НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___» _____ 2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Платформа для онлайн-резервування місць у закладах харчування»

Виконав:
студент IV курсу, групи КП-02
Якобчук Роман Валерійович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Катін Павло Юрійович _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри ОТ ФІОТ, к.т.н, доцент,
Долголенко Олександр Миколайович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Євгенія СУЛЕМА
«___»_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту
Якобчуку Роману Валерійовичу

1. Тема проєкту «Платформа для онлайн-резервування місць у закладах харчування», керівник проекту Катін Павло Юрійович, доцент кафедри ПЗКС, к.т.н., доцент, затверджені наказом по університету від «30» травня 2024 р. №2205-С.

2. Термін подання студентом проєкту «14» червня 2024 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   – огляд існуючих програмних рішень;
   – обґрунтування вибору засобів розроблення;
   – структурна організація програмних засобів;
   – аналіз розробленої платформи;

5. Перелік обов'язкового графічного матеріалу:
   – структура бази даних. ER-діаграма;
   – алгоритм перевірки вільних місць (креслення);
   – архітектура платформи (плакат);
   – інфраструктура платформи(плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2023 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	13.11.2023	
2.	Розроблення та узгодження технічного завдання	22.11.2023	
3.	Розроблення структури онлайн-платформи	15.12.2023	
4.	Підготовка матеріалів першого розділу дипломного проєкту	28.12.2023	
5.	Розроблення дизайну сторінок та графічних елементів	01.02.2024	
6.	Підготовка матеріалів другого розділу дипломного проєкту	19.02.2024	
7.	Програмна реалізація онлайн-платформи	11.03.2024	
8.	Тестування онлайн-платформи	18.03.2024	
9.	Підготовка матеріалів третього розділу дипломного проєкту	27.03.2024	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	15.04.2024	
11.	Підготовка графічної частини дипломного проєкту	22.04.2024	
12.	Оформлення документації дипломного проєкту	27.05.2024	

Студент Роман ЯКОБЧУК

Керівник проєкту Павло КАТІН

АНОТАЦІЯ

Даний дипломний проект присвячений створенню онлайн-платформи для бронювання місць у закладах харчування.

Платформа являє собою веб-застосунок, який передбачає три типи користувачів: звичайних користувачів, менеджерів закладів та адміністраторів платформи. Звичайні користувачі можуть реєструватися на платформі, переглядати список закладів із детальною інформацією про них, додавати заклади до вибраного, переглядати новини та спілкуватися з менеджерами закладів у режимі реального часу через вбудовані чати. Головною функцією для цих користувачів є можливість бронювання місць у закладах із вибором дати та часу відвідування. Менеджери закладів можуть реєструвати свої заклади на платформі, додавати, редагувати та видаляти інформацію про них, публікувати новини, переглядати бронювання та взаємодіяти з користувачами через чати. Адміністратори платформи мають повні права на управління контентом, користувачами та налаштуваннями системи.

На платформі впроваджено комплексну систему безпеки та керування доступом. Доступ до даних та функціоналу платформи мають виключно зареєстровані користувачі після проходження процедури аутентифікації. Права доступу для різних типів зареєстрованих користувачів чітко розмежовані, обмежуючи можливості залежно від ролі. Конфіденційність облікових даних користувачів гарантується зберіганням паролів у зашифрованому вигляді за допомогою криптографічних хеш-функцій.

У даному дипломному проекті розроблено архітектуру онлайн-платформи, алгоритми взаємодії між клієнтською та серверною частинами, процедури обробки бронювань, комунікації між користувачами, а також дизайн та зручність використання веб-інтерфейсу.

ABSTRACT

This diploma project is dedicated to creating an online platform for online reservations of seats in catering establishments.

The platform is a web application that provides for three types of users: ordinary users, establishment managers, and platform administrators. Ordinary users can register on the platform, view a list of establishments with detailed information about them, add establishments to their favorites, view news, and communicate with establishment managers in real-time through built-in chats. The main function for these users is the ability to book seats at establishments by selecting the date and time of their visit. Establishment managers can register their establishments on the platform, add, edit, and delete information about them, publish news, view bookings, and interact with users through chats. Platform administrators have full rights to manage content, users, and system settings.

The platform implements a comprehensive security and access management system. Only registered users have access to the platform's data and functionality after going through the authentication procedure. Access rights for different types of registered users are clearly delineated, limiting capabilities depending on the role. The confidentiality of user account data is guaranteed by storing passwords in encrypted form using cryptographic hash functions.

In this diploma project, the architecture of the online platform, algorithms for interaction between the client and server parts, procedures for processing bookings, communication between users, as well as the design and usability of the web interface have been developed.

ДП.045440-01-90 Платформа для онлайн-резервування місць у закладах харчування.
Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Платформа для	5	
	онлайн-резервування		
	місць у закладах		
	харчування.		
	Технічне завдання		
ДП.045440-03-81	Платформа для	67	
	онлайн-резервування		
	місць у закладах		
	харчування.		
	Пояснювальна записка		
ДП.045440-04-51	Платформа для	4	
	онлайн-резервування		
	місць у закладах		
	харчування.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Платформа для	27	
	онлайн-резервування		
	місць у закладах		
	харчування.		
	Керівництво користувача		
ДП.045440-06-99	Платформа для	1	
	електронної освіти.		
	Резервування місць.		
	Структура бази даних.		
	ER-діаграма		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-07-99	Платформа для	1	
	онлайн-резервування		
	місць у закладах		
	харчування.		
	Алгоритм перевірки		
	вільних місць.		
	Блок-схема алгоритму		
ДП.045440-08-98	Платформа для	1	
	онлайн-резервування		
	місць у закладах		
	харчування.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2023 р.

