

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Додаток планування розваг та меню готелів»

Виконав студент групи ІПЗ-19-3
ШВЕД Оксана Миколаївна

Керівник роботи:
КРАВЧЕНКО Світлана Миколаївна

Рецензент:
ЧИЖМОТРИЯ Олексій Володимирович

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії
програмного забезпечення
Андрій МОРОЗОВ
«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ШВЕД Оксана Миколаївна**

Керівник роботи: **КРАВЧЕНКО Світлана Миколаївна**

Тема роботи: **«Додаток планування розваг та меню готелів»**

затверджена Наказом закладу вищої освіти від «14» лютого 2023 р., №70с

Вихідні дані для роботи: **методи та засоби розробки веб-додатків, аналіз проблематики додатку планування розваг та меню готелів, використання для розробки веб-додатку вебфреймворку Next.JS та платформу Node.JS.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Кравченко С.М.	16.02.2023р.	16.02.2023р.
2	Кравченко С.М.	16.02.2023р.	16.02.2023р.
3	Кравченко С.М.	16.02.2023р.	16.02.2023р.

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі	16 лютого 2023 – 25 лютого 2023	Виконано
2	Пошук, огляд та аналіз аналогічних розробок	25 лютого 2023 – 4 березня 2023	Виконано
3	Формулювання технічного завдання та опрацювання літературних джерел	4 березня 2023 – 18 березня 2023	Виконано
4	Проектування структури системи	18 березня 2023 – 27 березня 2023	Виконано
5	Написання програмного коду	27 березня 2023 – 15 квітня 2023	Виконано
6	Тестування роботи додатку	15 квітня 2023 – 3 травня 2023	Виконано
7	Попередній захист	3 травня 2023	Виконано

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу додатка планування розваг та меню готелів і пояснювальної записки. Пояснювальна записка до випускної роботи містить 80 сторінок, 56 ілюстрацій та 1 таблицю.

Метою роботи є розробка веб-додатку планування розваг та меню готелів засобами Next.JS та Node.JS, що систематизуватиме в собі інформацію про готелі, їхнє меню та розваги, та надасть змогу зручно переглядати дані, а також записуватися на різні заходи та створювати нагадування.

У роботі визначено основні завдання на розробку системи, також проаналізовано аналоги створеного програмного продукту та обрано засоби реалізації. Проведено розробку структури бази даних, обрано зручний та сучасний стек технологій та розроблено концепцію програмного продукту. Також описана програмна реалізація, її результати тестування.

КЛЮЧОВІ СЛОВА: ВЕБ-ДОДАТОК, ОДНОСТОРІНКОВИЙ ДОДАТОК, ГОТЕЛЬ, ЗАПИС, ЗРУЧНІСТЬ, ВІДПОЧИНОК, NEXT.JS, NODE.JS, GOOGLE CALENDAR.

					ІПЗ.КР.Б – 121 – 23 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Додаток планування розваг та меню готелів Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		О. Швед						
Керівник		С. Кравченко					2	
Н. Контр.		О. Тичина				Житомирська політехніка, група ІПЗ-19-3		
Зав. каф.		А. Морозов						

ABSTRACT

The bachelor's thesis consists of a software complex of an entertainment planning application and a hotel menu and an explanatory note. The explanatory note to the thesis contains 80 pages, 56 illustrations and 1 table.

The purpose of the work is the development of a web application for planning entertainment and hotel menus using Next.JS and Node.JS, which will systematize information about hotels, their menus and entertainment, and provide an opportunity to conveniently view data, as well as record for various events and create reminders.

In the work, the main tasks for the development of the system are defined, analogues of the created software product are also analyzed and means of implementation are selected. The database structure was developed, a convenient and modern technology stack was selected, and the software product concept was developed. The software implementation and its test results are also described.

KEY WORDS: WEB APP, SINGLE PAGE APP, HOTEL, BOOKING, CONVENIENCE, HOLIDAY, NEXT.JS, NODE.JS, GOOGLE CALENDAR.

					ПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						3

ЗМІСТ

РЕФЕРАТ	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	8
ВСТУП	9
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ДОДАТКУ ПЛАНУВАННЯ РОЗВАГ ТА МЕНЮ ГОТЕЛІВ	11
1.1. Постановка задачі	11
1.2 Аналіз аналогів програмного продукту	12
1.3 Визначення архітектури веб-додатку	18
1.3.1 Визначення архітектури серверної частини веб-додатку	19
1.3.2 Визначення структури клієнтської частини веб-додатку	20
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	22
Висновки до розділу 1	24
РОЗДІЛ 2. ПРОЕКТУВАННЯ ДОДАТКУ ПЛАНУВАННЯ РОЗВАГ ТА МЕНЮ ГОТЕЛІВ	26
2.1. Визначення варіантів використання та структури системи	26
2.2. Розробка бази даних	29
2.3 Проектування та реалізація алгоритмів роботи додатку	33
2.4. Реалізація функціоналу додатку	33
2.4.1 Реалізація основної архітектури додатку	33
2.4.2 Реалізація компонента для входу в обліковий запис Google та створення нагадування у Google Calendar	40
2.4.3 Реалізація і налаштування конфігураційного файлу для з'єднання з базою даних	42

2.4.4 Реалізація контролера та створення власної кінцевої точки (endpoint) на сервері	43
Висновки до розділу 2	45
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-ДОДАТКОМ ПЛАНУВАННЯ РОЗВАГ ТА МЕНЮ ГОТЕЛІВ	47
3.1. Порядок встановлення та налаштування параметрів додатку	47
3.2. Структура інтерфейсу веб-додатку та порядок роботи	49
3.2.1. Інтерфейс та порядок роботи з панеллю власника готелів	52
3.2.2. Інтерфейс та порядок роботи з основною частиною додатка	55
3.3. Тестування роботи веб-додатку	68
Висновки до розділу 3	72
ВИСНОВКИ	73
ЛІТЕРАТУРА	75
ДОДАТКИ	78

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

DOM – об'єктна модель документа

SPA – односторінковий застосунок

БД – база даних

Telegram – багатоплатформовий сповісник

Чат-бот – програма, яка імітує справжню розмову з користувачем

Google Calendar – застосунок для тайм-менеджменту, розроблений Google

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						6

ВСТУП

Актуальність теми. Сучасне життя людей зараз тісно пов'язане з веб-технологіями. Зараз веб-сайти – це тісний зв'язок між підприємствами та клієнтами, і обом сторонам цей зв'язок комфортний.

Створення свого власного сайту зараз є обов'язковим для успіху підприємницької діяльності. Адже це забезпечує зручний обмін інформацією з клієнтами. Головним атрибутом того, що компанія сучасна та прогресивна є наявність у них сайту.

Поступово у ресторанах відмовляються від фізичного меню та вводять меню на власні сайти, аби позбавитися від зайвого використання паперу. Тому конкуренція за найзручніше та найгарніше меню збільшується. Тому було б не зайвим додати ще кращий функціонал, який дозволить не лише переглядати меню, але й заздалегідь замовляти його та приходити на час його готовності. А для ресторану було б зручно зробити всі замовлення в одному місці.

У готельному бізнесі основне – це привабити клієнтів, а це можна зробити за допомогою розважальних заходів для різних вікових категорій. Зазвичай клієнти готелів зупиняються в самому готелі тільки для сну, а ось для відпочинку вони віддають перевагу розважальним заходам: екскурсіям, концертам та іншим туристичним розвагам.

Метою кваліфікаційної роботи є проектування та розробка веб-додатку планування розваг та меню готелів, який призначений для спрощення обміну інформацією між готелями та клієнтами, систематизації інформації про розважальні заходи, які будуть проходити від конкретного готелю, зручного встановлення нагадування та замовлення бажаної страви серед меню заздалегідь.

Об'єктом дослідження є розробка, проектування та реалізація веб-додатку планування розваг та меню готелів.

Предметом дослідження є використання веб-орієнтованих технологій задля створення веб-додатку планування розваг та меню готелів.

Для розробки веб-додатку було використано вебфреймворк Next.JS та платформу Node.JS, також для створення швидких і легко керованих API JavaScript було використано платформу Node.JS – Strapi, з об’єктно-реляційною системою керування базами даних – PostgreSQL.

Розроблений веб-додаток може бути використаний як практичний сайт, який адаптований під різні готелі міст України з можливою подальшою розробкою свого функціоналу.

Випускна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 32 найменувань, 1 додатку, 1 таблиці та 56 ілюстрацій. Повний обсяг роботи становить 80 сторінок, з них 77 сторінок основного тексту.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ДОДАТКУ ПЛАНУВАННЯ РОЗВАГ ТА МЕНЮ ГОТЕЛІВ

1.1. Постановка задачі

Зараз практично кожний підприємець має власний сайт, який на зараз є необхідним чинником існування, що дозволяє конкурувати з іншими підприємцями, а також залучати нових клієнтів та отримувати більше розголосу про сайт та компанію.

Основною метою створення додатку планування розваг та меню готелів є спрощення процесу обміну інформації між клієнтом та компанією, а саме, готелями. Реалізація даного веб-додатку може бути корисною для клієнтів готелю в пошуках гарно провести відпочинок та економити свій час.

У кваліфікаційній роботі необхідно створити веб-додаток, який буде корисний для клієнтів готелю та буде мати можливість замовлення страв заздалегідь, а також записуватися на цікаві розважальні заходи і налаштовувати нагадування про них за необхідності.

Для реалізації поставленого завдання, основними етапами є:

1. Визначення вимог та функцій додатку;
2. Проектування дизайну додатку;
 - а) Створення дизайну для основної частини додатку;
 - б) Створення дизайну для адміністративної частини додатку;
 - с) Адаптація дизайну під різні розміри екранів;
3. Проектування та розробка архітектури додатку;
 - а) Розробка архітектури для серверної частини додатку;
 - б) Розробка архітектури для клієнтської частини додатку;
4. Розробка компонентів, сторінок та функціоналу додатку;
 - а) Функціонал зберігання інформації;
 - б) Компонент авторизації та реєстрації користувачів;
 - с) Компонент обробки та збереження зображень;

- d) Сторінка виведення готелів, меню та розваг;
- e) Функціонал замовлення страв з меню та відображення цього у чат-боті Telegram для адміністраторів;
- f) Функціонал для запису на розважальний захід, а також доступ до нагадування через Google Calendar;
- g) Компонент для відгуків;
- h) Компонент фільтрації даних;

5. Тестування роботи додатку.

Результатом реалізації поставленого завдання є додаток планування розваг та меню готелів, який систематизує інформацію про готелі, меню та розважальні заходи, а також робить зручною комунікацію клієнтів з готелями.

1.2 Аналіз аналогів програмного продукту

Було проведено аналіз аналогів програмного продукту серед наявних веб-додатків в Інтернеті та визначено їх переваги та недоліки.

Для користувачів завжди була потреба у зручному пошуку готелів, і разом з тим відвідування цікавих місць для задоволення як голоду, так і розваги. Тому буде зручно зробити це все в одному додатку.

Один з додатків, який є аналогом розробленого додатку, є «tripadvisor.com» [30]. Даний веб-додаток є необмеженим у розташуванні, тобто він не прив'язаний лише до України, переглядати готелі та цікаві місця можна для всього світу. На сайті представлено розбиття тем за категоріями, але, наприклад, фільтрації за типом того, куди можна сходити, немає. Для кожного закладу або місця є відгуки та оцінки (рис. 1.1).

Інформація про ресторан наявна майже повністю, але немає самих страв та їх цін. Є зручним те, що вказано, яка національну кухню готують у відповідному ресторані.

Інтерфейс додатку є зручним та інтуїтивно зрозумілим. Іноді є загромодження інформацією на деяких сторінках. Можна увійти в особистий кабінет. Інтерфейс додатку представлений лише на англійській мові.

Іншим доволі популярним аналогом є веб-додаток «airbnb.com.ua» [31] (рис.1.3). Додаток представляє бронювання номерів у будь-якому куточку світу. Також там представлені сторінки визначних місць. У додатку присутні багато категорій місць, але я не знайшла зручного пошуку по країнам, лише по частинам світу.

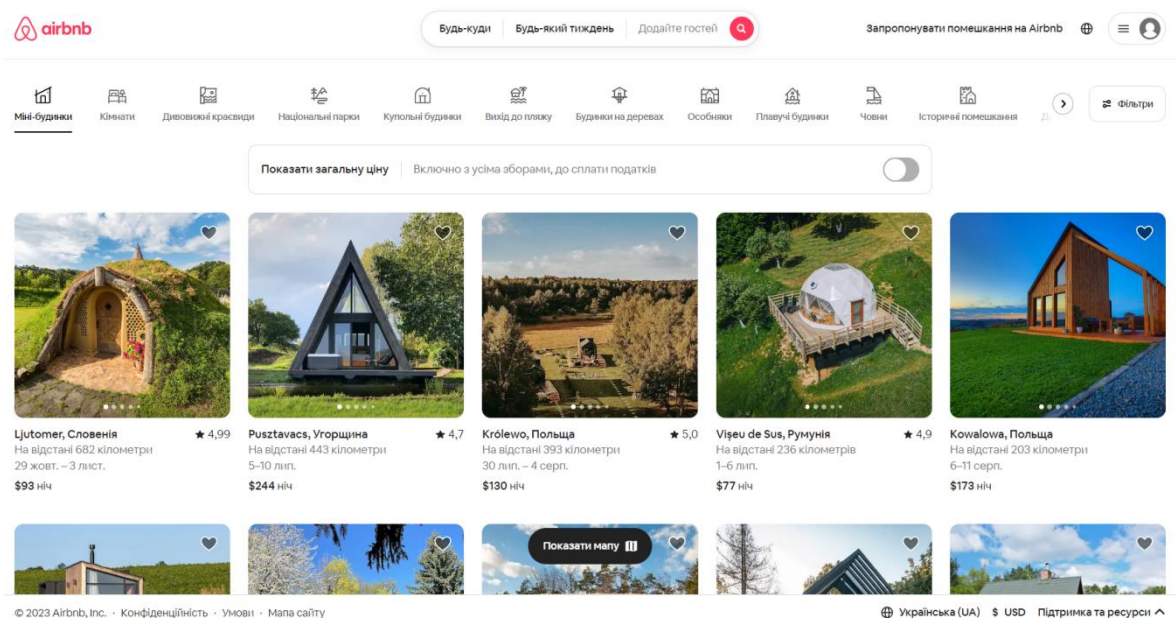


Рисунок 1.3 – Головна сторінка у веб-додатку «airbnb.com.ua»

Є можливість пошуку місць на карті, але, як на мене, це зроблено не дуже зручно, тому що спочатку потрібно шукати категорію місця, яке може бути в Україні, наприклад, і вже потім шукати на карті місто, у якому присутнє те, що шукаєш. Було б зручно зробити просто пошук по місту чи країні, а далі вже вибирати тип місця чи готелю.

Інформації про місце достатньо: опис, розташування, зручності, календар, також є відгуки. Також навіть для готелів є опис про його власника та можливість зв'язатися з ним.

Є можливість увійти в особистий кабінет. В плані локалізації додатку все чудово, є українська мова та багато інших. Також присутні різні вибори валют.

Інтерфейс додатку є досить привабливим та зручним, гарна кольорова палітра та швидкодія сайту не може не тішити. Також додаток є повністю адаптивним та зручний у використанні на мобільному пристрої (рис. 1.4).

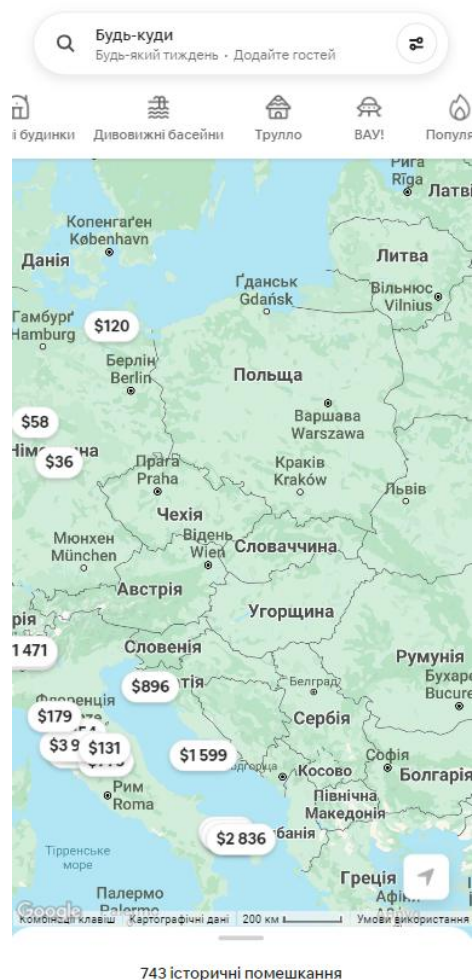


Рисунок 1.4 – Мобільна версія веб-додатку «airbnb.com.ua»

Останнім аналогом є веб-додаток «orentable.com» [32], який призначений для бронювання столиків у ресторанах (рис. 1.5). На жаль, замовити столик в Україні неможливо, тому що у списку доступних країн України немає, тому оберемо Польщу для прикладу.

