(port, () => {

  console.log(`Веб-додаток працює на порті ${port}...`);

});

Далі переходимо до створення необхідних моделей, а саме тур, користувач (Додаток Д), бронювання, відгуки та передбачаємо деяку дефолтні перевірки чи обмеження, яку будуть спрацьовувати перед пошуком, збереженням чи виведенням даних. Наприклад перевірка рівності пароля для підтвердження та дефолтного пароля.

Лістинг:

userSchema.methods.correctPassword = async function (

  candidatePassword,

  userPassword

) {

  return await bcrypt.compare(candidatePassword, userPassword);

};

Наступним нашим кроком буде створення функцій для взаємодії з модулем завдяки контролерам. Для користувача ми будемо використовувати - JSON об'єкт, який вважається одним з безпечних способів передачі інформації між двома учасниками. Для його створення необхідно визначити заголовок (header) із загальною інформацією по токену, корисні дані (payload), такі як id користувача, його роль і т.д. і підписи (signature).

А тому у контролері аутентифікації реалізуємо створення даного ключа, який пізніше ми будемо передати через кукі.

Лістинг:

const createSendToken = (user, statusCode, res) => {

  const token = signToken(user.\_id);

  const cookieOptions = {

    expires: new Date(

      Date.now() + process.env.JWT\_COOKIE\_EXPIRES\_IN \* 24 \* 60 \* 60 \* 1000

    ),

    httpOnly: true,

  };

  if (process.env.NODE\_ENV === 'production') cookieOptions.secure = true;

  res.cookie('jwt', token, cookieOptions);

  user.password = undefined;

  res.status(statusCode).json({

    status: 'success',

    token,

    data: {

      user,

    },

  });

};

Також реалізуємо основні функції по видаленню, оновленню, отриманню одного та всіх, створенню запису у handlerFactory, які ми надалі будемо використовувати, як основні заходи по роботі з даними, видозмінюючи їх лише у деяких випадках.

Наприклад при оновленні даних користувача, ми перевіряємо, чи був надіслан пароль для зміни, та попереджаємо, що дана функція відповідає за зміну лише ім’я, поштової скриньки та фото. З іншого боку, зміна паролю реалізована у контролері аутентифікації.

Лістинг:

exports.updateMe = catchAsync(async (req, res, next) => {

  if (req.body.password || req.body.passwordConfirm) {

    return next(new AppError('Цей маршрутизатон не відповідає за зміну паролю. Використовуйе /updateMyPassword.', 400 ) );

  }

  const filteredBody = filterObj(req.body, 'name', 'email');

  if (req.file) filteredBody.photo = req.file.filename;

  const updatedUser = await User.findByIdAndUpdate(req.user.id, filteredBody, {

    new: true,

    runValidators: true,

  });

});

Паралельно під час роботи з контролерами створюємо маршрутизатори, завдяки яким, ми матимемо змогу по запиту виконати необхідні дії.

Лістинг:

router.post('/signup', authController.signup);

router.post('/login', authController.login);

router.get('/logout', authController.logout);

router.patch('/updateMyPassword', authController.updatePassword);

router.get('/me', userController.getMe, userController.getUser);

router.patch(

  '/updateMe',

  userController.uploadUserPhoto,

  userController.resizeUserPhoto,

  userController.updateMe

);

І нарешті переходимо до реалізації вигляду. Спочатку, за допомогою шаблонізатору PUG, створюємо сторінку-основу, в яку ми будемо підвантажувати всі інші елементи сторінки в залежності від запиту користувача.

Упродовж всіх сторінок нас будуть супроводжувати шапка та підвал сайту, також не менш важливим є побічне меню, яке буде з’являтися при переході до інформації про користувача та буде залишатися з юзером при роботі з елементами користувацького меню та адміністративної панелі.