ПЛАТФОРМА ДЛЯ ОНЛАЙН-РЕЗЕРВУВАННЯ МІСЦЬ У ЗАКЛАДАХ ХАРЧУВАННЯ

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Павло КАТІН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Роман ЯКОБЧУК

2023

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: платформа для онлайн-резервування місць у закладах харчування.

Галузь застосування: інформаційні технології, сфера громадського харчування.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для оптимізації та автоматизації процесу бронювання місць у закладах громадського харчування з метою підвищення ефективності їх роботи та надання зручного онлайн-сервісу для відвідувачів.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Онлайн-платформа повинна забезпечувати такі основні функції:

1. Аутентифікація та авторизація користувача, процедуру відновлення пароля.
2. Підтримка трьох ролей користувачів: Звичайний користувач, менеджер закладу, адміністратор платформи.
3. Для Відвідувачів: пошук та перегляд інформації про заклади харчування, можливість додавати заклади до вибраного, переглядати новини, здійснювати онлайн-бронювання

столиків/місць з вибором дати та часу, спілкуватися з менеджерами закладів через вбудовані чати.

4. Для Менеджерів: можливість реєструвати свій заклад, управляти інформацією про нього, публікувати новини, переглядати та обробляти бронювання, спілкуватися з відвідувачами через чати.
5. Для Адміністраторів: повний контроль та управління контентом платформи, менеджерами та звичайними користувачами.
6. Відображення місцезнаходження закладів на інтерактивній карті для зручного перегляду.
7. Забезпечення безпеки та конфіденційності даних шляхом аутентифікації за JWT, хешування паролів, розмежування прав доступу.

Додаткові вимоги:

1. Локалізація платформи українською та англійською мовами.
2. Адаптивний дизайн для різних пристроїв та розширень екрану.
3. Можливість перемикання між світлою та темною темами інтерфейсу для зручного перегляду в різний період доби.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1. Пояснювальна записка.
2. Програма та методика тестування.
3. Керівництво користувача.
4. Креслення:
   - «Структура бази даних. ER-діаграма»;
   - «Алгоритм перевірки вільних місць. Блок-схема алгоритму».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи..... 16.11.2023

Розроблення та узгодження технічного завдання..... 27.11.2023

Підготовка матеріалів першого розділу дипломного проєкту.....28.12.2023

Розроблення структури онлайн-платформи..... 17.01.2024

Підготовка матеріалів другого розділу дипломного проєкту..... 05.03.2024

Розроблення дизайну сторінок та графічних елементів..... 25.03.2024

Програмна реалізація та тестування онлайн-платформи.....15.04.2024

Підготовка третього розділу дипломного проєкту..... 25.04.2024

Підготовка четвертого розділу дипломного проєкту..... 07.05.2024

Підготовка матеріалів графічної частини проєкту..... 20.05.2024

Оформлення технічної документації проєкту.....27.05.2024

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2024 р.

ПЛАТФОРМА ДЛЯ ОНЛАЙН-РЕЗЕРВУВАННЯ МІСЦЬ У ЗАКЛАДАХ ХАРЧУВАННЯ

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Павло КАТІН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Роман ЯКОБЧУК

2024

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення.

IT (Information Technology) – інформаційні технології.

API (Application Programming Interface) – це набір правил, які визначають, як програми можуть взаємодіяти між собою, обмінюючись даними та виконуючи функції.

БД – база даних.

РБД (реляційна база даних) – це тип бази даних, яка організована на основі реляційної моделі даних. У реляційній базі даних дані зберігаються у вигляді таблиць, які складаються з рядків і стовпців.

CRUD (Create, Read, Update, Delete) – базові операції для роботи з даними в інформаційних системах.

HTTP (Hypertext Transfer Protocol) – протокол передачі даних для веб-ресурсів.

TSX – це розширення файлів у TypeScript, яке використовується для опису компонентів у React-додатках.

JSON (JavaScript Object Notation) – це простий формат обміну даними, заснований на JavaScript, зручний для читання та запису як для людей, так і для машин.

HTML (HyperText Markup Language) – це мова розмітки, яка використовується для створення та відображення веб-сторінок у браузері.

CSS (Cascading Style Sheets) – це мова стилів, яка використовується для визначення як елементи повинні відображатися на екрані.

REST (Representational State Transfer) – це архітектурний стиль для розробки програмного забезпечення, який використовує принципи створення, зчитування, оновлення та видалення ресурсів за допомогою стандартних HTTP-методів.

SPA (Single Page Application) – веб-додаток, що завантажується на одній сторінці та динамічно оновлює контент без перезавантаження.

UI (User Interface) – користувацький інтерфейс, засіб взаємодії між людиною та програмним забезпеченням.

UX (User Experience) – досвід взаємодії користувача з продуктом, що включає зручність, ефективність та задоволеність.

URL (Uniform Resource Locator) – це формат адреси для доступу до ресурсів в Інтернеті.

Адаптивний дизайн – підхід до дизайну веб-сайтів, який забезпечує оптимальне відображення контенту на різних пристроях і розмірах екранів.

Анімація – процес створення візуальних ефектів, які дають враження руху чи зміни.