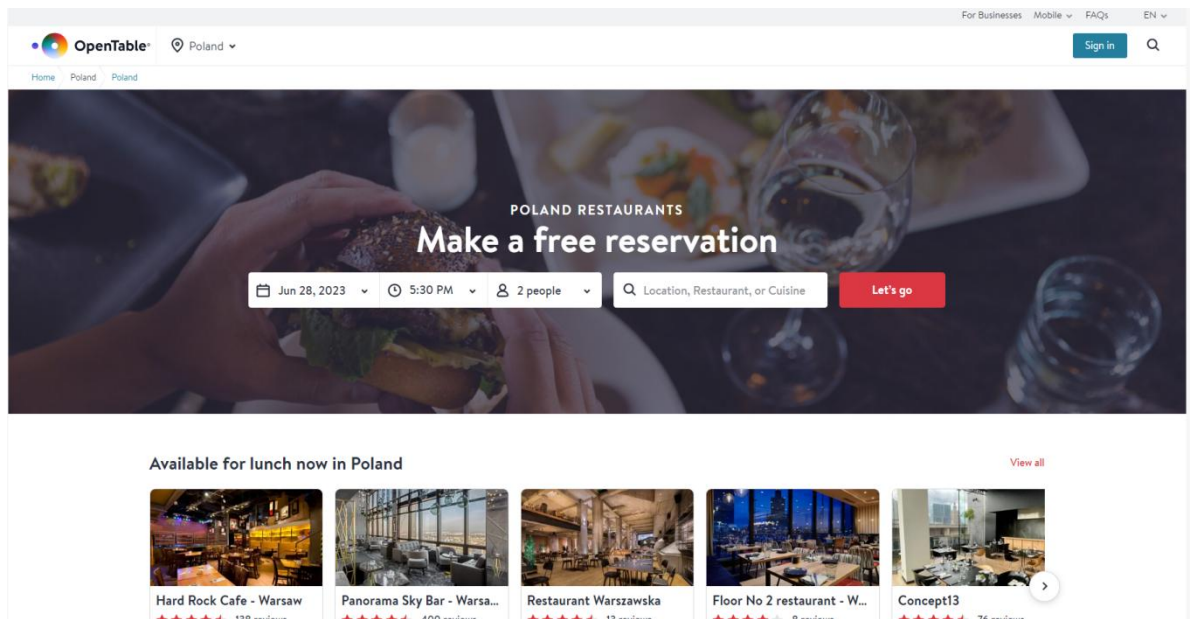


Рисунок 1.5 – Головна сторінка з розташуванням у Польщі в додатку «opentable.com»

Додаток надає можливість пошуку ресторану, а також точне вказання дати та часу бажаного бронювання столику з вказанням кількості людей. Після пошуку можна вільно вибирати бажаний ресторан (рис. 1.6).

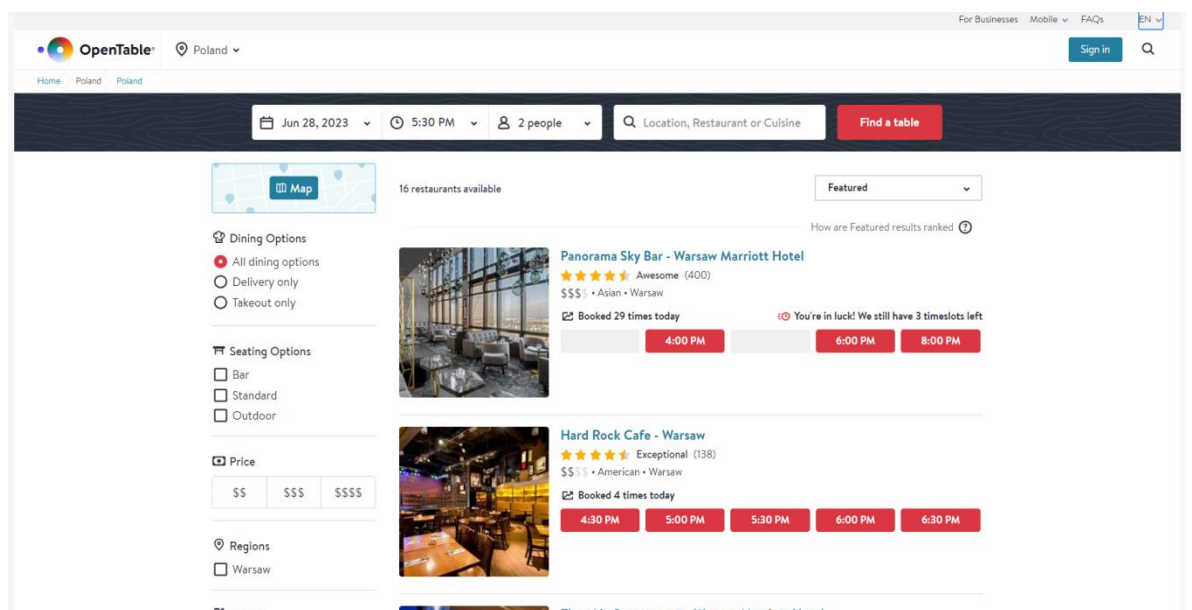


Рисунок 1.6 – Пошук ресторанів у Польщі в додатку «opentable.com»

Є зручна фільтрація з лівого боку екрану, що дозволяє більш точно знайти бажане. Також є вибір ресторану за картою, щоб вибрати найближчий до користувача. При переході на ресторан (рис. 1.7) є вся детальна інформація про ресторан, а також страви, які там можна скуштувати. Відгуки також

присутні на даному сайті. Та саме головне, що можна забронювати столик за часом, який запропоновано.

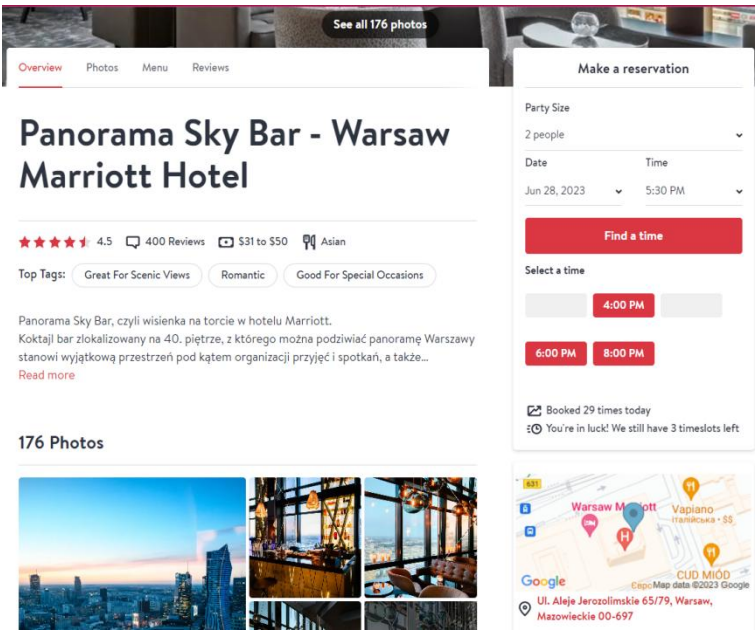


Рисунок 1.7 – Сторінка вибраного ресторану в додатку «opentable.com»

Інтерфейс додатку є зручним та стильним, переклад присутній, але української мови немає. Також сайт є адаптивним як на мобільних пристроях, так і на широких екранах.

Для порівняння вище описаних веб-додатків було створено порівняльну таблицю 1.1.

Таблиця 1.1

Порівняння веб-додатків аналогів

Характеристика	tripadvisor.com	airbnb.com.ua	opentable.com
Перегляд цікавих місць	Зручний, але опису немає	Незручний, тому що складно шукати по містам	Відсутній
Перегляд ресторанів	Зручний	Відсутній	Дуже зручний
Перегляд готелів	Зручний	Зручний	Відсутній
Бронювання номерів готелю	Відсутнє	Зручне	Відсутнє

Бронювання столику в ресторані	Відсутнє	Відсутнє	Зручне
Адаптивність інтерфейсу	Адаптивний	Адаптивний	Адаптивний
Пошук та фільтрація	Є пошук, але немає фільтрації	Зручний пошук та фільтрація	Зручний пошук та фільтрація
Дизайн	Непоганий, але загромаджений	Чудовий	Чудовий

Переглянувши таблицю 1.1, можна дійти до висновку, що кожен з цих додатків виконує лише частину з поставлених задач перед розробленим додатком. Додаток «tripadvisor.com» має інформацію як про ресторани, готелі, так і про розважальні заходи та місця, але в ньому відсутні будь-які бронювання або запис на заходи. «airbnb.com.ua» є гарним додатком для бронювання номерів готелів, а також перегляд визначних місць, але у ньому немає ресторанів. Додаток «opentable.com», навпаки, має ресторани та бронювання столиків, але у ньому немає готелів та визначних місць. Тому, жоден з додатків не має усі поставлені задачі в один момент, а для користувачів важливо це скомпонувати.

Згідно з результатів порівняння аналогів, було виявлено всі переваги та недоліки задля того, щоб гарно спроектувати додаток планування розваг та меню готелів.

1.3 Визначення архітектури веб-додатку

Визначення архітектури додатку є одним із завдань кваліфікаційної роботи. Правильно буде зробити веб-додаток, адже це буде завжди актуально та зручно. У зв'язку з потребою зберігання, систематизації та редагування даних лише обмеженою групою користувачів, було визначено, що найкращим

рішенням буде використання клієнт-серверної архітектури для додатку. Це дозволить забезпечити централізоване збереження та керування даними на сервері, а клієнтська частина додатку буде відповідальна за взаємодію з користувачем.

1.3.1 Визначення архітектури серверної частини веб-додатку

Серверна частина програми передбачає розгляд наступних архітектурних підходів:

1. Модульна архітектура;
2. Мікросервісна архітектура;
3. Монолітна архітектура.

Модульна архітектура – це підхід до розробки програмного забезпечення, який передбачає поділ системи на окремі модулі з чітко визначеними функціональними областями. Кожен модуль виконує конкретні завдання і може бути розроблений та тестований окремо. Якщо проаналізувати необхідний функціонал, то все ж таки модульна архітектура серверної частини буде не дуже доречною для додатку планування розваг та меню готелів. Необхідно, щоб створена система була задіяна уся, тому що додаток не буде занадто масштабним для поділу серверної частини на модулі.

Якщо розглядати мікросервісну архітектуру, то даний підхід до розробки програмного забезпечення базується на розбитті програмної системи на невеликі незалежні сервіси, які працюють разом із використанням легко змінюваних та масштабованих компонентів. Кожен сервіс фокусується на конкретній функціональності і може бути розроблений, розгорнутий та масштабований окремо від інших сервісів. Дана архітектура теж не є доцільною, адже мікросервісна архітектура є більш доречною для масштабних додатків, які містять функції, які можуть працювати незалежно одна від одної. У цьому випадку така архітектура буде лише сповільняти розробку додатку. Окрім того, частини веб-програми є досить взаємозалежними і не можуть бути окремо.

Монолітна архітектура є традиційним підходом до побудови серверних додатків, де весь функціонал і логіка додатку розміщені в одному монолітному блоку. В такій архітектурі весь код, бази даних, інтерфейси користувача та інші компоненти розміщені в одному проєкті та розгортаються як єдиний додаток. Масштабування окремих компонентів може бути важким через їх залежності від інших частин додатку, але для власного додатку не потрібно якесь складне масштабування, тому було обрано дану архітектуру для розробки веб-додатку планування розваг та меню готелів.

Для реалізації серверної частини було обрано фреймворк Node.js – Strapi [9], який базується на монолітній архітектурі. Він є досить простим у використанні та ефективно справиться з поставленими задачами для серверної частини.

Отже, монолітна архітектура є найбільш оптимальним варіантом серед розглянутих архітектур. Використовуючи цю архітектуру, розробка додатків не є надто складною, добре масштабується для додатків середнього розміру і спрощує процес розробки нових функцій.

1.3.2 Визначення структури клієнтської частини веб-додатку

Визначивши серверну архітектуру, варто обрати схожу до неї архітектуру на клієнтській частині. Необхідно обрати такий самий сучасний стек технологій, який буде гарно компонувати до серверної частини. Залишилось обрати вид клієнтського додатку.

Було розглянуто такі види клієнтських додатків:

1. SPA (Single Page Application) – односторінковий додаток;
2. MPA (Multi Page Application) – багатосторінковий додаток.

В SPA все наповнення завантажується одним разом при першому запуску додатку. Після цього взаємодія з додатком відбувається без перезавантаження сторінки. Всі необхідні дані та шаблони сторінок завантажуються одним разом, а зміни наповнення на сторінці відбуваються динамічно за допомогою JavaScript. SPA забезпечують плавну та швидку навігацію без

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						18

перезавантаження сторінок. Приклади фреймворків/бібліотек для розробки SPA: React, Next.js, Angular, Vue.js.

Головною перевагою SPA є те, що він є швидким та зручним, він зменшує навантаження на сервер, тому що у SPA сервер передає один раз все необхідне наповнення та шаблони сторінок, а подальші динамічні зміни відбуваються на клієнтському боці. Це дозволяє зменшити навантаження на сервер та знизити кількість запитів.

Але є й недоліки у SPA. Основним недоліком є те, що для початкового завантаження потрібно більше часу, оскільки все наповнення SPA завантажується один раз, особливо якщо додаток має великий обсяг даних або залежить від зовнішніх бібліотек і ресурсів. Також недоліком є те, що деякі старі браузері можуть не повністю підтримувати сучасні функції JavaScript, які використовуються в SPA. Також з огляду на те, що наповнення SPA завантажується динамічно, пошукові системи можуть мати складнощі з індексацією та інтерпретацією сторінок, що може негативно вплинути на SEO. Але ці недоліки аж ніяк не перекреслюють переваги, адже це просто певні аспекти, які потрібно взяти до уваги.

Тепер перейдемо до багатосторінкових додатків. МРА використовує традиційний підхід, де кожна сторінка перезавантажується при навігації. У МРА кожна сторінка має свою унікальну URL-адресу і завантажується окремо з сервера при кожній навігації. Кожна сторінка може мати свої власні ресурси (HTML, CSS, JavaScript), які завантажуються окремо. Приклади фреймворків/бібліотек для розробки МРА: Django, Ruby on Rails, Laravel.

Перевагами МРА є те, що розробка МРА зазвичай менш складна порівняно з розробкою SPA. Кожна сторінка може бути розроблена окремо, що спрощує розподіл роботи між командою розробників та дозволяє використовувати традиційні підходи до розробки веб-сторінок. Також оскільки кожна сторінка МРА має свою унікальну URL-адресу, пошукові системи можуть легко індексувати та інтерпретувати сторінки МРА. Це полегшує оптимізацію для пошукових систем та може позитивно вплинути на SEO.

вашого веб-додатку. МРА має кращу підтримку для старих браузерів, оскільки він використовує стандартну модель перезавантаження сторінок.

Недоліками МРА є те, що кожен раз, коли користувач переходить на нову сторінку, сервер повинен відправити повний HTML-код сторінки, включаючи все наповнення, стилі, скрипти та інші ресурси. Це може призвести до перевантаження сервера, особливо при великому навантаженні або при запитах багатьох користувачів одночасно. Завантаження повної сторінки з сервера може займати багато часу, особливо якщо сторінка містить велику кількість вмісту або вимагає складних операцій на стороні сервера. Це є також вагомим недоліком МРА. Недостатня плавність також може впливати на враження користувачів у погану сторону, тому що при використанні анімацій або переходів між сторінками може виникати помітна затримка або миготіння.

Ці недоліки МРА можуть бути помітнішими в сучасних додатках, де користувачі очікують швидкого та плавного взаємодії з додатком. Для таких сценаріїв часто використовують SPA, які мають набагато більше переваг та позбавлені деяких недоліків МРА. Тому було вирішено обрати саме односторінковий додаток, тим більше, що це є сучасним та зручним.

Тому було вирішено, що клієнтська частина буде представлена у вигляді SPA-додатку. Такий підхід значно спрощує розробку та пришвидшує завантаження сторінки.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Оскільки для серверної частини було обрано сучасний фреймворк Strapi, також потрібно обрати для клієнтської частини сучасну технологію. Серед таких технологій є React.JS, але ще краще використати вебфреймворк Next.JS, що дозволяє розробляти веб-додатки на основі React.JS [19].

При монолітній архітектурі на сервері, на клієнтській частині ми буде використовувати Next.JS, в основі якого є компоненти, тобто, модулі.

Використання компонентів є зручним та дозволяє підлаштовувати програмний код під різні схожі ситуації та перевикористовувати його.

Також було використано додаткові сервіси для функціональності додатку:

- Google Calendar API – сервіс, що дозволить зв’язатися з календарем користувача та зробити нагадування [3];
- Telegram Bot API – сервіс, що створений для зв’язку програмного коду власного додатку з власне створеним чат-ботом [24];
- Styled Components – бібліотека, що дозволяє працювати зі стилями прямо в JavaScript [16];
- Supabase – це відкрите хмарне середовище, яке надає повний набір інструментів для розробки веб-додатків з використанням бази даних, конкретно у даному випадку використовується для авторизації користувача в Google [3];
- Redux Thunk – це middleware (проміжний програмний засіб) для бібліотеки Redux, який дозволяє використовувати асинхронні дії (асинхронні диспетчери) у Redux-проекті;
- ESLint – популярний інструмент статичного аналізу коду для JavaScript. Використовується для виявлення помилок, невідповідностей до стандартів коду та потенційно проблемного коду в проекті [29].

Було сформовано системні вимоги, за умови виконання яких, система працюватиме стабільно:

1. Вимоги до середовища виконання:

Система повинна задовольняти вимогам на комп’ютері, що знаходиться в наступній мінімальній комплектації:

- операційна система Windows 7 або вище;
- 32-розрядний (x86) або 64-розрядний (x64) процесор із тактовою частотою 1 ГГц або вище;
- 1 ГБ (для 32-розрядних процесорів) або 2 ГБ (для 64-розрядних процесорів) ОЗУ;

					ІПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						21

- 10 ГБ (для 32-розрядної системи) або 14 ГБ (для 64-розрядної системи) вільного місця на жорсткому диску;
- 2. Вимоги до СУБД та доступу до даних:
 - У ядрі системи повинна бути представлена СУБД реляційного доступу.
- 3. Вимоги до клієнтського пристрою
 - операційна система з підтримкою браузерів (або їх наступних версій):
 - Chrome 50
 - Firefox 47
 - Edge 18
 - Opera 38
 - 512 МБ ОЗУ

Висновки до першого розділу

У першому розділі проаналізовано вимоги до програми та предметної області. Визначено завдання та етапи виконання завдань. Також було визначено стек технологій, за допомогою яких буде розроблятися додаток.

Було виконано детальний порівняльний аналіз різних аналогічних додатків з метою ідентифікації їх переваг та недоліків. Цей аналіз зробив можливим визначення найкращих аспектів кожного з додатків, які можуть бути використані при проектуванні власного додатку планування розваг та меню готелів.

Також було проведено аналіз різних архітектурних підходів для розробки веб-додатків, і в результаті було визначено, що додаток для планування розваг та меню готелів буде реалізовано за клієнт-серверною архітектурою. Зазначено, що серверна частина буде розроблена у форматі монолітного додатку.