Лістинг:

doctype html

html

  head

    block head

      meta(charset='UTF-8')

      meta(name='viewport' content='width=device-width, initial-scale=1.0')

      link(rel='stylesheet' href='/css/styles.css')

      link(rel='shortcut icon' type='image/png' href='/img/favicon.png')

      title Travel | #{title}

  body

    // HEADER

    include \_header

    // CONTENT

    block content

      h1 This is a placeholder heading

    // FOOTER

    include \_footer

Також у нас буде окремі фали для відображення одного туру, відгуку, детальної інформації про тур, сторінка помилки, вхід або реєстрація користувача, його акаунт, форми для створення та зміни даних, таблиці для відображення віх даних.

Наступним кроком буде створення файлу index.js, через який буде реалізована взаємодія з кнопками та посиланнями для виконання певних алгоритмів, перенаправлення на інші сторінки, або відправка даних.

Висновки до розділу

При виконанні даного розділу курсової роботи було спроектовано загальні алгоритми роботи програмного додатку, розроблені функціональні алгоритми та спроектоване повне програмне забезпечення.

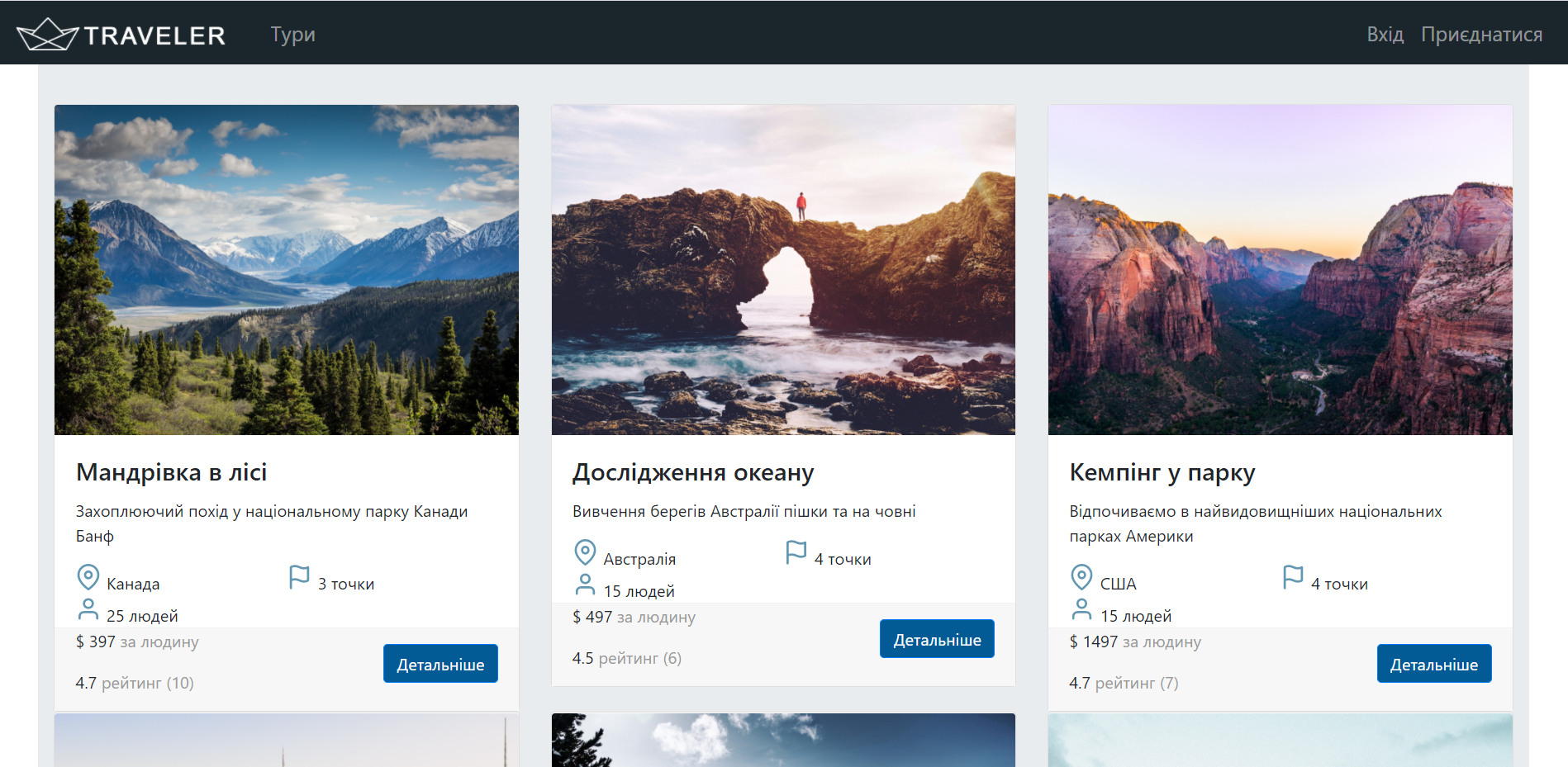
Також ми створили простий та лаконічний інтерфейс, для взаємодії користувача з основними реалізованими методами та управління БД.