Синхронізація – узгодження даних між різними джерелами або пристроями.

Фреймворк – це платформа для розробки програмного забезпечення, яка надає основний каркас, набір інструментів та бібліотек для створення додатків.

Бібліотека – це набір готових функцій або класів, які можна використовувати для вирішення конкретних задач.

BSON (Binary JavaScript Object Notation) – це двійковий формат подання даних, схожий на JSON, але призначений для зберігання та передачі даних в бінарному вигляді, що робить його більш компактним і ефективним.

Токен аутентифікації – це унікальний ключ, який використовується для підтвердження користувача та надання доступу до певних ресурсів або сервісів.

Хешування – це процес перетворення будь-якого довільного вхідного тексту у значення фіксованої довжини, яке називається хешем.

СКБД (система керування базами даних) – це програмне забезпечення, яке використовується для створення, управління та маніпуляції базами даних. СКБД забезпечує інструменти та середовище для організації, зберігання, доступу, оновлення та управління даними в базі даних.

SQL (Structured Query Language) – це мова програмування, яка використовується для управління запитів до реляційних баз даних.

NoSQL (Not Only SQL) – це підхід до баз даних, який відходить від традиційної реляційної моделі і пропонує більш гнучкі способи зберігання і управління даними. NoSQL-бази даних зазвичай не використовують SQL для запитів і можуть мати різні структури даних.

RESTful API (Representational State Transfer Application Programming Interface) – це архітектурний стиль розробки веб-сервісів, який використовує HTTP-протокол для обміну даними між клієнтом і сервером.

Міддлвера (middleware) – це програмний компонент, що розташовується між клієнтською і серверною частинами програмного забезпечення, для полегшення взаємодії між ними.

ВСТУП

У сучасному світі з розвитком інформаційних технологій все більше уваги приділяється підвищенню ефективності управління різними процесами. В галузі громадського харчування це зумовлює потребу в автоматизації процесів резервування місць у закладах.

Переважна більшість впровадженої на сьогодні в закладах харчування системи резервування базувалася на принципі бронювання через контакт-центри або безпосереднім зверненням користувачів до персоналу закладу. З розвитком інформаційних технологій перед споживачем відкрилася можливість зберігати дані в режимі дистанційного доступу.

Сучасні розробки дозволяють комерційним структурам, у тому числі закладам харчування, забезпечити дистанційні форми обслуговування користувачів через інформаційні системи керування. Інформаційна система дає змогу здійснити автоматизацію окремих комерційних операцій.

Реалізація онлайн-платформи для бронювання місць у закладах харчування надає можливість ефективніше управляти процесами резервування у даному сегменті комерційної діяльності.

Даний дипломний проект присвячено розробленню платформи для онлайн-резервуванню місць у закладах харчування.

1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Огляд проблеми, яку вирішує платформа

У сучасному світі, де зручність та ефективність є ключовими чинниками для багатьох людей, індустрія громадського харчування стикається з певними проблемами при організації бронювання місць у закладах харчування. Багато відвідувачів ресторанів, кафе та барів зіштовхуються з труднощами, такими як відсутність зручного онлайн-інтерфейсу для перегляду вільних місць, брак актуальної інформації про заклади, складнощі зі зв'язком. Ці проблеми можуть негативно вплинути на задоволеність клієнтів та ефективність роботи закладів.

З іншого боку, самі заклади харчування також стикаються з низкою викликів через відсутність єдиної платформи для управління резерваннями та просування своїх послуг. Це включає труднощі з відстеженням резервувань, обмежену присутність в онлайн-середовищі та втрачені можливості для маркетингу та залучення нових клієнтів.

Створення онлайн-платформи покликано вирішити ці проблеми, забезпечуючи зручний та ефективний процес бронювання місць у закладах харчування. Платформа поєднує в собі зручний інтерфейс для користувачів, канали зв'язку між закладами та відвідувачами. Це дозволяє оптимізувати процес вибору та бронювання місць, підвищити зручність для всіх учасників та стимулювати розвиток індустрії громадського харчування.

Для успішної реалізації цього проекту необхідно виконати наступні кроки:

1. Виконати аналіз поточних рішень для виявлення недоліків та можливостей для покращення.
2. Дослідити конкурентні платформи для визначення аспектів, які можна покращити за допомогою нового рішення.

3. Скласти перелік функціональних можливостей, які забезпечуватиме розроблювана платформа для задоволення потреб користувачів.
4. Вибрати технологічний стек для розробки програмного забезпечення, обґрунтувавши вибір та пояснити його потенційне використання.
5. Реалізувати програмну розробку онлайн-платформи для резервування місць у закладах харчування.
6. Провести тестування системи для забезпечення її правильної та стабільної роботи.

1.2. Аналіз існуючих програмних рішень

Для успішного створення онлайн-платформи необхідно ретельно проаналізувати проблеми, визначити можливості для вдосконалення та провести ґрунтовне дослідження наявних на ринку програмних рішень. Це дозволить краще зрозуміти предметну область, виявити прогалини та переваги існуючих платформ, що стане підґрунтям для розробки нового, більш досконалого продукту.

Буде розглянуто кілька популярних онлайн-платформ для бронювання місць у закладах харчування, які діють на українському ринку, а саме: Reservble.com [1], Restoran.ua [2], Reston.ua [3]. Аналіз їх функціональності, переваг та недоліків допоможе визначити аспекти, які потребують покращення або модернізації для задоволення потреб користувачів.