Була встановлена структура клієнтської частини, яка буде реалізована у формі односторінкового додатку (SPA). Обраний підхід є найбільш

оптимальним для досягнення поставлених завдань, а також надасть можливість масштабувати додаток і розробляти новий функціонал у майбутньому.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ДОДАТКУ ПЛАНУВАННЯ РОЗВАГ ТА МЕНЮ ГОТЕЛІВ

2.1. Визначення варіантів використання та структури системи

Для забезпечення неперервного виконання необхідних функцій і відповідності встановленим стандартам та специфікаціям, наведено високорівневі вимоги, які повинна задовольняти веб-орієнтована система планування розваг та меню готелів. Ці вимоги допомагають забезпечити безперебійну роботу системи та виконання всіх необхідних стандартів і специфікацій.

Бізнес вимоги

1. Основні цілі: проект створюється з метою спростити взаємодію між власниками готелів та автоматизації певних процесів.
2. Представлення проекту: проект буде реалізовано у вигляді сайту, що містить структуровану інформацію по даній тематичній області.

Вимоги користувачів

1. Можливість створення та редагування особистих даних як власника, так і відвідувачів готелю;
2. Можливість зручного внесення інформації;
3. Можливість створення меню та планів розважальних заходів;
4. Можливість зміни інформації або її видалення;
5. Можливість переглядати список як готелів, меню, так і розважальних заходів;
6. Можливість робити замовлення страв заздалегідь;
7. Можливість запису на розважальний захід, а також створення нагадування у Google Calendar.

Вимоги персоналу веб-додатку (власники готелів):

1. Можливість перегляду списку, створення, редагування та видалення усіх даних, які стосуються лише готелів власника.

2. Можливість зв'язку з користувачами, коли вони замовляють страви, та останні вже будуть відправлятися у чат-бот як замовлення.

Характеристика об'єкта комп'ютеризації

Користувач додатку буде мати можливість переглядати інформацію про активні розважальні заходи, мати можливість пошуку та фільтрації, а також легко працювати з додатком через простоту та зручність.

Функціональні вимоги:

1. Авторизація у системі: в системі повинна бути представлена можливість реєстрації користувача та присвоєння йому відповідної ролі (користувач, власник готелю).
2. Можливість збереження інформації: система повинна зберігати інформацію і надавати можливість користувачу керувати нею.
3. Можливість входу в Google обліковий запис та встановлення нагадування в Google Calendar.

Нефункціональні вимоги:

1. Сприйняття
 - час, необхідний для навчання звичайних користувачів – 3-4 години, а для навчання просунутих користувачів – 1 година;
 - час відгуку для типових задач – не більше 15 секунд, для складних завдань – не більше 30 секунд.
 - інтерфейс має бути зрозумілим та з відповідними підказками.
2. Надійність
 - доступність – час, що витрачається на обслуговування системи не повинно перевищувати 6% від загального часу роботи.
 - середній час безвідмовної роботи – 15 робочих днів.
 - максимальна норма помилок та дефектів в роботі системи – 1 помилка на десять тисяч рядків коду.
3. Продуктивність
 - система повинна підтримувати мінімум 40 одночасно працюючих користувачів, пов'язаних із загальною базою даних.

					ІПЗ.КР.Б – 121 – 23 – ІЗ	Арк.
						25

4. Можливість експлуатації

- масштабування – система повинна мати можливість збільшувати потужності (продуктивність), зі збільшенням користувачів таким чином, щоб це не вплинуло негативно на її роботу.

Аналіз вимог користувачів дав підстави для визначення варіантів використання додатку планування розваг та меню готелів. На рис. 2.1 наведено діаграму використання програмної системи.

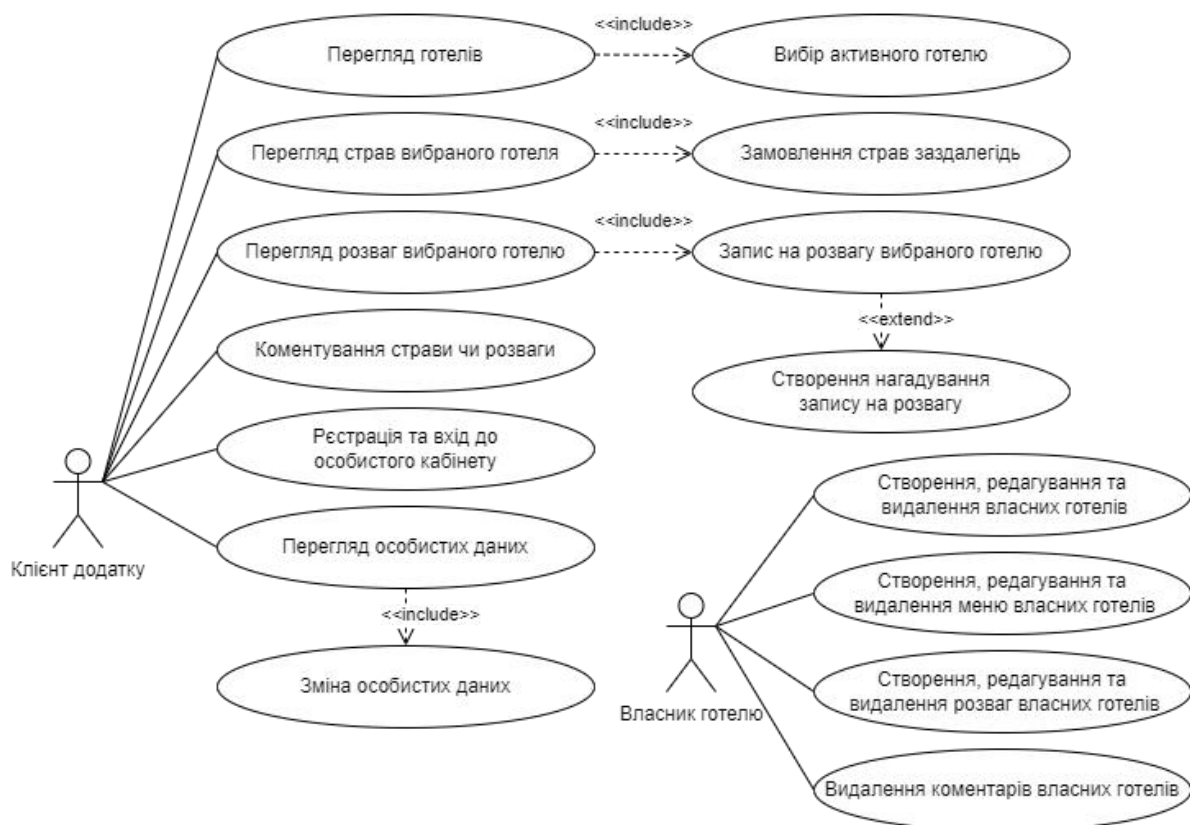


Рисунок 2.1 – Варіанти використання додатку планування розваг та меню готелів

У результаті аналізу функціональних вимог було створено модульні діаграми (рис. 2.2), що відображають складові частини системи, діаграму сутностей (рис. 2.3), яка визначає класи, що відображають відповідні таблиці у базі даних, та наближену діаграму класів модуля, за допомогою прикладу модуля відгуків (рис. 2.4).

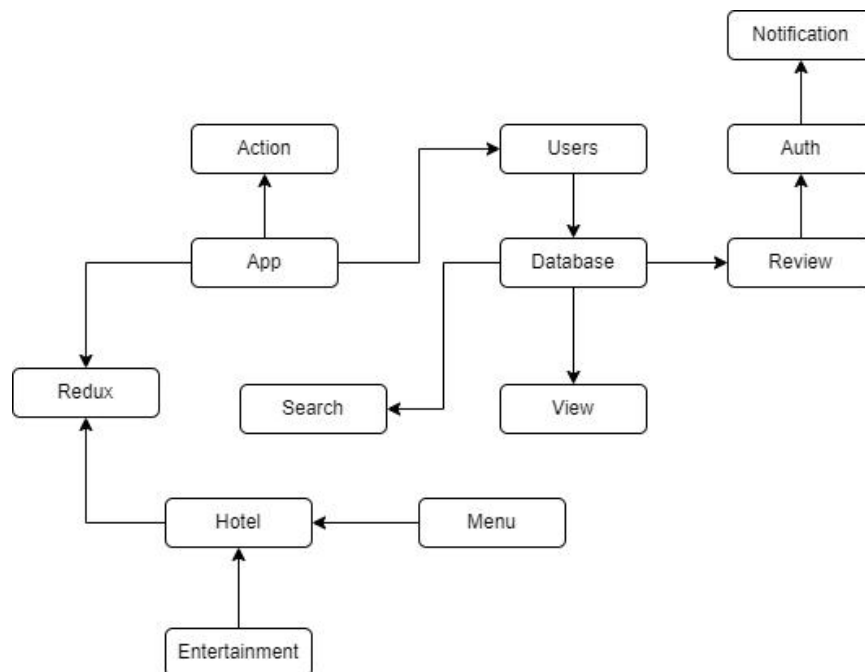


Рисунок 2.2 – Діаграма модулів додатку планування розваг та меню готелів

На діаграмі модулів зображені частини системи – модулі, кожен з яких відповідає за свою дію. Модулі можуть залежати один від одного.

Було спроектовано 14 модулів:

1. App – модуль, що виступає як вихідний для додатку, виконує ініціалізацію API, але при цьому не має жодної залежності інших модулів від нього;
2. Action – модуль, який виступає за будь-які дії з боку як користувача, так і власника готелю;
3. Auth – модуль, що відповідає за авторизацію та інші задачі пов'язані з сесією користувача;
4. Users – модуль, що надає інформацію щодо користувачів;
5. Database – відповідає за зв'язок та надсилання запитів до бази даних;
6. View – модуль, який відповідає за відображення;
7. Review – модуль відгуків, дозволяє залишити відгук для певної сутності, та отримати всі відгуки до певної сутності;
8. Notification – модуль, який відповідає за надсилання повідомлень в Telegram Bot або запис нагадування в Google Calendar;

9. Search – модуль, який відповідає за індексацію сутностей для пошуку та фільтрації сутностей;
10. Redux – модуль, який відповідає за зберігання даних у додатку;
11. Hotel – модуль для готелів, їх відображення, редагування та видалення;
12. Menu – модуль для меню, їх відображення, редагування та видалення;
13. Entertainment – модуль для розваг, їх відображення, редагування та видалення;

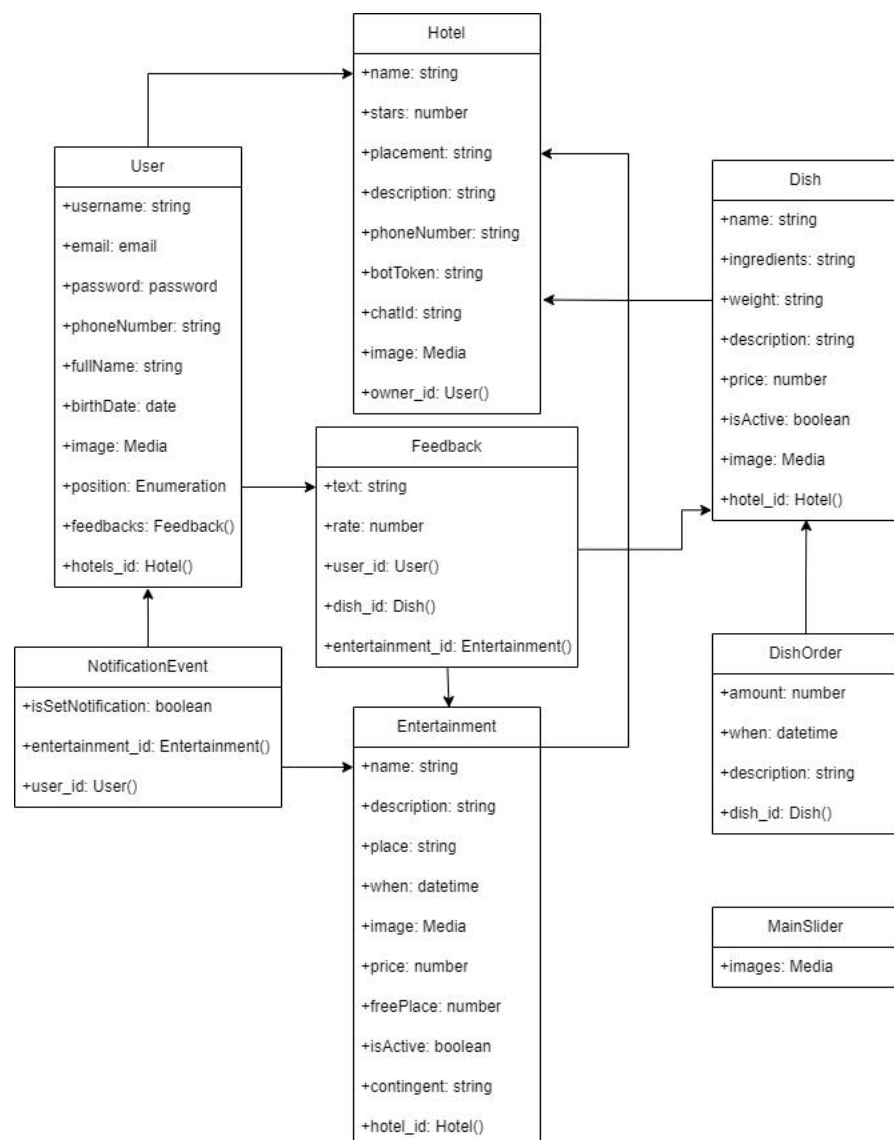


Рисунок 2.3 – Діаграма класів сутностей додатку планування розваг та меню готелів

Зображена діаграма класів сутностей відображає тільки класи, які мають відповідний відображення в базі даних. Сутності в цій діаграмі виключно містять інформацію і не мають методів. Операції над сутностями здійснюються через контролери. В додатку присутні 8 сутностей:

1. User – зберігає дані користувача;
2. Hotel – зберігає дані готелю;
3. Dish – зберігає дані страви;
4. Entertainment – зберігає дані розважального заходу;
5. DishOrder – зберігає дані про замовлення страви;
6. NotificationEvent – зберігає дані про запис на розважальний захід та нагадування;
7. Feedback – зберігає дані про відгук до страви чи розважального заходу;
8. MainSlider – зберігає зображення для слайдеру на головній сторінці;

2.2. Розробка бази даних

У якості СУБД було обрано PostgreSQL, адже вона відома своєю стабільністю, масштабованістю та великими можливостями.

Тому завдяки зручному інтерфейсу Strapi, було створено наступні таблиці:

- Dish – таблиця, яка відповідає за страву у конкретному готелі;
- DishOrder – таблиця, яка відповідає за замовлення страви користувачем;
- Entertainment – таблиця, яка відповідає за розважальний захід в конкретному готелі;
- Feedback – таблиця, яка відповідає за відгуки як для страв, так і для розважального заходу;
- Hotel – таблиця, яка відповідає за готелі;
- NotificationEvent – таблиця, яка відповідає за запис користувачів на розважальний захід, а також за те, чи поставив користувач нагадування в Google Calendar;

- З вихідних таблиць була побудована діаграма бази даних:



Ця діаграма відображає всі таблиці та їх зв'язки. З неї видно, що одна таблиця не має зв'язків. Це зумовлено тим, що дані з цієї таблиці ніколи не використовуються, оновлюються або зберігаються разом з даними інших таблиць.

2.3 Проектування та реалізація алгоритмів роботи додатку

Як основний алгоритм, було вирішено використовувати звичайний алгоритм односторінкового додатку (SPA) для клієнтської частини системи (див. Рисунок 2.6). При переході на сторінку додатку або відкритті мобільного додатку, сервер надсилає користувачеві актуальну версію сторінки (клієнтську частину). Потім, залежно від дій користувача, клієнт надсилає запити на сервер. Після отримання відповіді від серверу, клієнт змінює вигляд сторінки.



Рисунок 2.5 – Загальний алгоритм роботи додатку

Склавши загальний алгоритм додатку, далі необхідно продумати алгоритм для окремих випадків. Для представлення подібних сценаріїв зазвичай використовуються діаграми діяльності (activity diagram). Даною діаграмою можна додатково роз'яснити деякі дії у додатку.

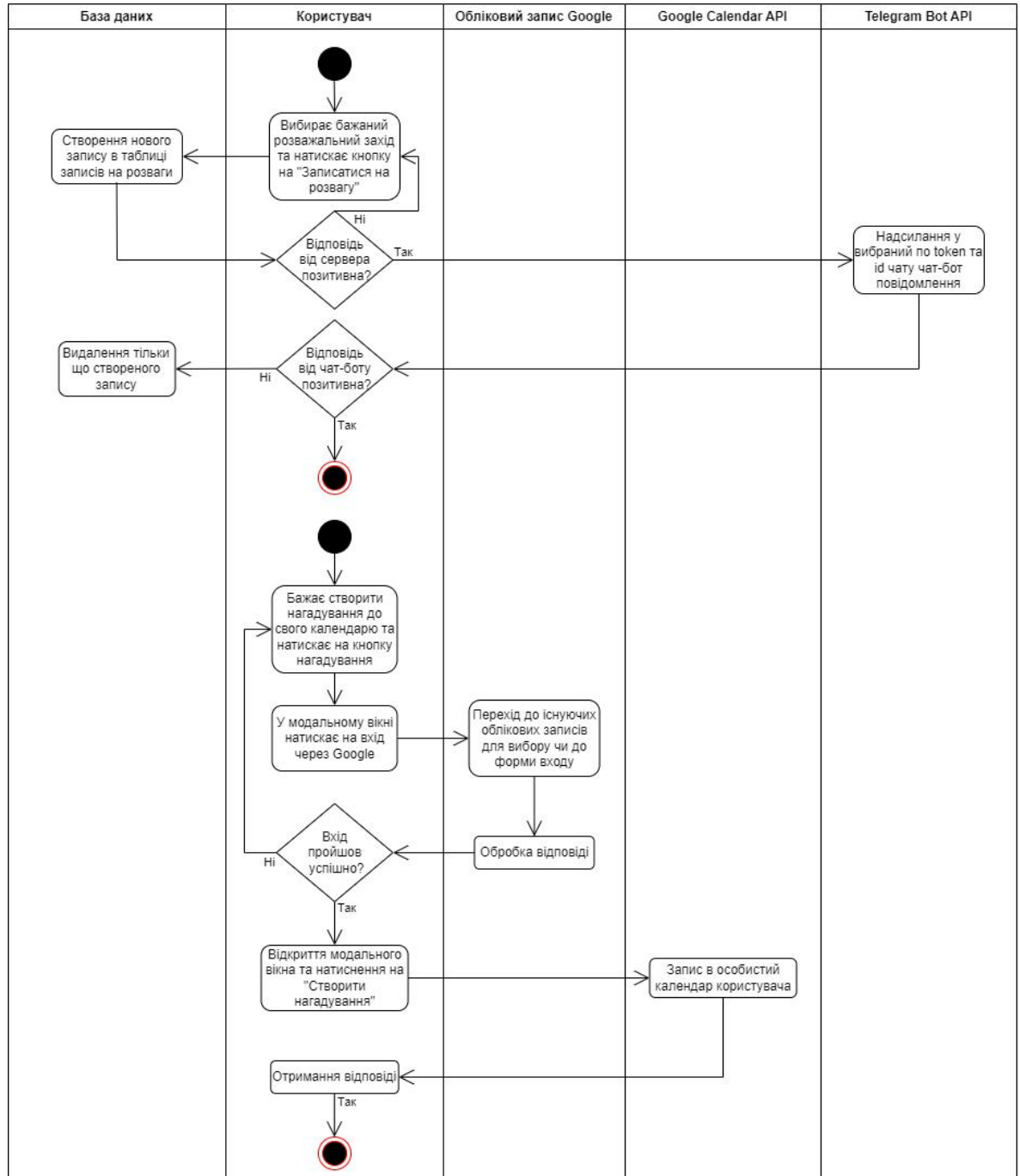


Рисунок 2.6 – Діаграма діяльності «Запис користувача на розважальний захід»

Розглянемо процес запису користувача на розважальний захід (рис. 2.7). Спочатку користувач вибирає розважальний захід, який він би хотів відвідати.