Результатом проведеного аналізу є розробка веб-сайту.

# РОЗДІЛ 3 ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ

3.1 Опис роботи з програмним додатком

Після того, як користувач перейде на головну сторінку нашого веб-додатку, він бачить коротку інформацію про тури (Рис.3.1), а саме обкладинку, назву, короткий опис, ціну та рейтинг, а також кнопку, що дозволяє перейти на детальну сторінку про обраний тур.

Рис.3.1: Головна сторінка

Після того, як він перейде за посиланням до повної інформацій про тур, перед ним завантажтися нова сторінка (Рис.3.2), на якій буде обкладинка обраного туру з його назвою, блок з короткими фактами про мандрівку, секція, де ми можемо побачити туристичних гідів, відповідальних за проведення цієї подорожі, а також повне описання туру, де може бути вказана менш важлива, об’ємна секція з текстом.

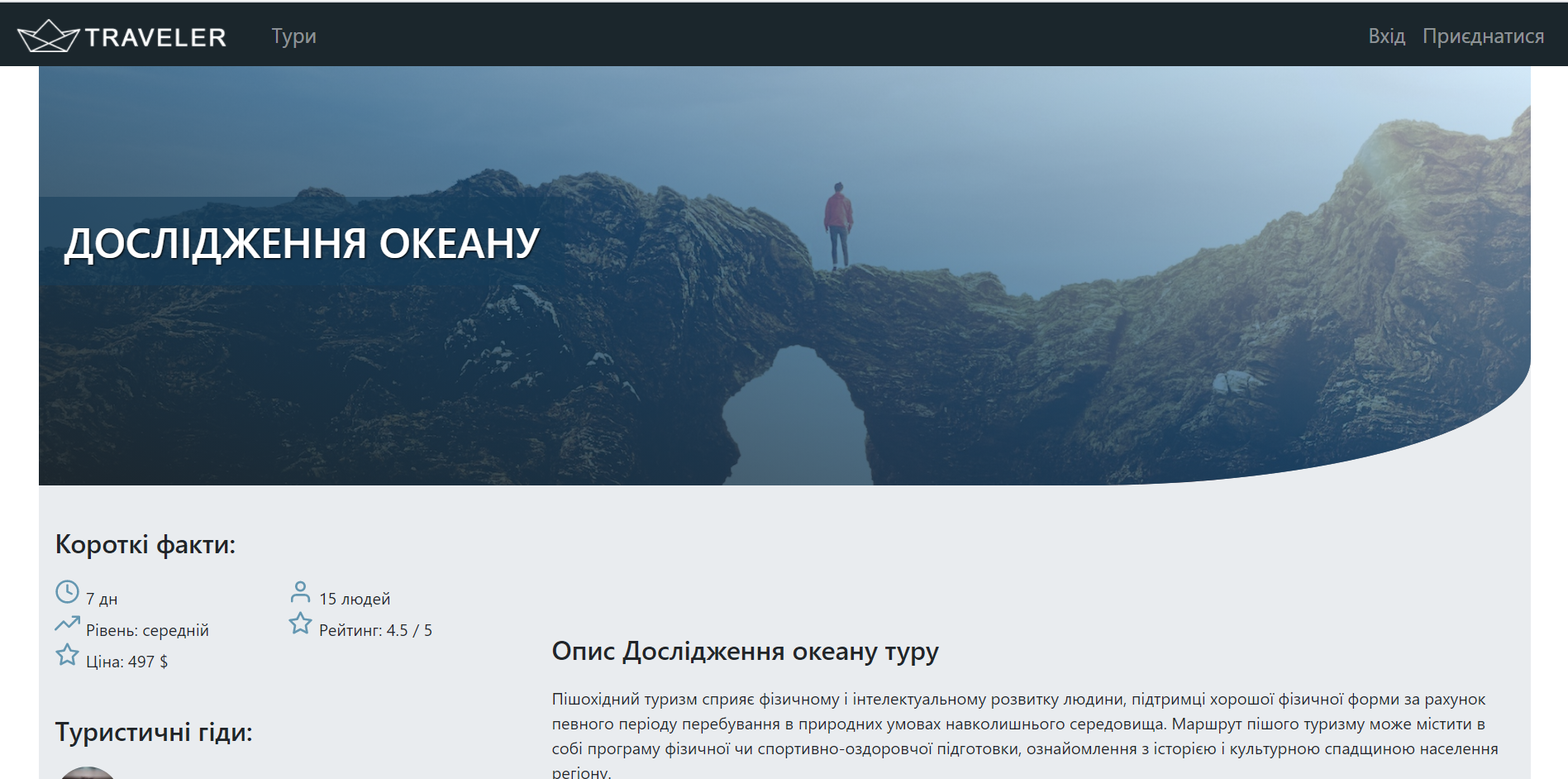


Рис.3.2: Детальна інформація про тур

Далі йде невеликий блок з фотографіями, для того, щоб користувач мав змогу скласти краще враження про тур, а також невелика карта (Рис.3.3), де вказану точки, які впродовж мандрівки необхідно відвідати та короткий опис над кожним із них (день і місто).



Рис.3.3: Секція карти у детальному огляді тура

Одразу після карти, користувач бачить раніше залишень відгуки від наших клієнтів (Рис.3.4), де вказане ім’я, фотографія автора, текст відгуку та оцінка туру, що відображається у кількості підсвічених зірочок.

Також клієнт може залишити одне власне враження про тур, обравши із випадаючого списку оцінку та написавши короткий текст у необхідне поле. Ще далі йде блок з можливістю забронювати обраний тур.