1.2.1. Reston.ua

Reston.ua – один з найпопулярніших українських онлайн-сервісів для бронювання столиків у закладах харчування. Платформа існує з 2012 року і на даний момент налічує більше 1 000 партнерських закладів у різних регіонах України.

Основною перевагою Reston.ua є простий та інтуїтивний пошук закладів за низкою критеріїв, таких як тип закладу, вибір кухні, місцезнаходження тощо, проте немає фільтрації за таким важливим показником як ціновий діапазон. Користувачі також можуть ознайомитися з детальною інформацією про заклади, включаючи фотографії інтер'єру та екстер'єру, відгуки відвідувачів та контактні дані.

Бронювання столиків на Reston.ua відбувається в режимі онлайн прямо на сайті або через мобільний додаток. Процес є повністю автоматизованим та зручним для користувачів. Крім того, платформа регулярно пропонує різноманітні акції, знижки та спеціальні пропозиції від своїх партнерських закладів. Платформа надає рестораторам зручну панель управління, де вони можуть керувати своїм профілем, оновлювати інформацію про заклад, додавати акції та спеціальні пропозиції.

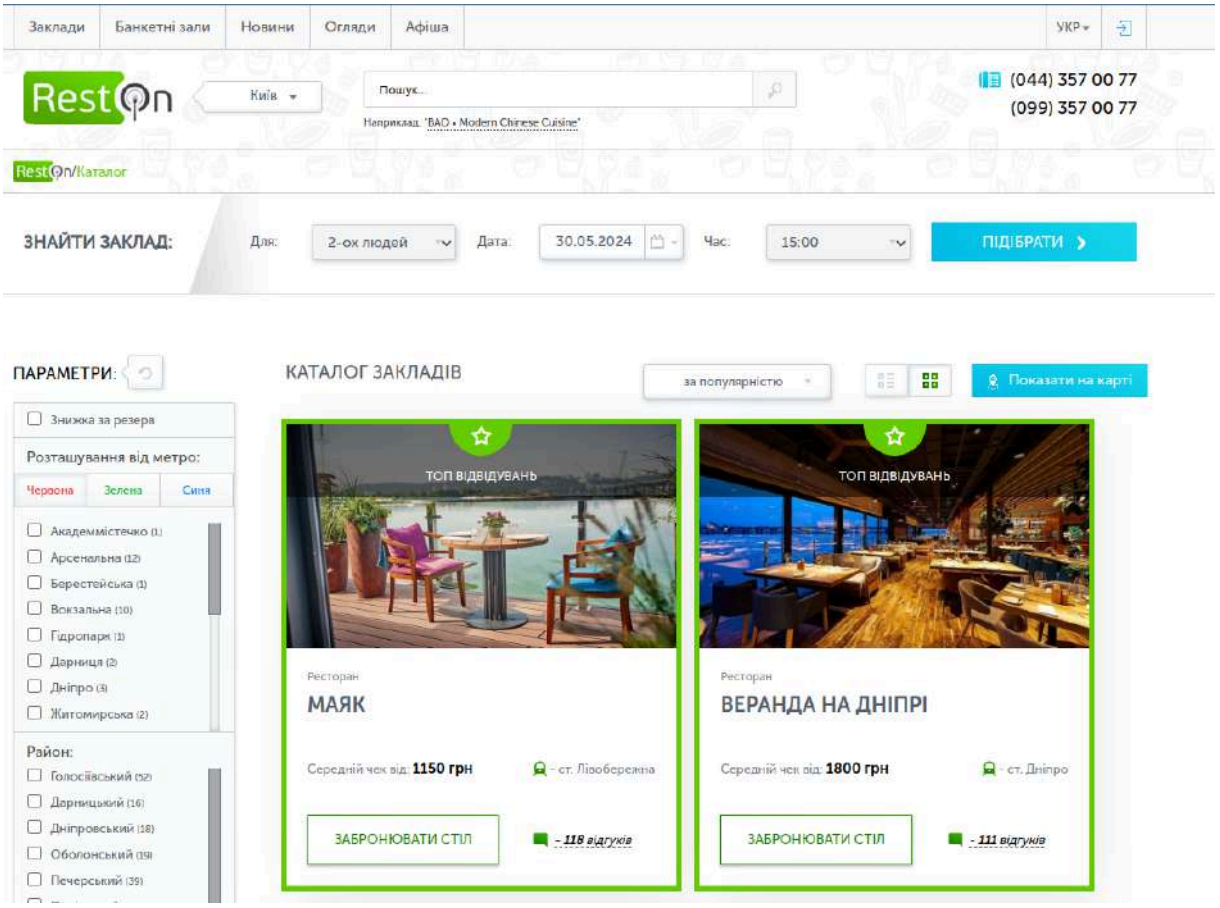


Рис. 1.1. Головна сторінка пошуку закладів з фільтрацією

Перелік основних особливостей Reston.ua:

1. Докладна інформація про заклади: меню, фотографії, рейтинги, відгуки відвідувачів.
2. Можливість забронювати місця онлайн без дзвінків.
3. Перегляд актуальних новин від закладу, які можуть включати в себе акції, знижки та подій.
4. Перегляд відфільтрованих закладів на мапі.

Отже, Reston.ua є доволі популярною та затребуваною в Україні, вона поки що обмежена лише українським ринком і тільки у великих містах та не представлена в інших країнах, у той час як мета даної платформи – доступність на усій території України та вихід на міжнародний ринок.