На сторінці конкретного розважального заходу є кнопка “Записатися на розвагу”, за умови, якщо користувач увійшов у систему. Натискаючи на цю кнопку, відправляється запит на записування даних до БД. Якщо відповідь повертається позитивна, то надсилається повідомлення до Телеграм боту, який є у власності власника готелю. Якщо відповідь від чат-боту прийде помилкою, то необхідно видалити створений запис у БД. Якщо відповідь приходить без помилки, то дія з записом на розважальний захід завершена.

Далі за бажанням користувача він може створити нагадування для себе у Google Calendar. На сторінці вибраної розваги у разі того, якщо користувач на неї записаний, буде відображена кнопка встановлення нагадування. Якщо натиснути на неї, буде запропоновано увійти в обліковий запис Google. При натисканні на неї користувача перенаправить на сторінку входу Google. Обравши або зайшовши до облікового запису, буде повернута відповідь з успішним чи не успішним входженням до облікового запису. Якщо відповідь прийшла з помилкою, користувач може спробувати ще раз. Якщо відповідь без помилок, то перейшовши знову до модального вікна нагадувань, користувач побачить свою електронну пошту. Далі буде запропоновано або вийти з облікового запису Google, або ж створити нагадування. Якщо обрати “Створити нагадування”, то запит піде до Google Calendar API. Створиться нагадування у особистий календар користувача. Якщо все пройде успішно, то у календарі користувач зможе побачити нагадування на розважальний захід.

2.4. Реалізація функціоналу додатку

2.4.1 Реалізація основної архітектури додатку

Додаток буде складатися з клієнтської та серверної частини, кожна з яких буде окремим додатком та знаходитиметься в окремому каталозі «next» та «strapi» відповідно.

Серверна частина додатку була розроблена з використанням інтерфейсу командного рядка та фреймворку “Strapi”. У створеному проєкті буде наявний

					ІПЗ.КР.Б – 121 – 23 – ІЗ	Арк.
						33

модуль та функція для запуску, а всі необхідні бібліотеки вже наявні для коректної роботи додатку.

Функція “bootstrap” у файлі “app.example.js” є початковою точкою входу в додаток. В цій функції створюється об'єкт додатку і налаштовуються його параметри:

```
const config = {
  locales: [
    // 'ar',
    // 'fr',
    // 'cs',
    // 'de',
    // 'dk',
    // 'es',
    // 'he',
    // 'id',
    // 'it',
    // 'ja',
    // 'ko',
    // 'ms',
    // 'nl',
    // 'no',
    // 'pl',
    // 'pt-BR',
    // 'pt',
    // 'ru',
    // 'sk',
    // 'sv',
    // 'th',
    // 'tr',
    // 'uk',
    // 'vi',
    // 'zh-Hans',
    // 'zh',
  ],
};

const bootstrap = (app) => {
  console.log(app);
};

export default {
  config,
  bootstrap,
};
```

В каталозі налаштування також є файл “server.js”, і цей модуль відповідає за налаштування серверу та його завантаження:

```
module.exports = ({ env }) => ({
  host: env('HOST', '0.0.0.0'),
  port: env.int('PORT', 1337),
  app: {
    keys: env.array('APP_KEYS'),
  },
});
```

```
webhooks: {
  populateRelations: env.bool('WEBHOOKS_POPULATE_RELATIONS', false),
},
});
```

З боку клієнтської частини додатку було використано вебфреймворк Next.JS, який створюється за допомогою інтерфейсу командного рядка.

Файл “_app.js” є початковою точкою входу в додаток. Цей файл є центральною точкою контролю над рендерингом додатку в Next.JS. Він дозволяє налаштовувати та впливати на рендеринг усіх сторінок додатку:

```
import { Provider } from 'react-redux';
import store from '@redux/store';
import Layout from '@sections/Layout/Layout';
import { CookiesProvider } from 'react-cookie';
import { SessionContextProvider } from '@supabase/auth-helpers-react';
import { createClient } from '@supabase/supabase-js';
import PropTypes from 'prop-types';
import '@styles/globals.css';

const supabase = createClient(
  'https://qvbbeeblpqvcqcrbut.supabase.co',

  'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InF2YmJwZWVibHBxb3ZjcWNYbnV0Iiwicm9sZSI6ImFub24iLCJpYXQiOiJlE2ODY4NDM5MjUsImV4cCI6IjE6MjAwMjQxOTkyNX0. aqNbn-dsdxgx4Kz_IN88MxomtPUMKu60SQbiDIqviQ',
);

export default function App({ Component, pageProps }) {
  return (
    <CookiesProvider>
      <Provider store={store}>
        <SessionContextProvider supabaseClient={supabase}>
          <Layout>
            <Component {...pageProps} />
          </Layout>
        </SessionContextProvider>
      </Provider>
    </CookiesProvider>
  );
}

App.defaultProps = {
  pageProps: {},
};

App.propTypes = {
  Component: PropTypes.func.isRequired,
  pageProps: PropTypes.object,
};
```

У даному файлі є наступні компоненти:

- `CookiesProvider` – компонент, що надає доступ до читання, зміни та видалення cookies;

- Provider – компонент з бібліотеки Redux, що надає доступ до його стану;
- SessionContextProvider – компонент, який використовується для керування станом сеансу або авторизації в додатку, у нашому випадку для авторизації в Google;
- Layout – компонент, що надає обгортку для кожної сторінки;
- Component – компонент, до якого приходить в _app.js та є сторінкою, на яку перейшов користувач;

Для зручної роботи зі стилями було використано бібліотеку “styled-components”. Вона дозволяє писати CSS-стили безпосередньо в JavaScript-коді, що спрощує створення та управління стилями компонентів. Наведемо приклад стилізації:

```
import styled from 'styled-components';

export const StyledContainer = styled.div`
  display: flex;
  flex-direction: column;
  width: 100%;
  max-width: 1440px;
  margin: 0 auto;

  @media (max-width: 1450px) {
    max-width: 1200px;
    padding: 0 10px;
  }
  @media (max-width: 1200px) {
    max-width: 992px;
  }
  @media (max-width: 992px) {
    max-width: 768px;
  }
  @media (max-width: 768px) {
    max-width: 576px;
  }
`;
```

Для спрощення управління станом додатка було використано redux-thunk – середовище (middleware) для бібліотеки Redux, що дозволяє обробляти асинхронні дії в Redux-додатках. Стандартно Redux дозволяє обробляти синхронні дії, які представлені об'єктами дій (actions). Однак, для додатку планування розваг та меню готелів потрібно використовувати асинхронні операції, такі як запити до сервера та взаємодію з асинхронними API, щоб

зручно керувати цими операціями. Є й інші різновиди Redux, але це середовище буде найдоцільніше використати, адже воно просте та зручне, а також не займає багато місця при встановленні. Об'єкт “store” є центральним елементом для управління станом додатка. Він зберігає стан додатка і дозволяє виконувати дії зі зміною стану та підписуватися на зміни стану. Далі буде наведено код файлу “store.js”:

```
import { createStore, applyMiddleware } from 'redux';
import thunk from 'redux-thunk';
import { rootReducer } from '@redux/reducers';

const store = createStore(rootReducer, applyMiddleware(thunk));

export default store;
```

Основна роль “store.js” полягає у визначенні конфігурації store та його початкового стану. У цьому файлі встановлюється middleware, редуктори та інші налаштування, необхідні для роботи з Redux. Функція createStore використовується для створення об'єкта store, applyMiddleware для використання middleware та rootReducer для об'єднання всіх редукторів додатка.

Для роботи redux-thunk також потрібні редуктори та дії, які будуть запускати послідовності подій. Щоб простіше було працювати, всі редуктори потрібно об'єднати в один файл:

```
import { combineReducers } from 'redux';
import auth from '@redux/reducers/auth';
import feedbacks from '@redux/reducers/feedbacks';
import collection from '@redux/reducers/collection';
import activeHotel from '@redux/reducers/activeHotel';

export const rootReducer = combineReducers({
  auth,
  feedbacks,
  collection,
  activeHotel,
});
```

Функція combineReducers отримує об'єкт, в якому ключі відповідають назвам полів стану, а значення – редукторам, які відповідають за ці поля. В результаті виклику combineReducers створюється основний редуктор, який об'єднує функціонал усіх переданих редукторів.

Отож, ці конфігурації дозволяють керувати станом додатку, що покращує його продуктивність і дозволяє зосередитись лише на тих частинах

користувачького інтерфейсу, з якими користувач активно взаємодіє, незалежно від місця у додатку.

Створимо build нашого додатку, тобто, це процес компіляції та побудови готового для виконання варіанту додатку, який може бути розгорнутий та запуснений на сервері або виконаний на клієнтському пристрої:

Route (pages)	Size	First Load JS
λ /	27.4 kB	208 kB
└ css/dcd4e44737d35dad.css	4.61 kB	
└ /_app	0 B	180 kB
○ /404	1.43 kB	182 kB
λ /api/telegram	0 B	180 kB
λ /cabinet	35.1 kB	223 kB
λ /dishes	4.59 kB	195 kB
λ /dishes/dish/[id]	7.46 kB	194 kB
λ /entertainments	14.9 kB	205 kB
λ /entertainments/entertainment/[id]	6.67 kB	194 kB
└ λ /hotels	4.27 kB	194 kB
+ First Load JS shared by all	184 kB	
└ chunks/framework-2c79e2a64abdb08b.js	45.2 kB	
└ chunks/main-4fe93428a0daae30.js	32.2 kB	
└ chunks/pages/_app-bbbea64bf52919b1.js	101 kB	
└ chunks/webpack-9ea31454a0f783e4.js	1.86 kB	
└ css/c74924dc2e17c9f0.css	3.68 kB	
λ (Server) server-side renders at runtime (uses <code>getInitialProps</code> or <code>getServerSideProps</code>)		
○ (Static) automatically rendered as static HTML (uses no initial props)		
info - Creating an optimized production build ...		

Рисунок 2.7 – Створення build додатку

2.4.2 Реалізація компонента для входу в обліковий запис Google та створення нагадування у Google Calendar

Один з головних принципів React.JS – це "одне джерело правди". Цей принцип позначає те, що стан додатку повинен бути збережений і керований лише в одному місці – в компонентах React. Такий підхід сприяє простоті та передбачуваності розробки, оскільки зміни стану відбуваються тільки в одному місці – в компонентах React. Це також полегшує налагодження коду, оскільки причиною проблем зазвичай є неочікувані зміни стану, які можна відслідковувати і виправити в одному місці.

Перейдемо до реалізації компоненту, який буде відповідати за встановлення нагадувань.

```
const session = useSession();
const supabase = useSupabaseClient();

const googleSignIn = async () => {
  const { error } = await supabase.auth.signInWithOAuth({
    provider: 'google',
    options: {
      scopes: 'https://www.googleapis.com/auth/calendar',
      redirectTo:
`http://localhost:3000/entertainments/entertainment/${entertainmentId}`,
    },
  });
  if (error) {
    notify(
      <>
      <p>Щось пішло не так...</p>
      <p>Спробуйте ще раз увійти.</p>
      </>,
      '',
    );
  }
};
```

Для спрощення роботи з обліковим записом Google було обрано Supabase – відкриту платформу для розробки веб-додатків, яка надає інфраструктуру для зберігання та керування даними. Вона поєднує в собі функціональність бази даних та автентифікації. Даний код описує з'єднання з Supabase та доступом до вже налаштованого на платформі Google. При помилці буде з'являтися повідомлення, що щось пішло не так.

Далі розглянемо функцію у цьому ж самому компоненті, що відповідає за встановлення нагадування в Google Calendar:

```
const createCalendarEvent = async () => {
  const event = {
    summary: entertainment.name,
    description: entertainment.description,
    location: entertainment.place,
    start: {
      dateTime: entertainment.when,
    },
    end: {
      dateTime: entertainment.when,
    },
  };

  try {
    await axios.post(
      'https://www.googleapis.com/calendar/v3/calendars/primary/events',
    );
  }
};
```

```

    event,
    {
      headers: { Authorization: `Bearer ${session.provider_token}` },
    },
  );

  await api.put(`/api/notification-events/${notificationId}`, {
    data: {
      isSetNotification: true,
    },
  });

  notify('Нагадування успішно створено!');
  dispatch(
    getCollection(
      `/api/get-notification-event/${entertainmentId}/${userId}`,
    ),
  );
  setShow(false);
} catch (error) {
  notify(
    <>
    <p>Щось пішло не так...</p>
    <p>Спробуйте ще раз встановити нагадування.</p>
    </>,
    '',
  );
}
};

```

3 Google Calendar API працювати легко і це досить популярний календар серед усіх використовуваних, тому було обрано саме його для здійснення системи нагадувань. Спочатку необхідно взяти всі необхідні дані про розважальний захід, аби передати це до Google Calendar API. Разом із запитом ми маємо надіслати токен, який зберігається при вході до облікового запису Google протягом години. Також при встановленні нагадування необхідно це якось задокументувати в БД, тому на сервер ми відправляємо PUT-запит для встановлення для користувача того, що нагадування вже в нього в календарі. При успішному створенні буде оновлено у клієнтській частині те, що було змінено на сервері та користувачу повідомлять про успішність чи не успішність встановлення нагадування.

2.4.3 Реалізація і налаштування конфігураційного файлу для з'єднання з базою даних

Цей модуль використовується для ініціалізації підключення Strapi до бази даних під час запуску сервера та збірки додатку. У конфігураційних налаштуваннях необхідно вказати певні параметри, щоб забезпечити успішне та безпечне з'єднання, що дасть доступ до всіх даних, що зберігаються у базі даних. Програмний код конфігураційного файлу “database.js”:

```
const path = require('path');

module.exports = ({ env }) => ({
  connection: {
    client: 'postgres',
    connection: {
      host: 'localhost',
      port: 5432,
      database: "dypiom",
      user: "postgres",
      password: "shved258176",
    },
  },
  useNullAsDefault: true,
});
```

У даному коді створюється та експортується конфігураційний об'єкт, який використовується для підключення до бази даних PostgreSQL, у якому є наступні поля:

- client – вказує тип бази даних, у даному випадку “postgres” для PostgreSQL;
- connection – об'єкт, що містить налаштування підключення до бази даних. У ньому вказані наступні параметри;
 - host – хост бази даних, у даному випадку “localhost”;
 - port – порт, на якому працює база даних, 5432 – стандартний порт для PostgreSQL;
 - database – назва бази даних;
 - user – користувач бази даних;
 - password – пароль користувача бази даних;
- useNullAsDefault – опція, яка встановлює за замовчуванням використання значення NULL для стовпців без вказання значення.

2.4.4 Реалізація контролера та створення власної кінцевої точки (endpoint) на сервері

У даному випадку необхідно використовувати не тільки надані посилання Strapi, але й створити власні кінцеві точки, які будуть виконувати дії, які потребує додаток. Такий підхід надасть гнучкість у розробці API для взаємодії з базою даних.

Далі показано один із контролерів, який відповідає за те, по яким критеріям шукати страви:

```
/**
 * dish controller
 */

const { createCoreController } = require("@strapi/strapi").factories;

const schema = require("../content-types/dish/schema.json");

module.exports = createCoreController("api::dish.dish", ({ strapi }) => ({
  async getDishesByHotelId(context) {
    try {
      const { hotelId } = context.params;
      const dishes = await strapi.db.query("api::dish.dish").findMany({
        populate: ["image", "hotel_id"],
        where: { hotel_id: { id: hotelId } },
      });

      context.body = {
        status: 200,
        dishes,
        schema: schema.attributes,
      };
    } catch (error) {
      context.body = {
        status: 500,
        dishes: [],
        schema: schema.attributes,
      };
    }
  },
  async getDishesByOwnerHotelId(context) {
    try {
      const { ownerId } = context.params;
      const dishes = await strapi.db.query("api::dish.dish").findMany({
        populate: ["image", "hotel_id"],
        where: { hotel_id: { owner_id: ownerId } },
      });

      context.body = {
        status: 200,
        dishes,
        schema: schema.attributes,
      };
    }
  },
});
```

```

    };
  } catch (error) {
    context.body = {
      status: 500,
      dishes: [],
      schema: schema.attributes,
    };
  }
},
async getDishById(context) {
  try {
    const { id } = context.params;
    const dish = await strapi.db.query("api::dish.dish").findMany({
      populate: ["image"],
      where: {
        id,
      },
    });

    context.body = {
      status: 200,
      dish,
      schema: schema.attributes,
    };
  } catch (error) {
    context.body = {
      status: 500,
      dish: [],
      schema: schema.attributes,
    };
  }
},
}));

```

Тут є три функції, які відповідають за пошук страв за різними параметрами: за ідентифікатором готелю, за ідентифікатором власника готелю та за ідентифікатором страви. Також, потрібно створити свої власні кінцеві точки, щоб викликати вказані вище функції:

```

module.exports = {
  routes: [
    {
      method: "GET",
      path: "/get-dish/:hotelId",
      handler: "dish.getDishesByHotelId",
      config: {
        auth: false,
      },
    },
    {
      method: "GET",
      path: "/get-dishes-by-owner/:ownerId",
      handler: "dish.getDishesByOwnerHotelId",
      config: {
        auth: false,
      },
    },
  ],
};

```

```

{
  method: "GET",
  path: "/get-dishes/:id",
  handler: "dish.getDishById",
  config: {
    auth: false,
  },
},
],
};

```

Для кожної функції у контролері створена своя власна кінцева точка, а також метод запиту та назва функції, яка буде обробляти запит.