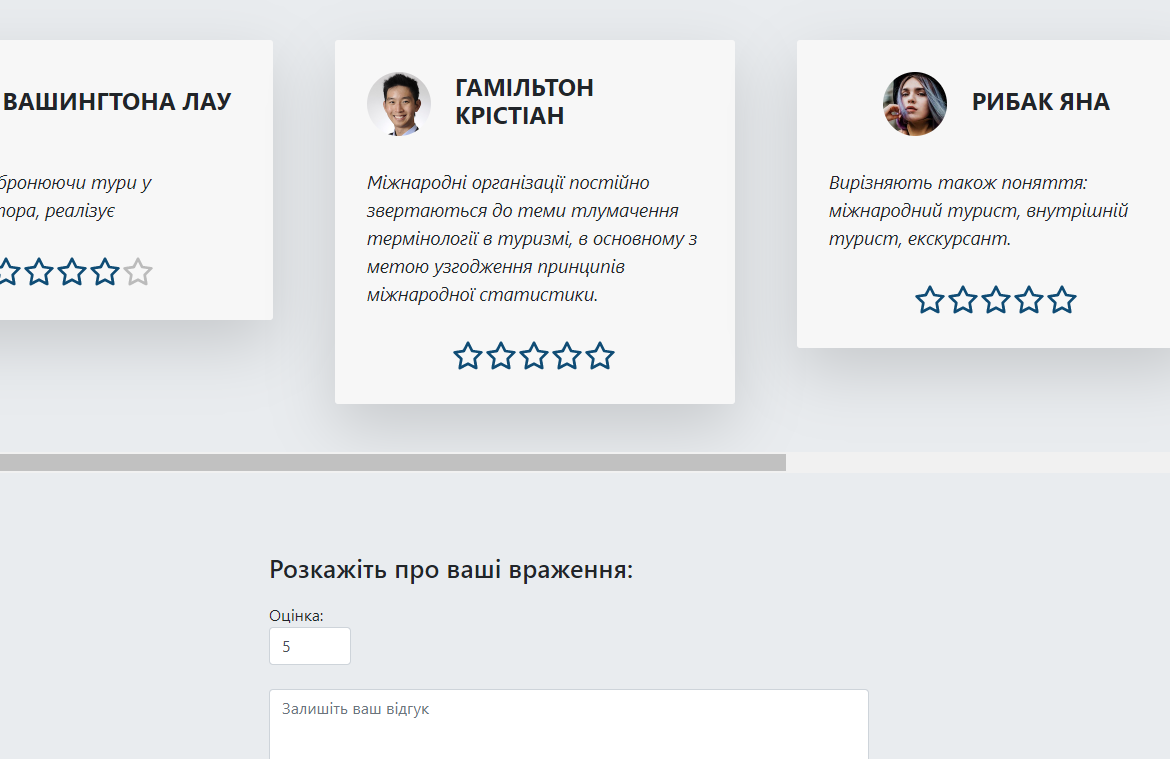


Рис.3.4: Секція відгуків у детальному огляді тура

Завдяки хедеру користувач може повернутися на головну сторінку, увійти в систему (Рис.3.5), зареєструватися, вийти або перейти до налаштування опубліковано запису, де він може змінити данні чи пароль.

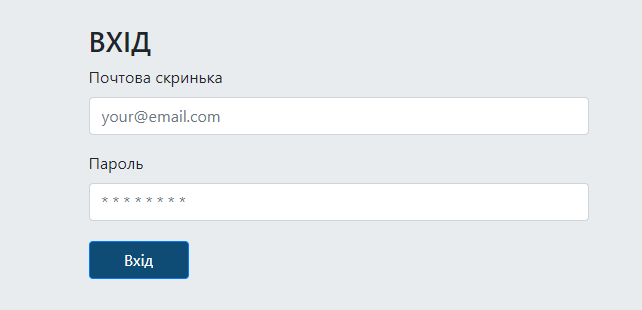


Рис.3.5: Вхід до системи

Завдяки сторонньому меню, користувач має змогу переглянути власні бронювання та відгуки, змінити інформацію у них, або видалити. За схожим принципом працює адміністративна панель, де доступна інформація про всі тури, користувачів, відгуки та бронювання з CRUD можливостями (Рис.3.6).



Рис.3.6: Всі тури

3.2 Тестування веб-додатку

Для зручного користуванням програмним продуктом та захисту БД та підтримки її коректної роботи ми передбачили помилки, які може зробити користувач.

Наприклад, якщо неавторизований користувач спробує забронювати тур, або відправити відгук, замість кнопки підтвердження у нього буде посилання на сторінку для входу (Рис.3.7).