1.2.2. Reservble.com

Reservble.com – це українська онлайн-платформа для бронювання столиків у ресторанах, кафе, барах тощо. Проте так як і попередня платформа вона обмежена вибором закладів лише у великих містах України.

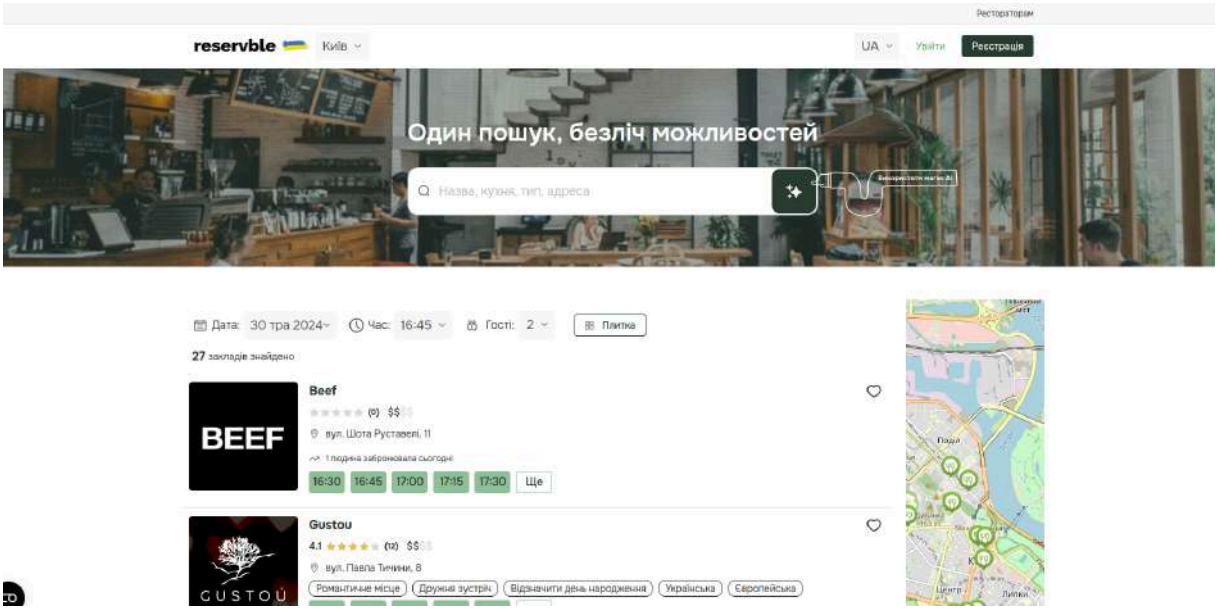


Рис. 1.2. Головна сторінка пошуку закладів з фільтрацією

Незважаючи на свій локальний характер, Reservble.com є зручним інструментом для бронювання столиків в Україні, але її географічна обмеженість може бути перешкодою для багатьох потенційних користувачів з інших регіонів країни. Користувачі можуть легко знайти потрібний заклад за допомогою фільтрів пошуку, таких як дата та час бронювання, кількість гостей та місцезнаходження. Процес бронювання столика здійснюється в кілька кліків прямо на сайті. Крім того, платформа пропонує систему відгуків та рейтингів закладів, а також регулярно оновлює спеціальні пропозиції та акції від партнерських закладів. Як і Reston.ua дана платформа має панель управління для рестораторів для керування бронюваннями, де вони можуть переглядати, редагувати та скасовувати бронювання.

Перелік основних особливостей Reservble.com:

1. Докладна інформація про заклади: меню, фотографії, рейтинги, відгуки відвідувачів.
2. Можливість забронювати місця онлайн без дзвінків.
3. Перегляд відфільтрованих закладів на мапі.
4. Пошук закладів поруч з метром або у певному районі міста.

Отже, Reservble.com як і Reston.ua має чималу популярність в Україні, проте вона так само обмежена лише українським ринком і тільки у великих містах.

1.2.3. Restoran.ua

Restoran.ua – це один з найстаріших та найбільших українських онлайн-сервісів для бронювання столиків у закладах харчування. Запущений у 2003 році, він швидко завоював довіру користувачів та закладів по всій Україні.

Основною перевагою Restoran.ua є зручний пошук закладів за різноманітними критеріями, такими як тип кухні, місцезнаходження, наявність паркінгу та інші параметри. Користувачі можуть ознайомитися з

детальною інформацією про заклади, включно з фотографіями інтер'єру, переглядом меню та відгуками попередніх відвідувачів.

Проте на відміну від попередніх платформ, Restoran.ua не надає можливості бронювання столиків, але також платформа регулярно пропонує різноманітні акції, знижки та спеціальні пропозиції від партнерських закладів.