Висновки до розділу 2

Під час розробки веб-додатку планування розваг та меню готелів були визначені функціональні та нефункціональні вимоги, бізнес вимоги, а також вимоги з боку користувачів. Разом з цими вимогами були встановлені варіанти використання системи. Була створена структура додатку та приблизна структура певних модулів додатку.

Були розроблені сутності додатку, на основі яких було спроектовано таблиці у базі даних, а також була створена діаграма бази даних для їх візуалізації.

Був розроблений основний алгоритм роботи додатку, задокументовані сценарії використання. Також були описані деякі процеси розробки як на клієнтській, так і на серверній частині.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-ДОДАТКОМ ПЛАНУВАННЯ РОЗВАГ ТА МЕНЮ ГОТЕЛІВ

3.1. Порядок встановлення та налаштування параметрів додатку

Додаток планування розваг та меню готелів складається з низки компонентів, що вимагає їх окремого розгортання та налаштування. Для відображення компонентів системи та візуалізації процесу їх розгортання була створена діаграма розгортання:

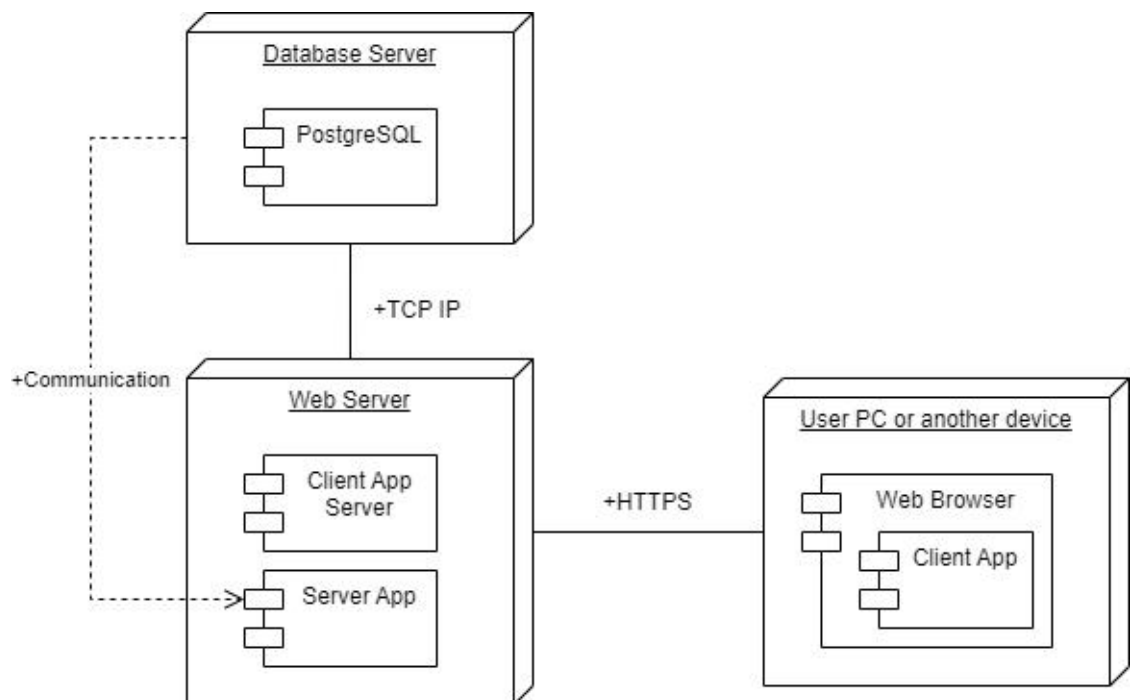


Рисунок 3.1 – Діаграма розгортання

На діаграмі розгортання зображена система, яка складається з трьох вузлів: серверу, серверу бази даних та комп'ютер або інший пристрій користувача. Зв'язок між комп'ютером або іншим пристроєм користувача та веб-сервером здійснюється через протокол HTTPS. Протокол HTTP використовується для взаємодії між клієнтським та серверним додатками всередині сервера.

Протокол HTTP/HTTPS базується на системі запитів/відповідей і використовує стандарт TCP для передачі даних без втрат. Взаємодія між

сервером бази даних та веб-сервером відбувається за допомогою протоколу TCP/IP.

Для того, щоб встановити додаток планування розваг та меню готелів, потрібно розгорнути і налаштувати компоненти системи. На сервері бази даних має бути встановлений сервер PostgreSQL, створена база даних та налаштовані дані для підключення. Клієнтський додаток потребує лише встановленого веб-браузера для кожного користувача.

Для правильної роботи сервера необхідно мати версію Node.JS версії 16 або вище. Також переглянемо файл налаштувань налаштування середовища, які використовуються при запуску додатку “.env” у каталозі “strapi”:

- HOST – вказує хост, на якому буде запущено сервер Strapi. Значення 0.0.0.0 означає, що сервер буде доступний на всіх доступних мережевих інтерфейсах;
- PORT – вказує порт, на якому буде прослуховуватися сервер Strapi;
- APP_KEYS – це ключі сервера, які використовуються для шифрування даних;
- API_TOKEN_SALT – використовується для генерації токенів API;
- ADMIN_JWT_SECRET – це секретний ключ, який використовується для підпису та перевірки JWT (JSON Web Tokens) для адміністративного доступу;
- TRANSFER_TOKEN_SALT – використовується для генерації токенів передачі даних;
- DATABASE_CLIENT – вказує клієнт бази даних, який буде використовуватися;
- DATABASE_FILENAME – вказує шлях та назву файлу бази даних;
- JWT_SECRET – секретний ключ, який використовується для підпису та перевірки JWT у загальному використанні;

Після правильно заповненого файлу конфігурації сервер має запуститися коректно.

3.2. Структура інтерфейсу веб-додатку та порядок роботи

У веб-додаток користувачі можуть зайти як з персонального комп'ютера, так і з мобільного телефону чи інших пристроїв. Сайт є повністю адаптивним для стандартних пристроїв.

Перше, що побачить користувач, перейшовши за посиланням на додаток, це головну сторінку (рис. 3.2).

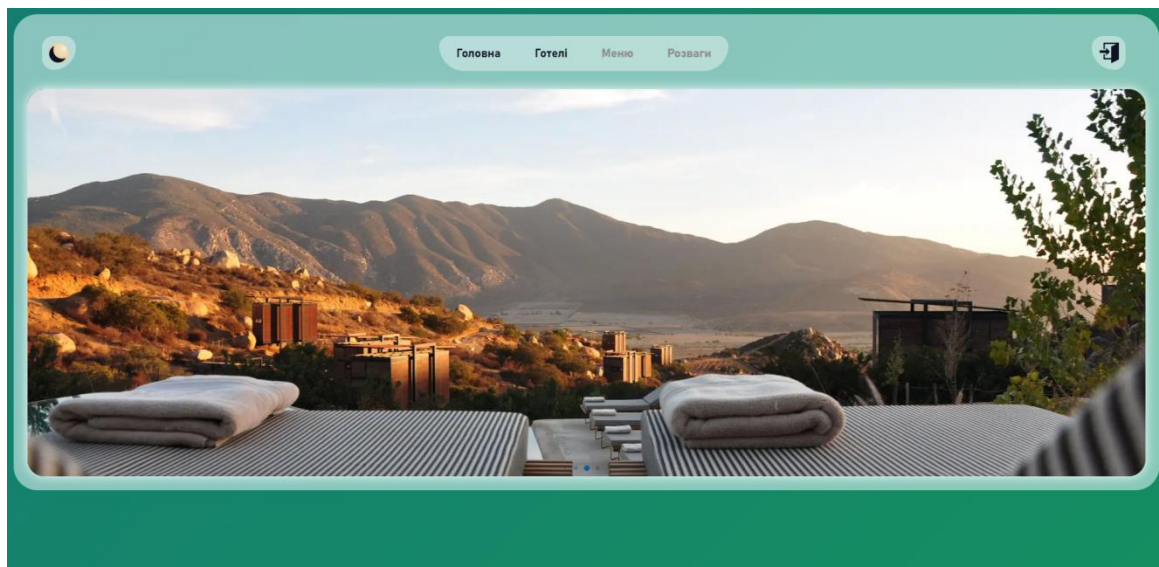


Рисунок 3.2 – Головна сторінка

У шапці додатку можна побачити активні та неактивні пункти. Це пов'язано з тим, що для того, щоб переглядати меню та розваги, необхідно вибрати активний готель у відповідному пункті. Якщо спробувати натиснути на неактивний пункт, ми побачимо наступне повідомлення:

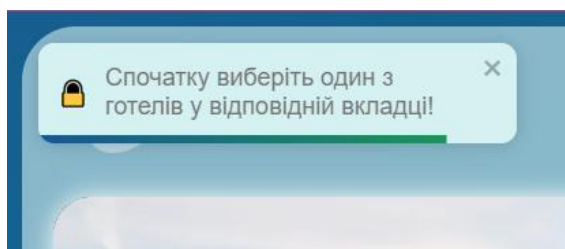


Рисунок 3.3 – Попередження про необхідність вибору активного готелю
Також у верхньому правому кутку екрану є кнопка входу на сайт (рис. 3.4).



Рисунок 3.4 – Кнопка входу в особистий кабінет

Натиснувши на кнопку входу, користувачу відкриється модальне вікно, у якому буде запропоновано ввести електронну пошту та пароль для входу (рис. 3.5).

Рисунок 3.5 – Модальне вікно входу в обліковий запис

Якщо ж користувач ще не має облікового запису, він може зареєструватися (рис. 3.6). При створенні користувач може обрати, чи буде він власником готелю.

Рисунок 3.6 – Модальне вікно реєстрації облікового запису

Коли користувач увійде в обліковий запис, то у шапці додатку з’явиться нова вкладка для переходу в кабінет (рис. 3.7).



Рисунок 3.7 – При вході додається нова вкладка “Кабінет”

Особистий кабінет звичайного користувача виглядає наступним чином (рис. 3.8).

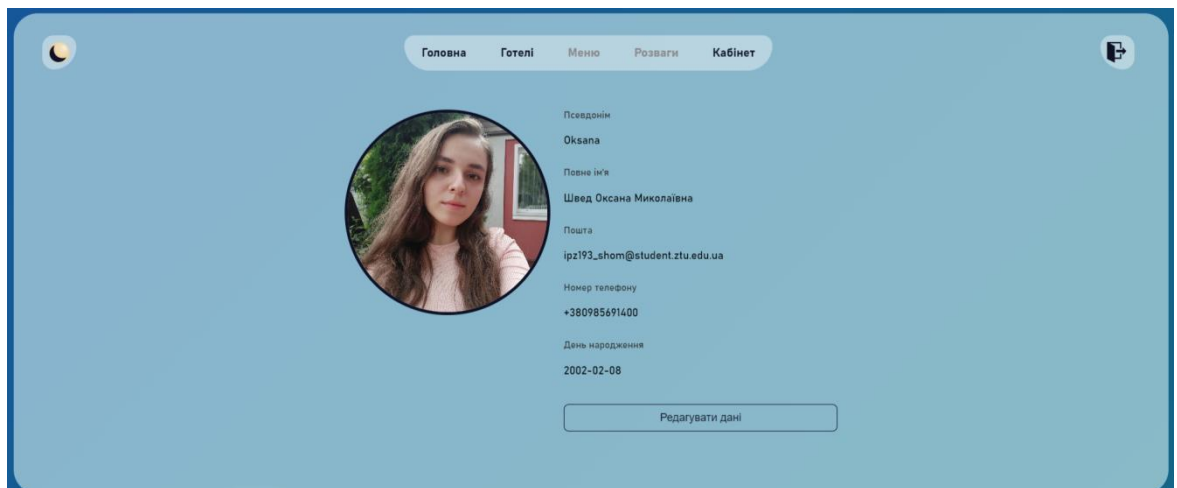


Рисунок 3.8 – Особистий кабінет звичайного користувача

В особистому кабінеті користувач може змінювати власні дані та змінювати аватар.

У додатку є дві ролі для користувача – звичайна користувацька та власника готелів.

3.2.1. Інтерфейс та порядок роботи з панеллю власника готелів

Для того, щоб отримати доступ до створення власного готелю, потрібно при реєстрації особистого запису вказати, що користувач бажає бути власником готелю. У такому випадку в особистому кабінеті знизу під інформацією користувача будуть вкладки (рис. 3.9), які відповідають за страви, розваги, відгуки та готелі.

Dish Entertainment Feedback Hotel										
Id	Name	Ingredients	Weight	Description	Price	Rate	IsActive	Image	Actions	
1	Поке боул з лососем	Рис, Лосось, Авокадо, Боби, Морква, Огірок, Цибуля зелена, Капуста червоноголова, Семена чіа, Кунжут	550 грам	Боул у перекладі з англ...	220	4.7	☒			
2	Риба під томатним соусом	Хек, Морква, Цибуля, Томатний соус, Помідори, Солодкий перець, Борошно	600 грам	Апетитна соковита туш...	200	4.5	☒			
9	Гранола з йогуртом та ягодами	Гранола, Йогурт з лохиною, Лохина, Малина, Манго, Ківі	500 грам	Йогурт з гранолою є на...	210	4.2	☒			
13	Лосось у вершковому соусі	Лосось, Борошно, Йогурт, Вершки, Помідори, Базилік, Пармезан, Лимон, Куркума	600 грам	Є страви, на які хочеть...	350	4.5	☒			
14	Поке боул салат	Курка, Помідори, Кукурудза, Боби, Капуста, Соус Песто, Базилік, Кетчуп	640 грам	Ще декілька років тому...	300	4.4	☒			

Рисунок 3.9 – Вкладки, які доступні тільки для власників готелів

Власник може створювати багато готелів та керувати тільки стравами та розважальними заходами, які відносяться до його готелів.

Dish Entertainment Feedback Hotel									
+									
Id Name Stars Placement PhoneNumber BotToken ChatId Description Image Actions									
1	Hermes	4	майдан Визволення, 6а, Житомир, Житомирська область, 10002	+380983456789	6028039864:AAFvY9lpSpIKSnBPhcKLyWur_oEdivlPPyM	549984598	Ми втілюємо багато в од...		 

Рисунок 3.10 – Готель власника

При створенні або редагуванні страви чи розважального заходу, якщо у власника є декілька готелів, є можливість вибирати, до якого готелю вони будуть відноситись:

Редагування

×

isActive

☒

rate

4,5

hotel_id

Hermes

Hermes

Zeus

Натисніть для завантаження фото або перетягніть сюди фото

Змінити

Рисунок 3.11 – Редагування того, до якого готелю відноситься страва

Також як і в користувача, так і тут, можливо змінити фотографію:

Редагування

isActive

☒

rate

4,5

hotel_id

Hermes

Змінити

Рисунок 3.12 – Вибір фотографії для страви

13

Лосось у вершковому соусі

Лосось, Борошно, Йогурт, Вершки, Помідори, Базилік, Пармезан, Лимон, Куркума

600 грам

Є страви, на які хочеть...

350

4,5

☒

Рисунок 3.13 – Фотографія була успішно змінена

При видаленні запису буде з'являтися модальне вікно з підтвердженням:

Видалення

Ви дійсно хочете видалити запис №14?