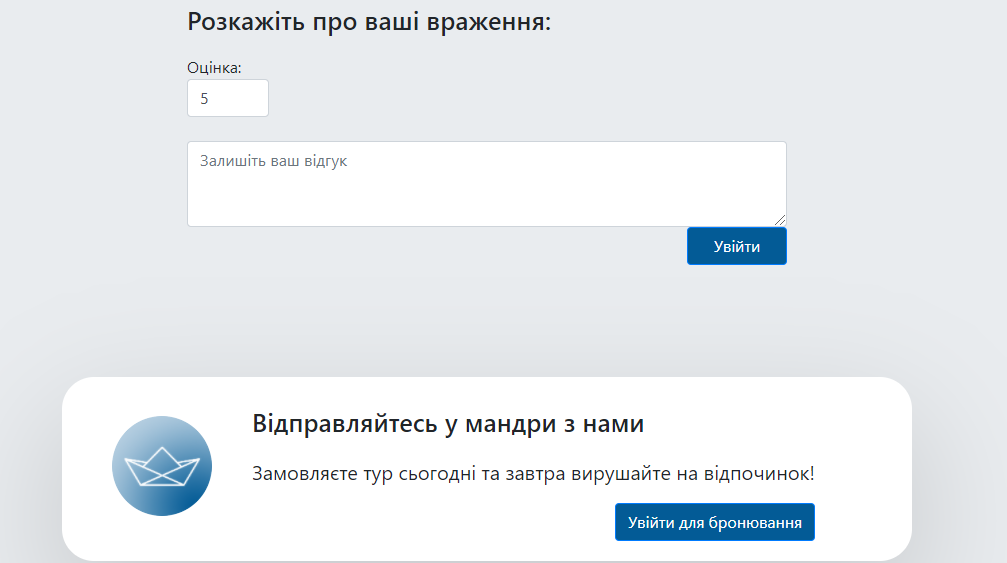


Рис.3.7: Якщо користувач неавторизований

Також, у формах, на всіх необхідних та обов’язкових полях ми передбачили валідацію даних, виведення помилки та примітку required, що не дозволить користувачу відправити формі не заповнивши поля (Рис.3.8).

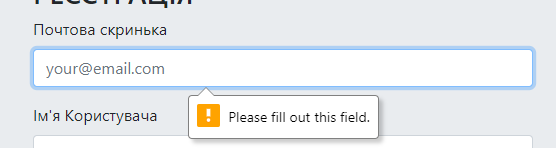


Рис.3.8: Обов’язкові поля

Також, завдяки використанню JWT, ми попереджаємо можливість простого користувача потрапити у адміністративну панель, де він зможе управляти всіма доступними даними.

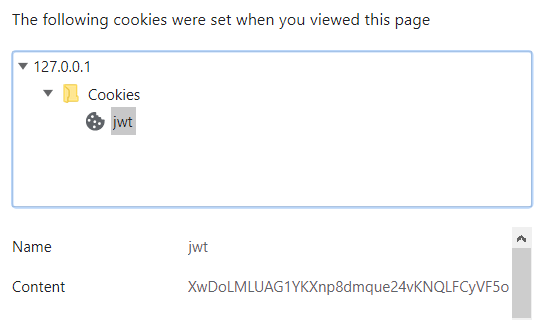


Рис.3.9: JWT

Висновки до розділу

При виконанні даного розділу ми прагнули проаналізувати поведінку користувача та максимально просто розглянути створений раніше інтерфейс, переосмислити його, внести деякі корегуючі зміни та пояснити алгоритм користування.

Також ми намагалися віднайти можливі слабкі сторони нашого програмного продукту, перевірити його на коректну роботу та проаналізувати можливі неправильні дії, які користувач може здійснити.

# ВИСНОВКИ

Під час виконання даної роботи було проаналізовані теоретичні засади проектування та реалізації CMS та існуючі веб-сайти на обрану тематику, визначена їх унікальність.

Створений адаптивний інтерфейсу веб-сайту за допомогою HTML, CSS, JavaScript, реалізовані математичні та алгоритмічі моделі функціонування системи на основі БД, спроектована БД за визначеною предметною областю та реалізована CMS.

У другому розділі було описано та спроектовано загальний алгоритм роботи веб додатку, за яким і було розроблено продукт.

У третьому розділі ми детально описали всі можливі дії, для полегшеного сприйняття користувача, а також проаналізували можливі помилки.

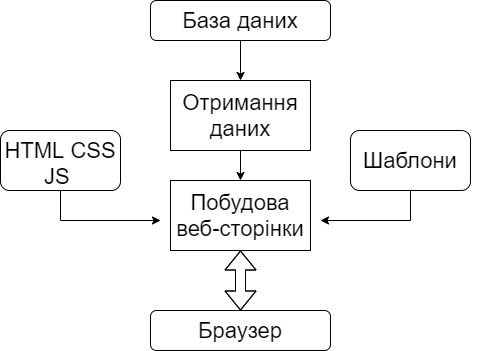
В результаті курсового проекту отримано веб застосунок, який відповідає функціональним вимогам поставленим у проекті.