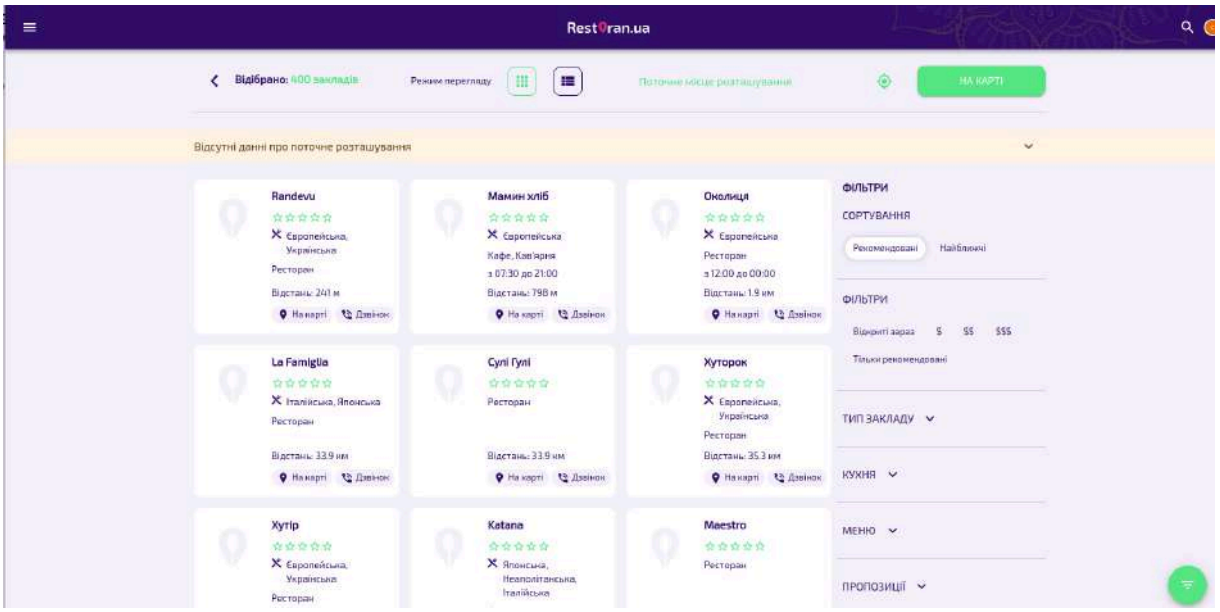


Рис. 1.3. Головна сторінка пошуку закладів з фільтрацією

Перелік основних особливостей Restoran.ua:

1. Докладна інформація про заклади: меню, фотографії, рейтинги, відгуки відвідувачів.
2. Перегляд закладів поруч.

Отже, Restoran.ua має чималу популярність в Україні, проте на відміну від попередніх платформ, даний сайт відрізняється відсутністю бронювання столиків онлайн, що є чималим мінусом при пошуку закладів.

1.3. Результати проведеного аналізу

Після детального вивчення онлайн-платформ Reston.ua, Reservble.com та Restoran.ua для бронювання місць у закладах харчування

в Україні сформуємо критерії порівняння платформ, для визначення сильних та слабких сторін:

1. Географічне охоплення – регіони, міста, в яких діє платформа. Чим ширша географія, тим більше вибір для користувачів.
2. Зручність пошуку та фільтрування – наявність різноманітних фільтрів (кухня, ціновий діапазон, місцезнаходження, тощо) для швидкого та точного пошуку бажаних закладів.
3. Можливості онлайн-бронювання – чи можна забронювати стіл безпосередньо на платформі онлайн або необхідно звертатися до закладу іншим шляхом.
4. Система відгуків та рейтингів – можливість користувачів залишати відгуки та рейтинги закладів для кращого вибору.
5. Картографічна візуалізація – можливість переглядати місцезнаходження закладів на інтерактивній карті.
6. Багатомовний інтерфейс – підтримка декількох мов для зручності іноземних користувачів.
7. Пошук закладів поблизу – можливість знаходити заклади поблизу поточного місцезнаходження користувача є зручною функцією, що допомагає швидко виявити найближчі варіанти
8. Інтерактивна комунікація з закладами – можливість прямого спілкування та інтерактивної взаємодії з представниками закладу через платформу є корисною функцією для уточнення деталей бронювання, умов обслуговування тощо.

Результат порівняння для вищеописаних програмних рішень за даними критеріями наведено у табл. 1.1.

Таблиця 1.1

Порівняльна характеристика існуючих програмних рішень

Назва / Критерії	Розроблене ПЗ	Reservble.com	Reston.ua	Restoran.ua

Продовження таблиці 1.1

Назва / Критерії	Розроблене ПЗ	Reservble.com	Reston.ua	Restoran.ua
Географічне охоплення	Україна	Певні міста України	Певні міста України	Певні міста України
Зручність пошуку та фільтрування	Так	Так	Ні	Так
Можливості онлайн-бронювання	Так	Так	Так	Ні
Система відгуків та рейтингів	Так	Так	Так	Так
Картографічна візуалізація	Так	Так	Так	Так
Багатомовний інтерфейс	Так	Так	Так	Так
Пошук закладів поруч	Так	Ні	Ні	Так
Інтерактивна комунікація з закладами	Так	Ні	Ні	Ні

Отже, базуючись на проведеному порівняльному аналізі існуючих онлайн-платформ для бронювання місць у закладах харчування за визначеними критеріями, можна зробити такі висновки, що жодна з розглянутих платформ (Reservble.com, Reston.ua, Restoran.ua) не є ідеальною та не задовольняє всі виділені критерії повною мірою. Проте, згідно з наведеними даними, передбачуваний продукт має потенціал стати більш функціональним та зручним для користувачів.

Ключові переваги майбутнього продукту:

1. Широке географічне охоплення по всій території України, на відміну від конкурентів, що обмежені певними містами.