Так

Ні

Рисунок 3.14 – Модальне вікно при видаленні запису

3.2.2. Інтерфейс та порядок роботи з основною частиною додатка

Ми вже розглядали вхід до облікового запису. Тому переглянемо вигляд додатку, коли користувач вже увійшов у систему на мобільній версії:

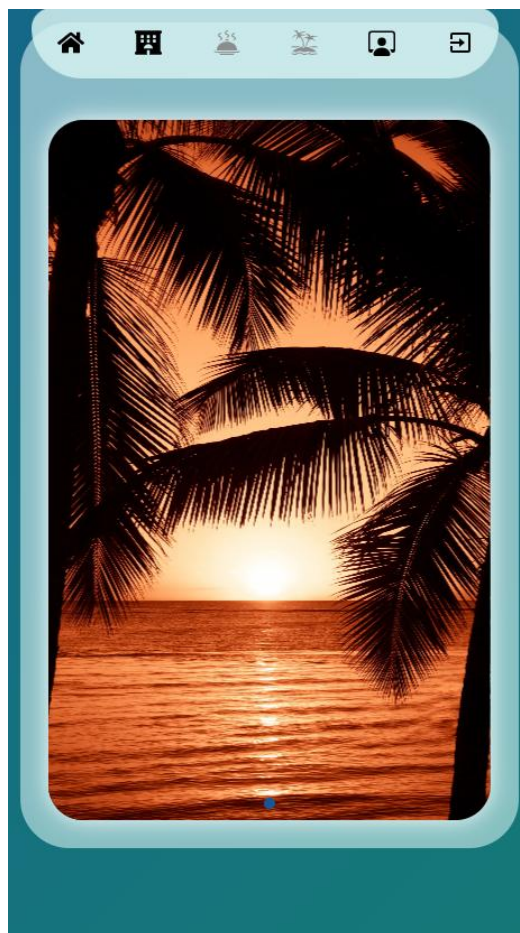


Рисунок 3.15 – Мобільна версія сайту, коли користувач вже увійшов в обліковий запис

Для доступу до всіх вкладок необхідно перейти на сторінку готелів та вибрати активний:

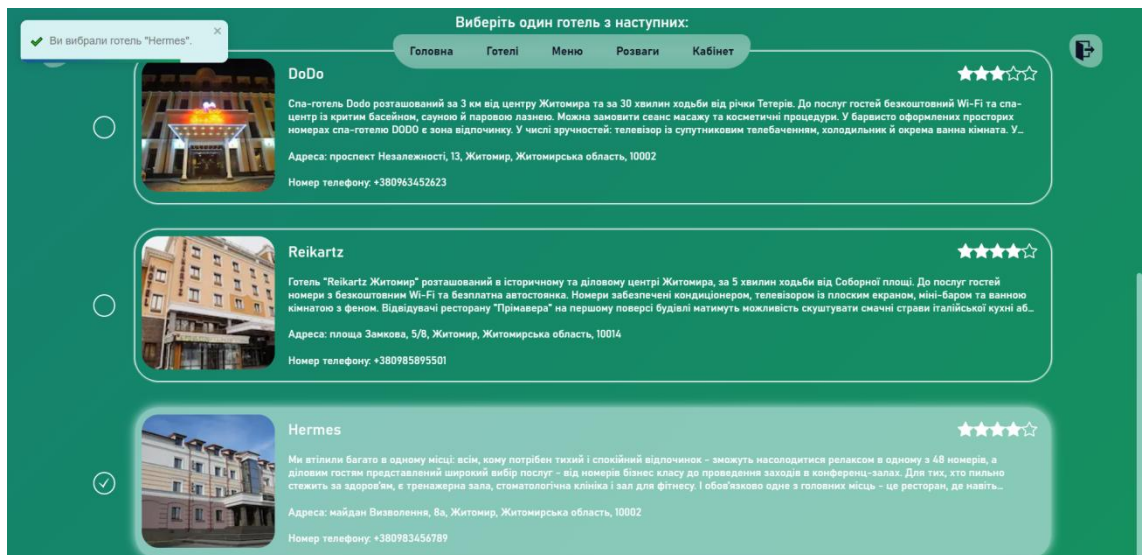


Рисунок 3.16 – Вибір активного готелю

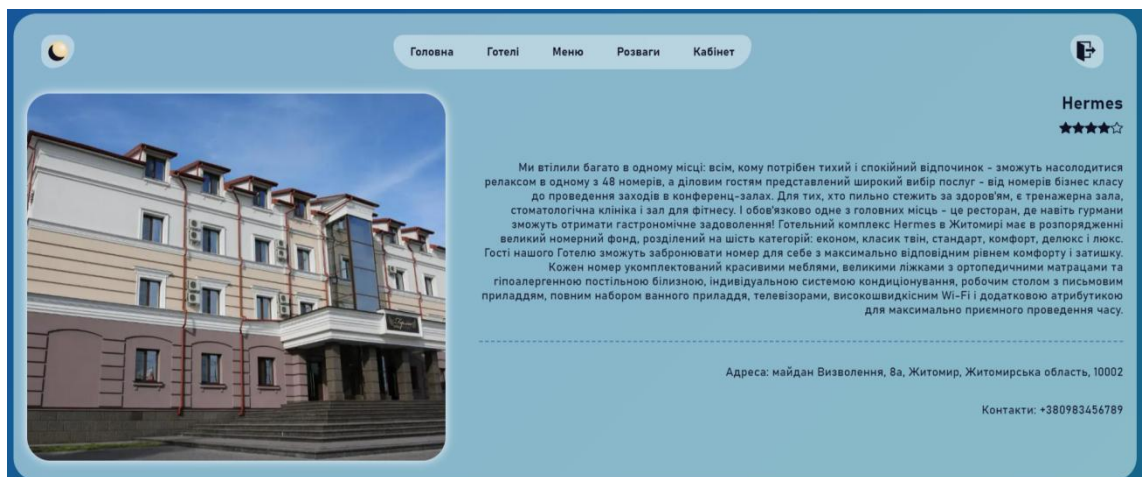


Рисунок 3.17 – Відображення активного готелю

Тепер користувачу доступні вкладки “Меню” та “Розваги”. Відкривши вкладку “Меню”, ми одразу побачимо можливе фільтрування, а також пошук:



Рисунок 3.18 – Сторінка “Меню”

При виборі страви за деякими продуктами, користувач може швидко шукати бажану страву:

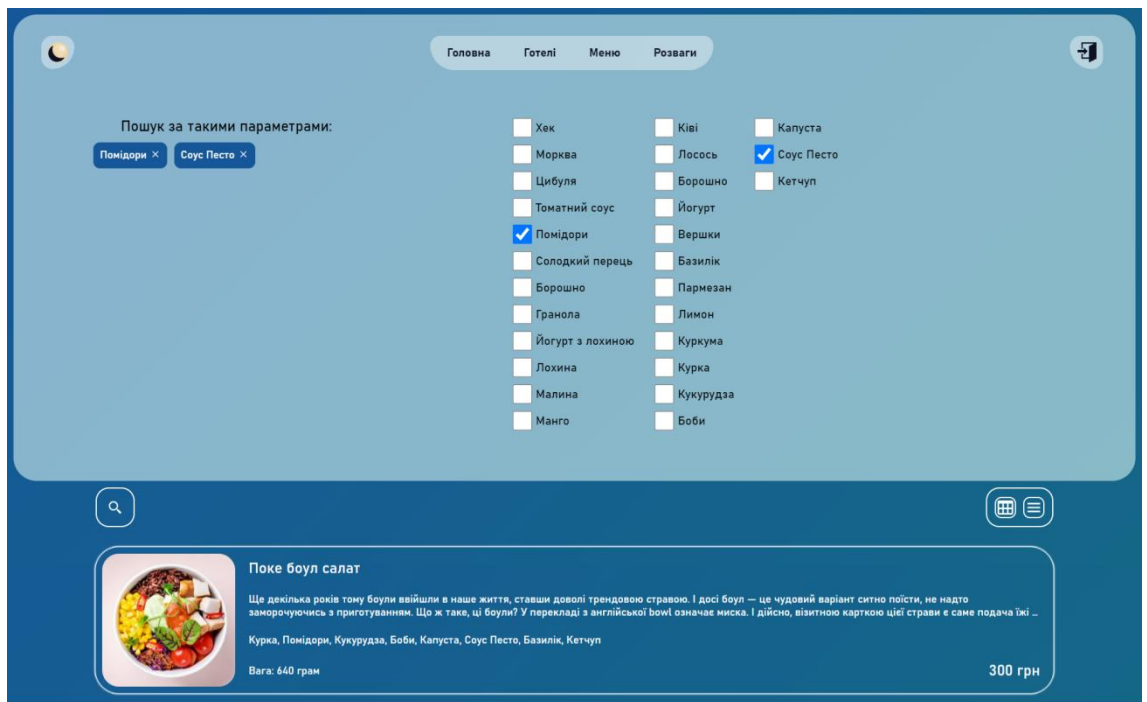


Рисунок 3.19 – Відфільтровані страви

Також можна зробити пошук серед страв та змінити відображення з списку на картки:

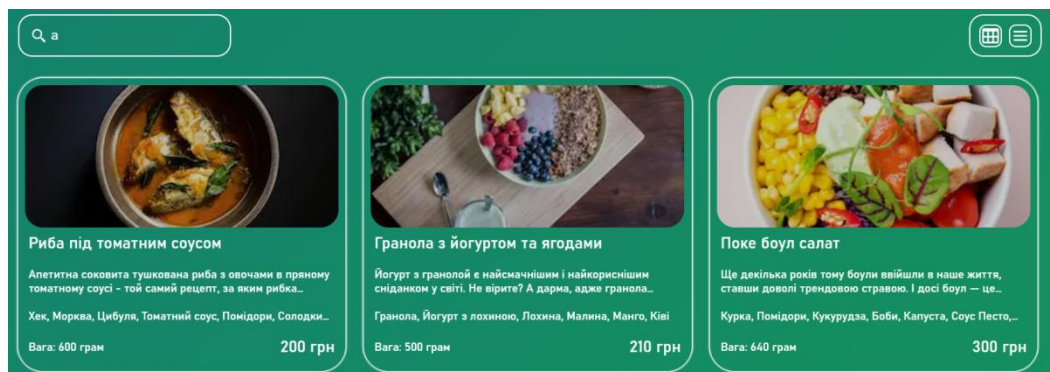


Рисунок 3.20 – Вигляд страв

Тепер перейдемо до сторінки конкретної страви. На цій сторінці (рис. 3.21) можна переглядати інформацію про страву, її відгуки, а також замовляти заздалегідь.

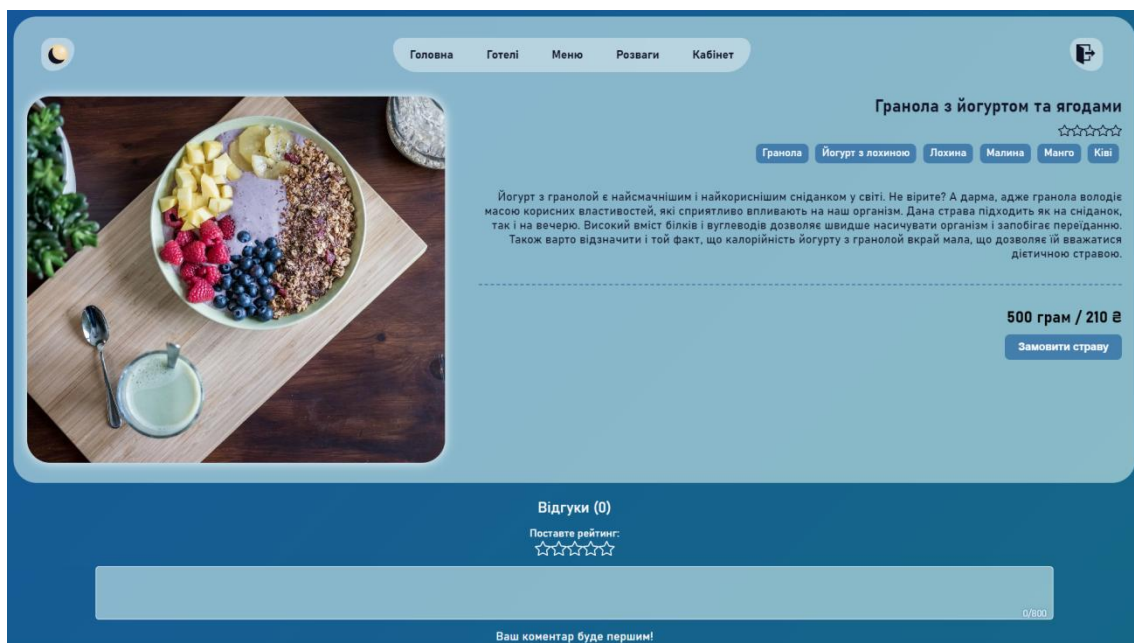


Рисунок 3.21 – Вигляд сторінки страви

Тепер перейдемо до замовлення страви. Для того, щоб це зробити, користувачу необхідно бути авторизованим, а також щоб у власника готелю був підв'язаний Телеграм бот.

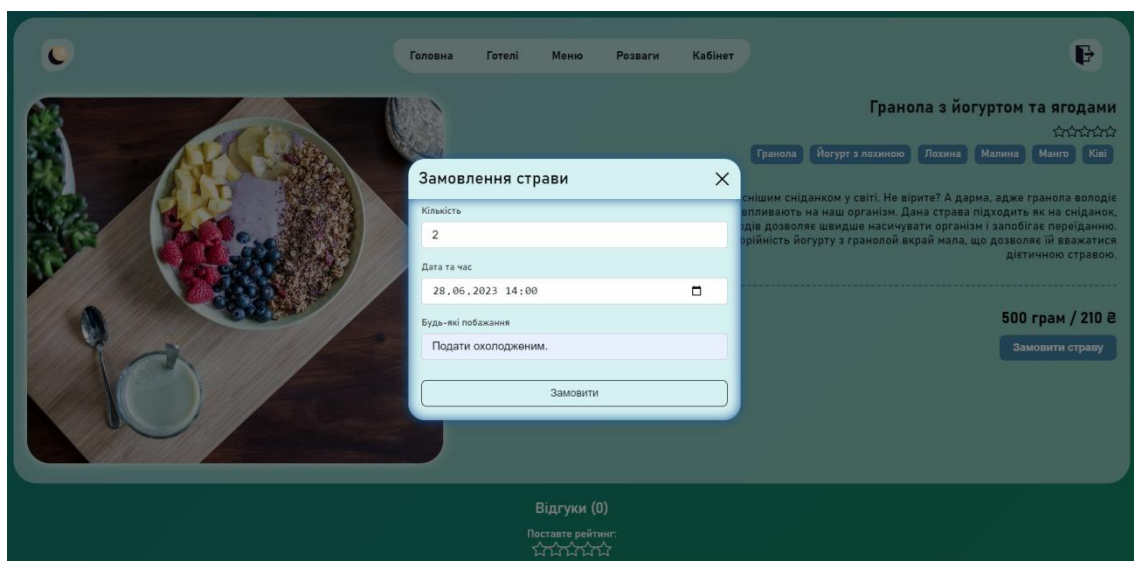


Рисунок 3.22 – Замовлення страви

Якщо все успішно пройде, то користувач отримає позитивне повідомлення (рис. 3.23), а для власника готелю у Телеграм бот прийде повідомлення (рис. 3.24).

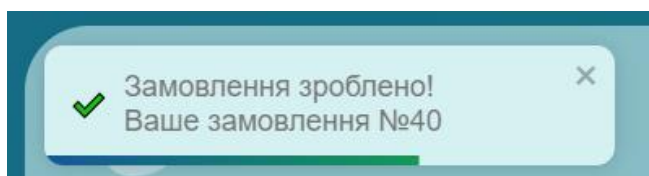


Рисунок 3.23 – Користувач успішно зробив замовлення



Рисунок 3.24 – Повідомлення у Телеграм боті

Тепер переглянемо розважальні заходи. Перейшовши на сторінку “Розваги”, вона буде схожою на сторінку “Меню”. Розважальні заходи тут можуть бути як активними, так і ні. Ті, що зазначені сірим кольором, є неактивними. На них вже неможливо зробити запис.

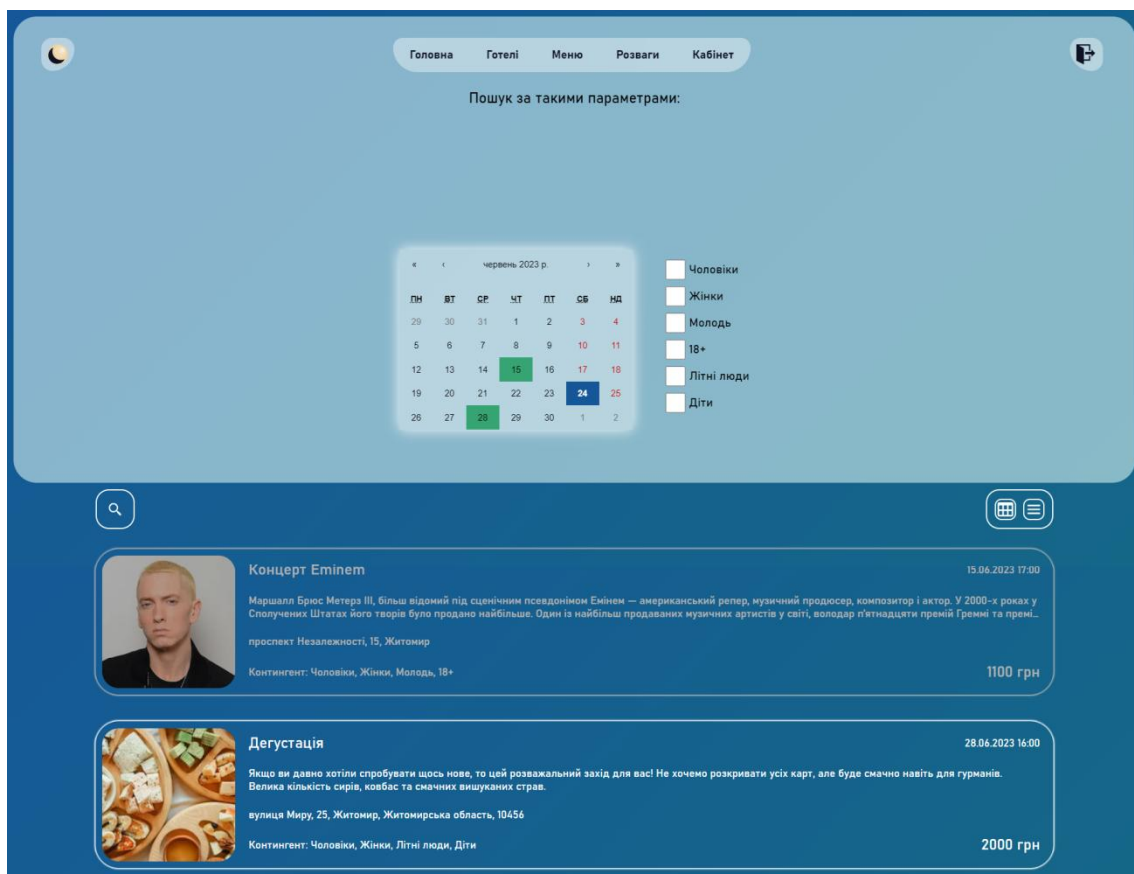


Рисунок 3.25 – Сторінка розважальних заходів

Тут також можна робити фільтрацію, але вже по двом параметрам: даті та контингенту. Спробуємо зробити фільтрацію:

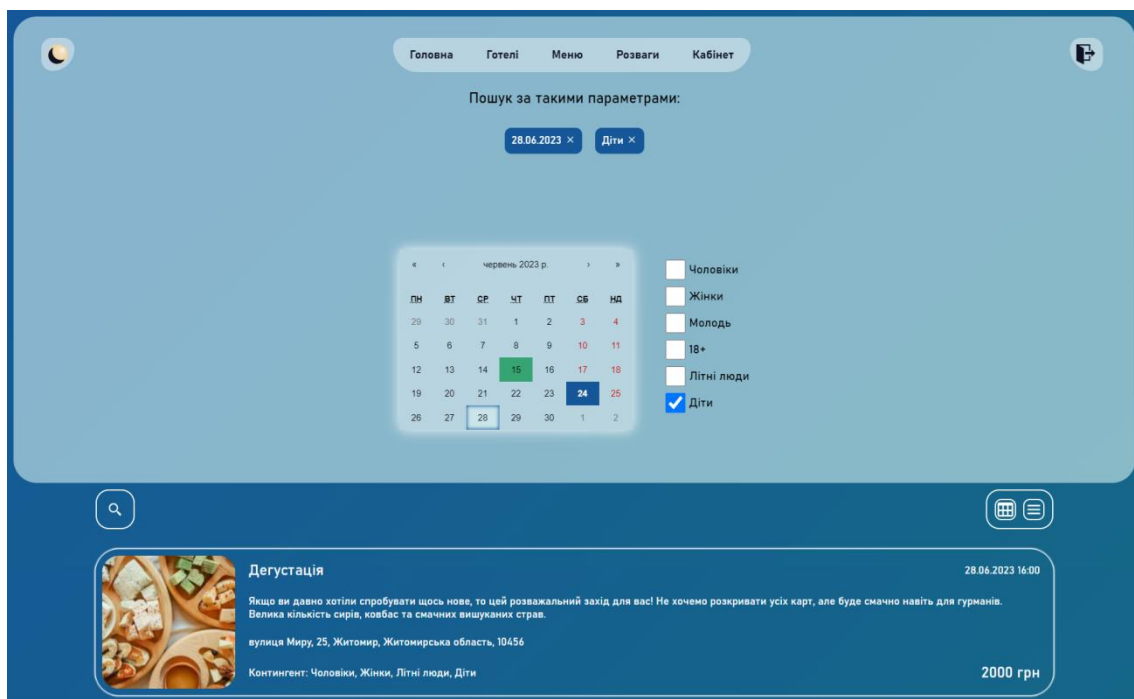


Рисунок 3.26 – Відфільтрована розвага

Тепер перейдемо до сторінки конкретної розваги. Знову ж таки, зробити запис на розвагу можна тільки якщо користувач авторизований та власник готелю має Телеграм бот. При натисканні на кнопку “Записатися на розвагу”, користувачу прийде відповідне повідомлення (рис. 3.27), а власнику готелю відповідно повідомлення у Телеграм бот (рис. 3.28).