# ВИКОРИСТАНІ ДЖЕРЕЛА

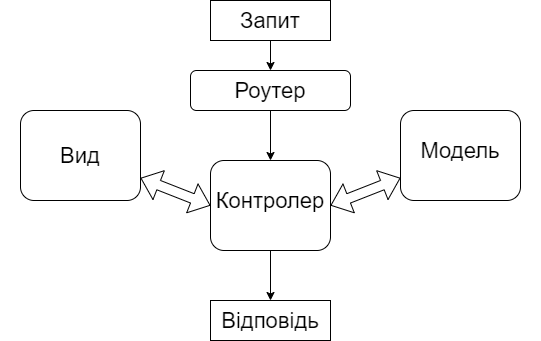
* + - 1. Trott. Node.js Documentation [Електронний ресурс] / Trott – Режим доступу до ресурсу: <https://nodejs.org/docs/latest-v13.x/api/>.
      2. Express Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://expressjs.com/ru/guide/routing.html>.
      3. MongooseJs Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://mongoosejs.com/docs/guide.html>.
      4. П'ять простих кроків для розуміння JSON Web Tokens [Електронний ресурс] / Борис Южаков – Режим доступу до ресурсу: <https://habr.com/ru/post/340146/>.
      5. Стив Розенберг. Веб-розробка із застосуванням Node і Express. Повноцінне використання стека JavaScript / Стив Розенберг., 2016.
      6. Бред Дейлі. Розробка веб-додатків за допомогою Node.js, MongoDB і: вичерпне керівництво / Бред Дейлі, Калеб Дейлі, Брендан Дейлі., 2020.
      7. Марейн Хавербеке. Виразний JavaScript. Сучасне веб-програмування / Марейн Хавербеке. – Пітер, 2019.
      8. Дуглас Крокфорд. JavaScript: сильні сторони / Дуглас Крокфорд..
      9. Кайл Бэнкер. MongoDB in Action / Кайл Бэнкер..