2. Поєднання всіх основних функцій – зручний пошук та фільтрування, онлайн-бронювання, система відгуків та рейтингів, картографічна візуалізація, багатомовна підтримка.
3. Наявність додаткових корисних можливостей, відсутніх у конкурентів – пошук найближчих закладів поблизу користувача та інтерактивна комунікація з представниками закладів прямо на платформі.

Такий комплексний підхід та урахування найкращих практик і недоліків існуючих рішень дозволить створити більш досконалий, зручний та орієнтований на потреби користувачів продукт для бронювання місць у закладах харчування.

Забезпечення широкого вибору закладів по всій країні, різноманітних функцій пошуку, бронювання та комунікації в одній платформі створить значні переваги для залучення та утримання користувачів порівняно з обмеженими можливостями конкурентних рішень.

Таким чином, розробка нового програмного продукту з урахуванням проведеного аналізу дозволяє максимально задовольнити потреби користувачів та зайняти провідні позиції на ринку бронювання місць у закладах харчування в Україні.

1.4. Висновки до розділу 1

Під час виконання вимог розділу 1 та аналізу матеріалів цього розділу дипломного проекту було виконано наступні задачі:

1. Проаналізовано існуючі онлайн-платформи (Reservble.com, Reston.ua, Restoran.ua), що дозволяють бронювати місця у закладах харчування.
2. Визначено переваги та недоліки наявних рішень та виявлено основні проблеми, з якими стикаються користувачі при бронюванні за допомогою цих платформ.

3. Проведено порівняння трьох платформ включно з поданим рішенням за низкою важливих критеріїв: географічне охоплення, зручність пошуку, можливості бронювання, системи відгуків, картографічна візуалізація, багатомовність, пошук поблизу та інтерактивна комунікація з закладами.
4. Створено порівняльну таблицю, де для кожної платформи відображено наявність або відсутність певних функцій за визначеними критеріями аналізу.
5. Сформовано список основних вимог та можливостей, яким має відповідати розроблювана платформа в рамках дипломного проекту для бронювання місць у закладах харчування з метою усунення виявлених недоліків існуючих рішень.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РОЗРОБЛЕННЯ

2.1. Вибір СКБД

Вибір правильної СКБД є критично важливим рішенням під час розробки будь-якого програмного забезпечення, оскільки воно визначає, як дані будуть зберігатися, керуватися та використовуватися в системі. Існують різні типи СКБД, кожна з яких має свої переваги та недоліки, тому важливо ретельно проаналізувати вимоги проекту та обрати найбільш відповідне рішення.

Реляційні бази даних SQL [4], такі як PostgreSQL [5] та MySQL [6], є традиційним вибором для багатьох додатків. Вони структурують дані у вигляді таблиць із фіксованими схемами, що забезпечує строгу цілісність даних та підтримку складних транзакцій. Однак, для деяких сценаріїв використання, таких як робота з неструктурованими або швидко змінними даними, РБД можуть бути менш оптимальними.

З іншого боку, NoSQL [7] бази даних, такі як MongoDB [8], пропонують гнучкіші схеми даних та часто кращу масштабованість і продуктивність для певних типів додатків. Вони можуть бути особливо корисними для додатків, які потребують обробки великих обсягів неструктурованих даних або мають складні вимоги до масштабування.

2.1.1. MongoDB

MongoDB – це документо-орієнтована, NoSQL система керування базами даних, яка базується на сховищі даних у вигляді BSON. Деякі ключові особливості MongoDB включають гнучкість схеми, підтримку великих обсягів даних та швидкодію.

Переваги:

1. Гнучкість схеми: MongoDB не потребує фіксованих схем, що дозволяє легко змінювати структуру даних без необхідності зміни таблиць чи колонок.

2. Підтримка великих обсягів даних: MongoDB ефективно працює з великими обсягами даних і може масштабуватись горизонтально за допомогою кластеризації, що робить його ідеальним для високонавантажених додатків.
3. Швидкодія: MongoDB використовує індексацію та кешування для швидкого доступу до даних, що забезпечує високу продуктивність при виконанні запитів.

Недоліки:

1. Відсутність підтримки транзакцій: MongoDB не підтримує складні транзакції, що може бути обмеженням для деяких додатків, які потребують гарантованої цілісності даних.
2. Менша зрілість: MongoDB, будучи NoSQL базою даних, може бути менш зрілим та стабільним порівняно з деякими реляційними СКБД.
3. Великий обсяг пам'яті: MongoDB може вимагати більше пам'яті для зберігання даних через використання BSON.

2.1.2. PostgreSQL

PostgreSQL – це реляційна СКБД з відкритим вихідним кодом, яка підтримує складні SQL запити та транзакції. Основні характеристики PostgreSQL включають розширену підтримку SQL, транзакційну безпеку та розширюваність.

Переваги:

1. Розширена підтримка SQL: PostgreSQL має багато функцій та можливостей SQL, таких як підзапити, внутрішні з'єднання та інші.
2. Транзакційна безпека: PostgreSQL гарантує транзакційну безпеку для забезпечення цілісності даних під час одночасних операцій.
3. Розширюваність: PostgreSQL підтримує різні розширення, такі як PostGIS для географічних даних.

Недоліки:

1. Складність налаштування: PostgreSQL може вимагати більше часу для налаштування та оптимізації порівняно з іншими СКБД.
2. Вимоги до ресурсів: PostgreSQL може вимагати більше ресурсів для ефективної роботи, особливо при виконанні складних запитів та операцій.