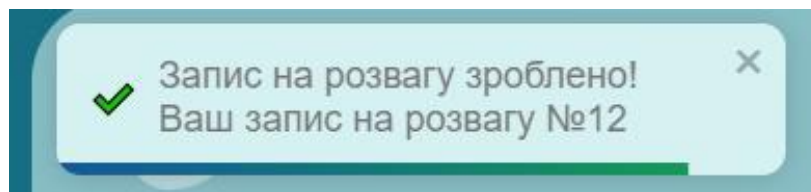


Рисунок 3.27 – Користувач успішно записався на розвагу

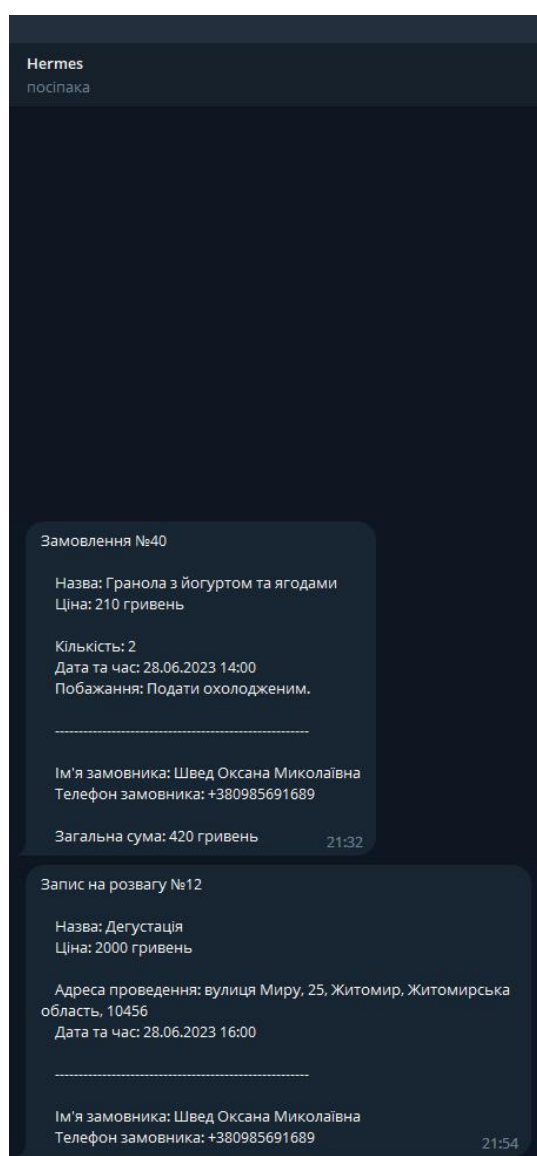


Рисунок 3.28 – Власник готелю отримав повідомлення про запис користувача на розвагу

Коли користувач вже записаний на розважальний захід, він може поставити собі нагадування у Google Calendar при натисканні на кнопку із дзвіночком:

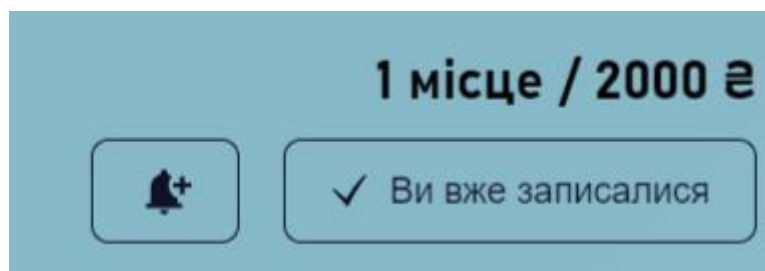


Рисунок 3.29 – Можливість встановлення нагадування

При натисканні на кнопку користувача просять увійти в обліковий запис Google:

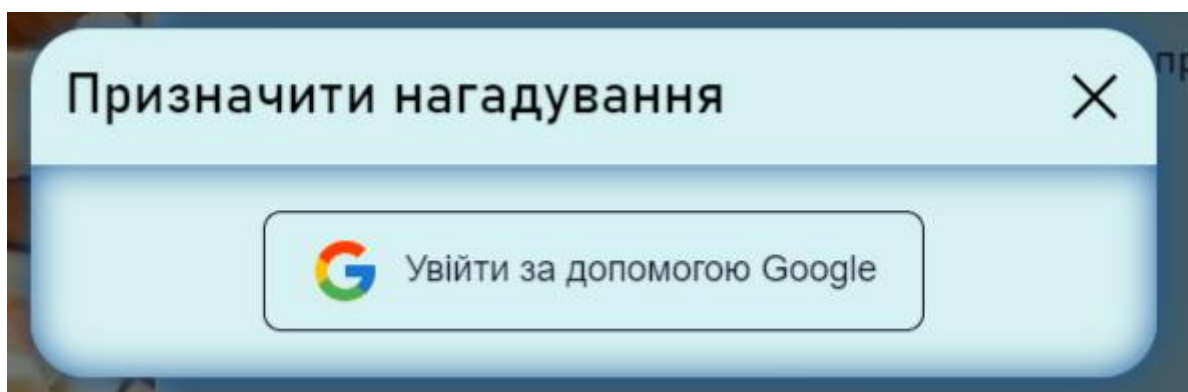


Рисунок 3.30 – Вхід за допомогою Google

Щоб увійти в обліковий запис Google, потрібно обрати обліковий запис:

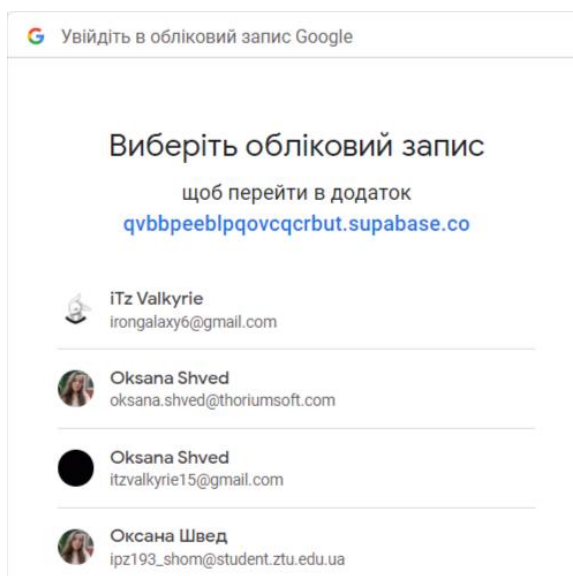


Рисунок 3.31 – Обрання облікового запису

Тепер користувач увійшов до облікового запису Google та може поставити нагадування до свого календаря Google:

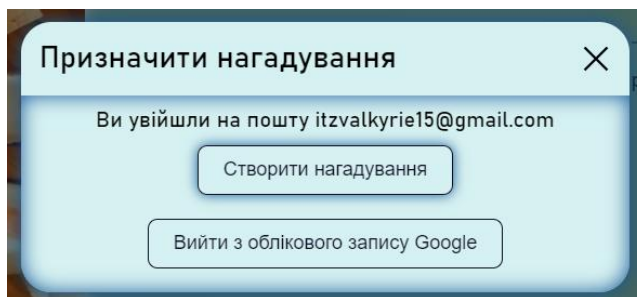


Рисунок 3.32 – Створення нагадування

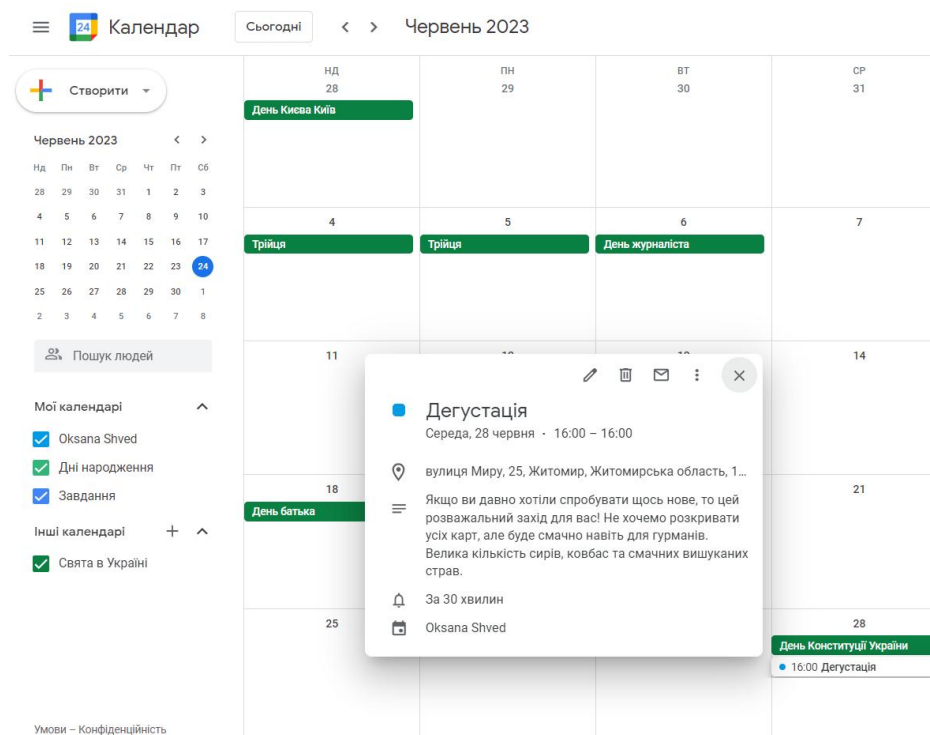


Рисунок 3.33 – В Google Calendar з'явилася подія

Тепер кнопка змінила свій вигляд та користувачу видно, що нагадування вже активне:



Рисунок 3.34 – Нагадування поставлено

Тепер перейдемо до кабінету користувача. У ньому ми можемо змінювати власні дані, змінювати пароль, головну фотографію та видаляти її.

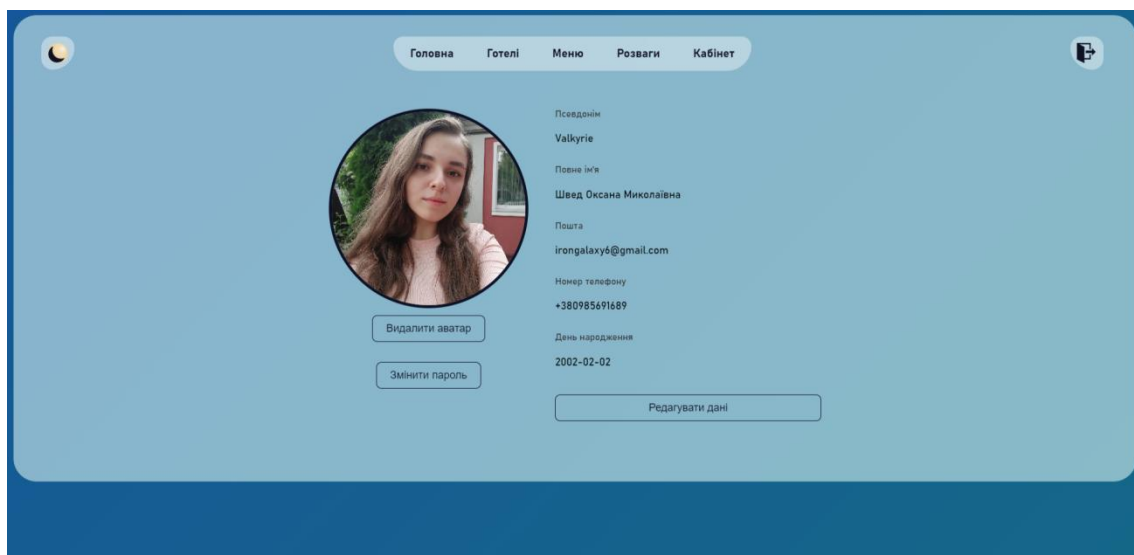


Рисунок 3.35 – Кабінет звичайного користувача

Отож, спробуємо змінити фотографію.

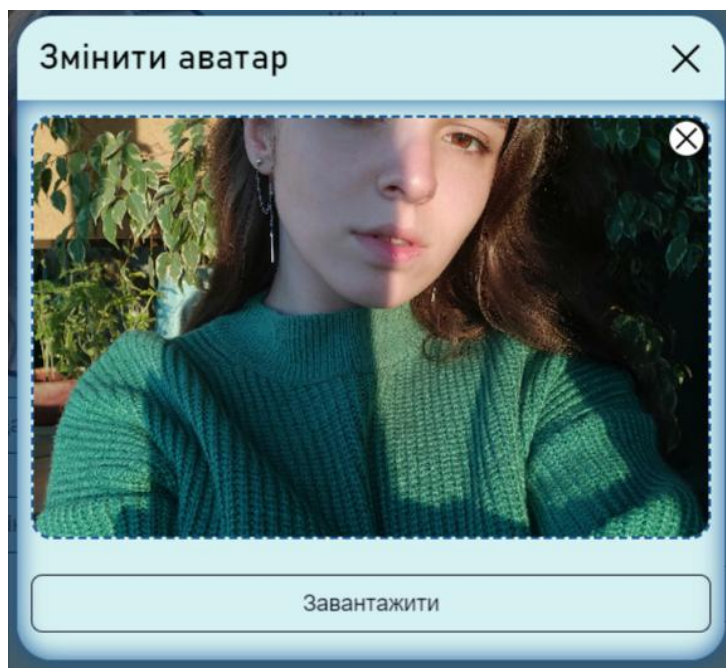


Рисунок 3.36 – Зміна фотографії на іншу

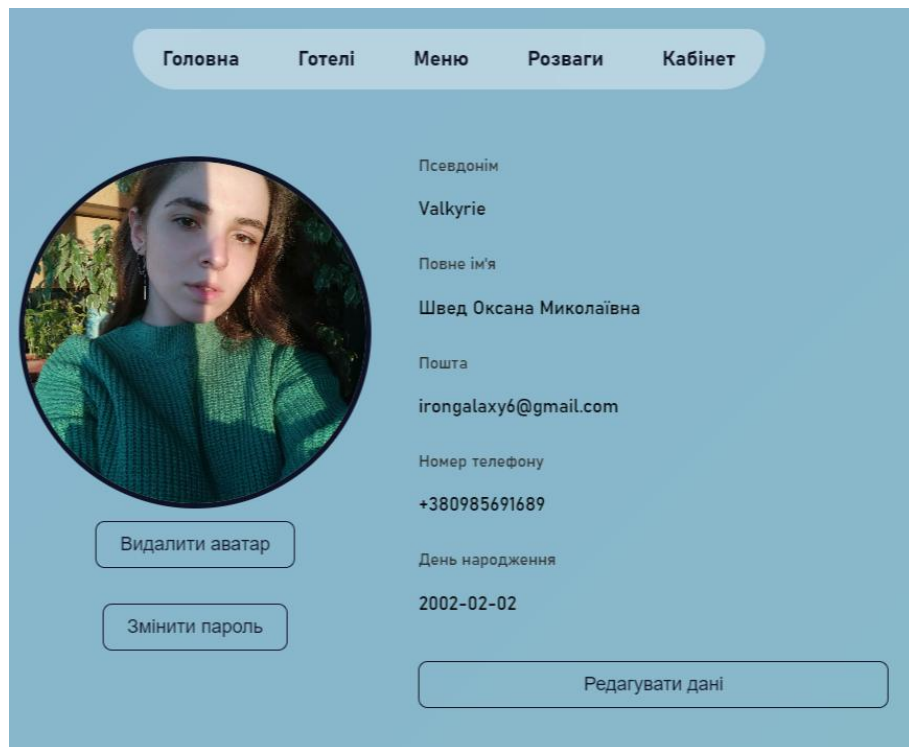


Рисунок 3.37 – Фотографія успішно змінена

Тепер видалимо головну фотографію:

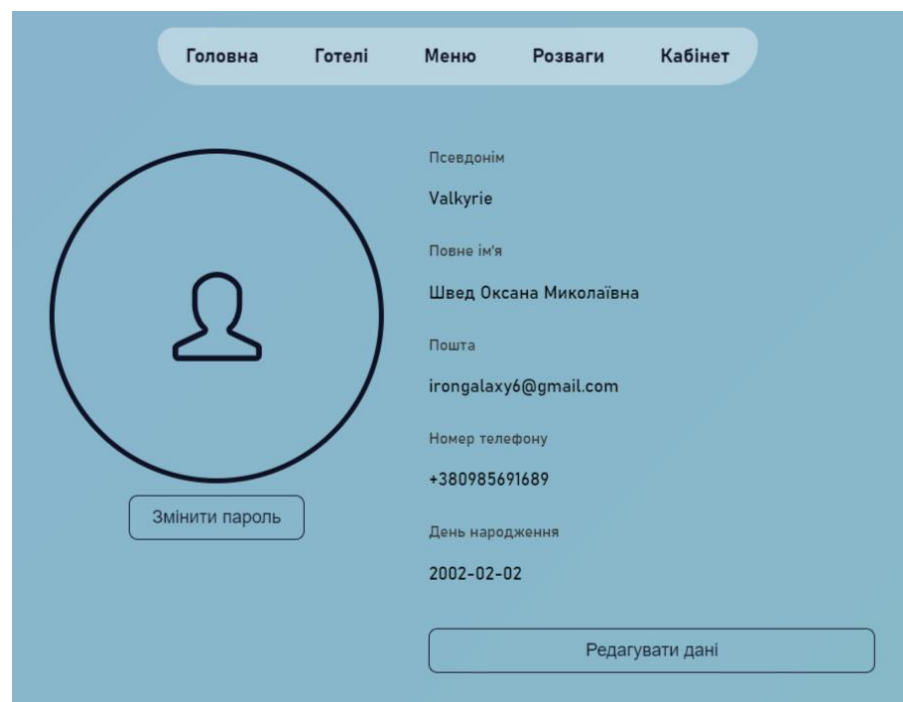


Рисунок 3.38 – Фотографія видалена

Тепер спробуємо змінити свої особисті дані:

Рисунок 3.39 – При валідації виникають помилки, тому збереження
НЕМОЖЛИВЕ

Рисунок 3.40 – Дані успішно змінено

Тепер змінимо пароль:

Рисунок 3.41 – Процес зміни паролю

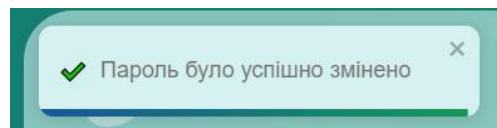


Рисунок 3.40 – Пароль змінено

Останньою дією зробимо вихід з облікового запису користувача та потрапимо на головний екран, а також доступ до кабінету буде більше недоступним до наступного входу:

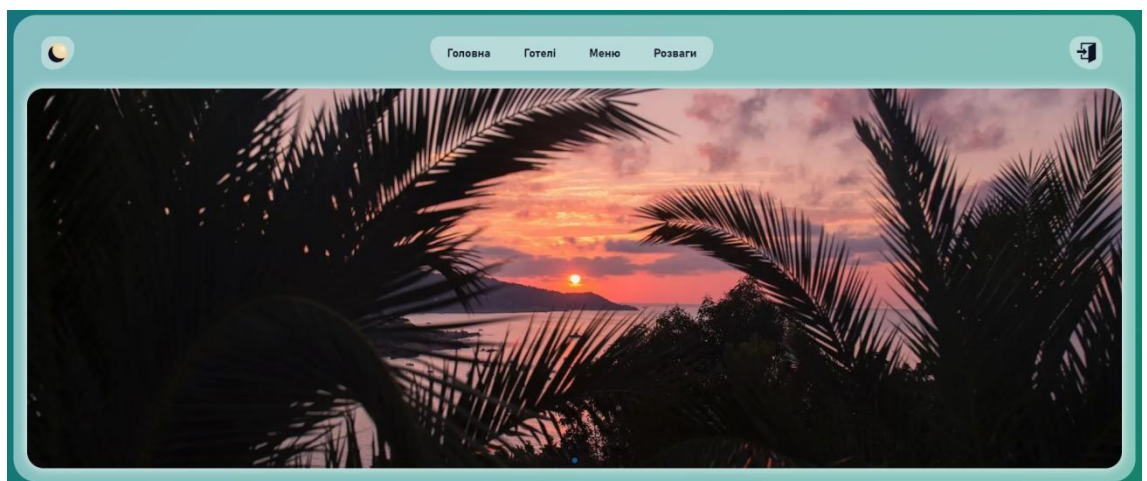


Рисунок 3.42 – Користувач не авторизований

3.3. Тестування роботи веб-додатку

Тестування відіграє важливу роль у забезпеченні якості та надійності додатку, шляхом виявлення й усунення помилок, недоліків та потенційних

проблем. Крім того, тестування дозволяє перевірити, чи задовольняє додаток вимогам та очікуванням користувачів.

Важливо проводити тестування під час розробки додатку, а також повне тестування функціоналу після завершення розробки. Після виявлення та виправлення помилок додаток ще раз піддавався тестування.

Існують різні техніки тестування [26], але використовувались найпростіші, адже у додатку немає складних проблем чи недоліків, їх всі можливо просто виявити. Тестування допомагає забезпечити високу якість програмного забезпечення та задоволення вимог користувачів. Воно спрямоване на перевірку правильності роботи окремих функцій, модулів або системи в цілому.

Більшість помилок зручно виявляти за допомогою техніки “передбачення помилок”, а для форм з цифровими значеннями – техніки “аналізу граничних значень”. Аналіз граничних значень спирається на припущення, що помилки найчастіше виникають саме на границях діапазону вхідних даних. Тому, якщо програма правильно працює для найменших та найбільших значень, ймовірно, вона також буде працювати правильно для інших значень всередині діапазону.

Таким чином, було знайдено дефекти клієнтської частини додатку:

передумови: користувач хоче змінити свої дані у кабінеті.

кроки:

1. Користувач входить до свого особистого кабінету.
2. Користувач натискає на кнопку “Редагувати дані”.
3. Користувач вводить дані у бажані поля форми та натискає на кнопку “Зберегти”.
4. Користувач знову натискає на кнопку “Редагувати дані”.
5. Натискає на кнопку “Відмінити”.

очікуваний результат: всі значення полів форми стають такими, що були до натискання кнопки “Редагувати дані”.

фактичний результат: всі значення полів форми стають такими, якими були ще тоді, коли користувач тільки увійшов в особистий кабінет.

Проблему було виправлено.

Під час розробки функціоналу додатку було складено набір тест-кейсів, які використовуються для перевірки коректності роботи основних функцій додатку.

1. Авторизація:

передумови: користувач бажає увійти до особистого кабінету, коли у нього вже є раніше створений обліковий запис.

кроки:

1. Користувач відкриває модальне вікно з входом до особистого кабінету.

2. Користувач вводить свою електронну пошту та пароль.

очікуваний результат: користувач успішно авторизується в системі та зберігається у сесії зі своїм унікальним токеном.

фактичний результат: користувач успішно авторизується в системі та зберігається у сесії зі своїм унікальним токеном.

2. Зміна особистої фотографії користувача:

передумови: користувач має особистий кабінет.

кроки:

1. Користувач входить до особистого кабінету.

2. Користувач натискає на особисту фотографію.

3. У модальному вікні користувач натискає на поле для зміни фотографії.

4. Користувач вибирає бажану фотографію.

5. Користувач натискає на кнопку “Завантажити”.

очікуваний результат: фотографія через декілька секунд змінюється на вибрану.

фактичний результат: фотографія через декілька секунд змінюється на вибрану.

3. Користувач бажає поставити нагадування на розважальний захід у Google Calendar:

передумови: користувач має бути вже записаним до вибраного розважального заходу.

кроки:

1. Користувач заходить на сторінку розважального заходу, де вже записаний.
2. Користувач натискає на кнопку нагадування.
3. У модальному вікні користувач входить до облікового запису Google.
4. Після успішного входу користувач знову натискає на кнопку нагадування.
5. Користувачу тепер доступно створити нагадування для даного розважального заходу.

очікуваний результат: при натисканні на створення нагадування у календарі користувача з'являється нагадування на дату та число, яке вказане у розважальному заході.

фактичний результат: при натисканні на створення нагадування у календарі користувача з'являється нагадування на дату та число, яке вказане у розважальному заході.

4. Надсилення коментаря для страви:

передумови: користувач має бути авторизованим.

кроки:

1. Користувач вибирає потрібну страву.
2. Знизу є форма для вводу коментаря, користувач записує його.
3. Користувач натискає на кнопку “Зберегти”.

очікуваний результат: коментар одразу з'являється нижче під формою.

фактичний результат: коментар одразу з'являється нижче під формою.

5. Вихід з облікового запису користувача:

передумови: користувач має бути авторизованим.

кроки:

1. Користувач у правому верхньому куті натискає на кнопку виходу.

очікуваний результат: користувач виходить з системи, у шапці зникає вкладка “Кабінет”, якщо користувач був на вкладці “Кабінет”, його перенаправляє на головну сторінку.

фактичний результат: користувач виходить з системи, у шапці зникає вкладка “Кабінет”, якщо користувач був на вкладці “Кабінет”, його перенаправляє на головну сторінку.

Висновки до розділу 3

У даному розділі було наведено послідовність кроків щодо встановлення та налаштування веб-додатку. Розглянуто як роботу з панеллю адміністратора, так і з основною частиною додатку. Детально розглянуто елементи інтерфейсу додатку та їх функціонал.

Додаток також був протестований з метою виявлення дефектів і підвищення рівня впевненості в якості веб-додатку. Створено різноманітні тест-кейси для спрощення пошуку дефектів. Виявлені дефекти було виправлено і знову протестовано, щоб переконатися у їх виправленні.

ВИСНОВКИ

В процесі виконання кваліфікаційної роботи бакалавра було реалізовано веб-додаток планування розваг та меню готелів.

Для початку, було сформована та проаналізована задача. Було проведено аналіз предметної області та порівняльний аналіз наявних аналогічних додатків з метою виявлення їх переваг та недоліків. Під час аналізу не вдалося знайти жодного додатку, що повністю відповідав було поставленим вимогам та задачам. З метою реалізації поставленої задачі було обрано монолітну архітектуру для серверної частини та модульну архітектуру для клієнтської частини, оскільки ці підходи виявилися найбільш оптимальними. Для реалізації веб-додатку було визначено середовище виконання Node.JS, а також обрані фреймворки Strapi та Next.JS як основні засоби.

Також були сформульовані та проаналізовані функціональні та нефункціональні вимоги до системи. Були визначені різні варіанти використання системи, а також визначені компоненти, що складатимуть систему, та з'ясовані зв'язки між ними. Сутності системи, які представляють дані, що будуть зберігатися в системі, також було спроектовано. Була розроблена база даних на основі цих сутностей, що визначили структуру та організацію даних у системі.

Був сформульований основний алгоритм роботи додатку, а також були розроблені алгоритми для різних частин функціоналу. Додаток планування розваг та меню готелів був спроектований та повністю реалізований. Була надана реалізація окремого компонента додатку, який відповідає за роботу з файлами та їх завантаження на сервер. Був наданий опис процесу створення сторінки клієнтської частини веб-додатку.

Була розроблена система, яка відповідає всім вимогам до додатку. Ця система включає функціонал для спрощення планування розваг та меню готелів, зручне додавання нових записів для власника готелю, а для користувачів зручний перегляд, коментування, запис та встановлення нагадувань.

Інтерфейс додатку є адаптивним і доступним на різних розширеннях екрану.

Також був сформований порядок встановлення та налаштування веб-додатку. Додаток пройшов тестування, під час якого були виявлені і виправлені проблеми.

Веб-додаток може бути налаштований та використовуваний для різних компаній. Реалізований веб-додаток може бути використаний для планування для будь-якого готелю.

ЛІТЕРАТУРА

1. Next.JS / The React Framework for the Web [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/>
2. Digital Ocean / Understanding Asynchronous Redux Actions with Redux Thunk [Електронний ресурс] – Режим доступу до ресурсу: <https://www.digitalocean.com/community/tutorials/redux-redux-thunk>
3. Supabase / Build in a weekend Scale to millions [Електронний ресурс] – Режим доступу до ресурсу: <https://supabase.com/>
4. Log Rocket / The guide to adding Google login to your React app [Електронний ресурс] – Режим доступу до ресурсу: <https://blog.logrocket.com/guide-adding-google-login-react-app/>
5. Console Cloud Google / Working in Google Calendar [Електронний ресурс] – Режим доступу до ресурсу: <https://console.cloud.google.com/>
6. Google Calendar API / Google Calendar API overview [Електронний ресурс] – Режим доступу до ресурсу: <https://developers.google.com/calendar/api/guides/overview>
7. Npm / React-Toastify allows you to add notifications [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/react-toastify>
8. Sanity / Add the Telegram Messaging API to your React application [Електронний ресурс] – Режим доступу до ресурсу: <https://www.sanity.io/guides/add-telegram-to-your-react-application>
9. Strapi / Strapi's documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.strapi.io/>
10. Strapi / Filtering [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.strapi.io/dev-docs/api/query-engine/filtering>
11. Strapi / Bulk Operations [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.strapi.io/dev-docs/api/query-engine/bulk-operations>

12. Strapi / REST API: Sort & Pagination [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.strapi.io/dev-docs/api/rest/sort-pagination>
13. Strapi / Populating [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.strapi.io/dev-docs/api/query-engine/populating>
14. Npm / React-icons [Електронний ресурс] – Режим доступу до ресурсу: <https://www.npmjs.com/package/react-icons>
15. Next.JS / Component Image — Architectural pattern, controllers, providers, and modules [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs/pages/api-reference/components/image>
16. Styled-Components / Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://styled-components.com/docs>
17. Styled-Components / Passed props [Електронний ресурс] – Режим доступу до ресурсу: <https://styled-components.com/docs/basics#passed-props>
18. Strapi / Authentication [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.strapi.io/dev-docs/plugins/users-permissions#authentication>
19. React.JS / Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/>
20. Next.JS / Server-side Rendering (SSR) [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs/pages/building-your-application/rendering/server-side-rendering>
21. Node.JS / Node.JS — це JavaScript-оточення побудоване на JavaScript-рушієві Chrome V8 [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/uk>
22. TechTarget / DEFINITION single-page application (SPA) [Електронний ресурс] – Режим доступу до ресурсу: <https://www.techtarget.com/whatis/definition/single-page-application-SPA>
23. Medium / Single-page application vs. multiple-page application [Електронний ресурс] – Режим доступу до ресурсу: <https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58>

24. Telegram Bot API / Telegram Bot API Documentation [Електронний ресурс]
– Режим доступу до ресурсу: <https://core.telegram.org/bots/api>
25. Medium / Write a simple Telegram Bot in Next.js [Електронний ресурс] –
Режим доступу до ресурсу: <https://blog.devgenius.io/write-a-simple-telegram-bot-in-next-js-b49379e77163>
26. QATestLab / ГЛОСАРІЙ ЛЕКЦІЙ №5 «ТЕСТ-ДИЗАЙН. ТЕСТ-КЕЙСИ» З
КУРСУ «ОСНОВИ ТЕСТУВАННЯ ПЗ» [Електронний ресурс] – Режим
доступу до ресурсу: <https://training.qatestlab.com/blog/course-materials/glossary-test-design/>
27. Bootstrap / Breakpoints [Електронний ресурс] – Режим доступу до ресурсу:
<https://getbootstrap.com/docs/5.0/layout/breakpoints/>
28. React.JS / Огляд хуків [Електронний ресурс] – Режим доступу до ресурсу:
<https://uk.legacy.reactjs.org/docs/hooks-overview.html>
29. ESLint / Find and fix problems in your JavaScript code [Електронний ресурс]
– Режим доступу до ресурсу: <https://eslint.org/>
30. Tripadvisor / Main Page [Електронний ресурс] – Режим доступу до
ресурсу: <https://www.tripadvisor.com/>
31. Airbnb / Main Page [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.airbnb.com.ua/>
32. OpenTable / Main Page [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.opentable.com/>

ДОДАТКИ

					ПЗ.КР.Б – 121 – 23 – ПЗ	Арк.
						76

Лістинг pages/api/telegram.js (власноруч створений API для зв'язку з Telegram Bot):

```
import TelegramBot from 'node-telegram-bot-api';

export default async (req, res) => {
  if (req.method === 'POST') {
    const { message, botToken, chatId } = req.body;

    const bot = new TelegramBot(botToken);

    try {
      await bot.sendMessage(chatId, message);
      res.status(200).end();
    } catch (error) {
      res.status(500).end();
    }
  } else {
    res.setHeader('Allow', 'POST');
    res.status(405).end('Method Not Allowed');
  }
};
```

Лістинг utils/telegram.js (для зв'язку з власноруч створеним API Telegram Bot):

```
import React from 'react';
import axios from 'axios';
import { notify } from '@utils/tooltipToastify';

const sendMessage = async (message, botToken, chatId) => {
  try {
    await axios.post('/api/telegram', { message, botToken, chatId });

    return { status: 200 };
  } catch (error) {
    notify(
      <>
      <p>Щось пішло не так з надсиланням Вашого замовлення...</p>
      <p>Спробуйте ще раз.</p>
      </>,
      '',
    );

    return { status: 500 };
  }
};
```

```
export { sendMessage };
```

Лістинг utils/validation.js (для валідації форм):

```
export const validationEmail = (value) => {
  return /^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i.test(value);
};

export const validationPassword = (value) => {
  return /^(?=.*\d)(?=.*[!@#$%^&*])[\w!@#$%^&*]{8,50}$/i.test(value);
};

export const validationPhoneNumber = (value) => {
```

					ІПЗ.КР.Б – 121 – 23 – ІЗ	Арх.
						77

```

    return /^+380\d{9}$/.test(value);
};

export const validationFullName = (value) => {
    return /^[A-ЯІЇЄґ][a-яіієґ']*?(?:-[A-ЯІЇЄґ][a-яіієґ']*?)?(?:[A-ЯІЇЄґ][a-яіієґ']*?)?{0,2}$/.test(
        value,
    );
};

export const validationDateServer = (value) => {
    return /^\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}\.\d{3}Z$/.test(value);
};

```