# ДОДАТКИ

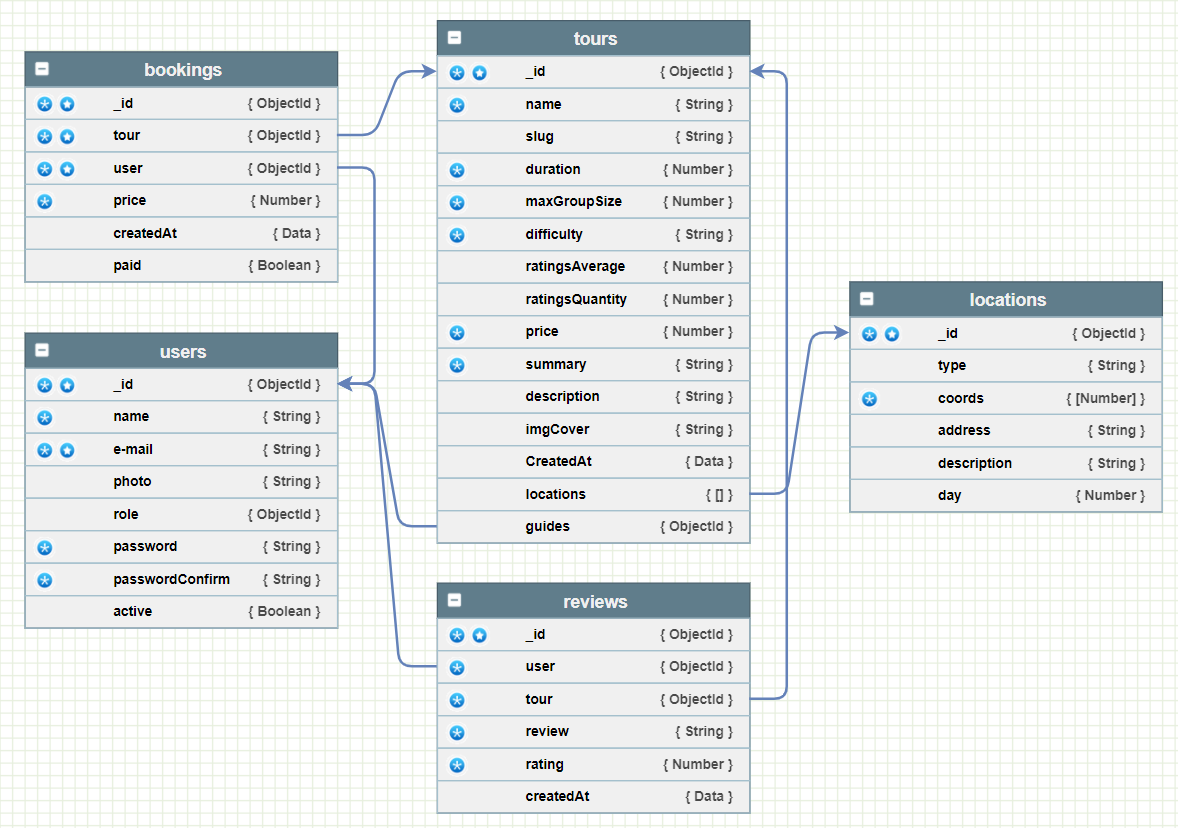
Додаток А



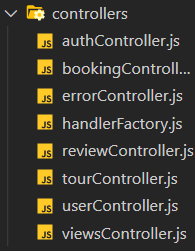
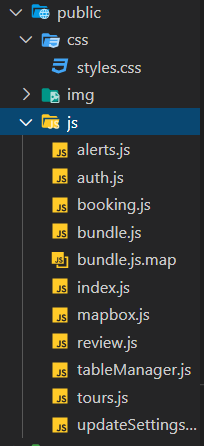
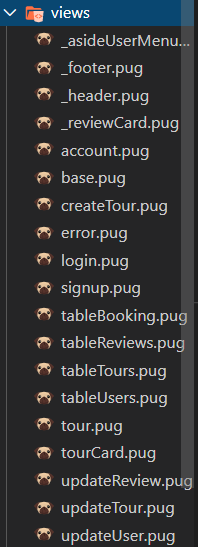
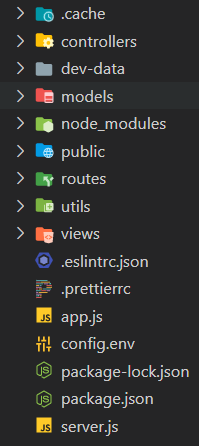
Додаток Б

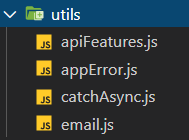
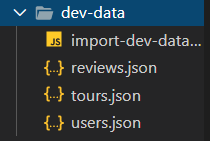
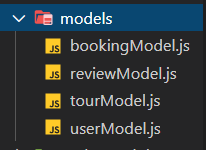
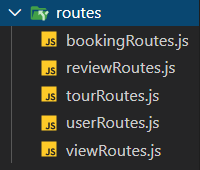


Додаток В



Додаток Г





Додаток Д

const userSchema = new mongoose.Schema({

  name: {

    type: String,

    required: [true, 'Введіть ваше ім\*я'],

  },

  email: {

    type: String,

    required: [true, 'Введіть адресу вашої поштової скриньки'],

    unique: true,

    lowercase: true,

    validate: [validator.isEmail, 'Не привильний формат!'],

  },

  photo: {

    type: String,

    default: 'default.jpg',

  },

  role: {

    type: String,

    enum: ['user', 'guide', 'lead-guide', 'admin'],

    default: 'user',

  },

  password: {

    type: String,

    required: [true, 'Вигадаєте пароль не менше, ніж на 8 симврлів!'],

    minlength: 8,

    select: false,

  },

  passwordConfirm: {

    type: String,

    required: [true, 'Підтвердіть ваш пароль!'],

    validate: {

      validator: function (el) {

        return el === this.password;

      },

      message: 'Паролі не співпадають!',

    },

  },

  passwordChangedAt: Date,

  active: {

    type: Boolean,

    default: true,

    select: false,

  },

});