2.1.3. MySQL

MySQL – це популярна реляційна СКБД, яка широко використовується у веб-розробці та інших галузях. Деякі особливості MySQL включають простоту використання, швидкодію та широку підтримку.

Переваги:

1. Простота використання: MySQL має простий синтаксис та інтуїтивно зрозумілу структуру таблиць.
2. Швидкодія: MySQL може працювати ефективно з невеликими до середніх обсягами даних та швидко виконувати прості запити.
3. Широка підтримка: MySQL підтримується багатьма хостинг-провайдерами і має велику спільноту користувачів, що забезпечує доступ до багатьох ресурсів та підтримки.

Недоліки:

1. Обмежена розширюваність: MySQL має менше розширень та можливостей порівняно з PostgreSQL.
2. Обмежена підтримка складних запитів: MySQL може не підтримувати деякі складні SQL запити та функції, доступні в інших реляційних СКБД.

2.1.4. Результати аналізу

Проаналізувавши наявні рішення, а також враховуючи особливості проєкту онлайн-платформи для бронювання місць у закладах харчування,

MongoDB є найбільш оптимальним вибором серед розглянутих СКБД з наступних причин:

1. Гнучкість схеми даних: MongoDB дозволяє легко змінювати структуру даних без необхідності зміни основної схеми бази даних. Це особливо корисно для платформи, де інформація про заклади, бронювання та користувачів може часто змінюватись.
2. Швидкодія та масштабованість: MongoDB забезпечує швидкий доступ до даних та можливість горизонтального масштабування за допомогою кластеризації, що дозволить забезпечити стабільну роботу платформи при зростанні навантаження.
3. Підтримка операцій реального часу: MongoDB підтримує операції реального часу, такі як оновлення даних про доступність місць та нові бронювання, що є критичним для онлайн-платформи бронювання.
4. Документо-орієнтований підхід: MongoDB використовує BSON для зберігання даних, що спрощує інтеграцію з сучасними веб-технологіями та полегшує розробку.

Отже, завдяки своїй гнучкості, продуктивності, масштабованості та орієнтації на роботу з неструктурованими даними, MongoDB є оптимальним вибором СКБД для розробки онлайн-платформи бронювання місць у закладах харчування.

2.2. Вибір технологій для серверної частини

Для створення ефективної та масштабованої серверної частини для онлайн-платформи бронювання місць у закладах харчування необхідно ретельно обрати стек технологій. Існує кілька популярних варіантів, таких як Node.js [9] з Express [10] та TypeScript [11], Java [12] з Spring Boot [13] та Python [14] з Django [15]. Кожен з них має свої переваги та недоліки, тому розглянемо кожен з них окремо для визначення найкращого рішення.

2.2.1. Node.js (Express з TypeScript)

Node.js – це середовище виконання JavaScript з відкритим кодом, побудоване на движку V8 [16] від Google, яке дозволяє виконувати JavaScript поза браузером.

Express – це мінімалістичний та гнучкий фреймворк для Node.js, який допомагає створювати веб-додатки та API. Він забезпечує зручні засоби для керування маршрутизацією, обробки запитів та відповідей, а також підтримку міддлверів.

TypeScript – це розширення JavaScript, розроблене Microsoft, що додає статичну типізацію. Він допомагає виявляти помилки на етапі компіляції, покращує читабельність коду та полегшує рефакторинг.

Переваги:

1. Висока продуктивність та масштабованість: Node.js використовує події-керовану, неблокуючу модель введення/виведення, що забезпечує високу пропускну здатність та ефективність.
2. Одна мова для клієнтської та серверної частини: використання JavaScript як на клієнті (для розробки клієнтської частини використовують JavaScript), так і на сервері, спрощує розробку та підтримку.
3. Масштабованість: Node.js легко масштабується як вертикально (збільшення потужності сервера), так і горизонтально (додавання нових вузлів).
4. Велика екосистема та активне співтовариство: Node.js має багато готових модулів та бібліотек, а також велике та активне співтовариство розробників.
5. Швидка розробка: завдяки простоті та гнучкості Node.js, Express та TypeScript, розробка може відбуватися швидше.

Недоліки:

1. Однопоточна модель: може викликати проблеми при виконанні

CPU-інтенсивних завдань, оскільки Node.js працює в однопоточному режимі.

2. Відносно новий статус: Node.js є відносно новою технологією, що може призводити до нестабільності в деяких випадках.

2.2.2. Java (Spring Boot)

Java – це потужна, об’єктно-орієнтована мова програмування, яка широко використовується для створення різноманітних додатків, включаючи веб-застосунки.

Spring Boot – це популярний фреймворк для розробки веб-додатків та мікросервісів на Java. Він забезпечує вбудований контейнер сервлетів (наприклад, Tomcat), автоматичну конфігурацію та безліч утиліт для спрощення розробки.

Переваги:

1. Надійність та стабільність: Java відома своєю надійністю та стабільністю, що робить її хорошим вибором для великих та складних проектів.
2. Багата екосистема: Spring Boot надає багато вбудованих можливостей та має велику кількість сторонніх бібліотек та інструментів.
3. Підтримка великих корпорацій: Java широко використовується великими корпораціями та має потужну підтримку.

Недоліки:

1. Складність налаштування та розгортання: Java та Spring Boot можуть бути складними у налаштуванні та розгортанні, особливо для новачків.
2. Більша потреба в ресурсах: Java-додатки зазвичай потребують більше ресурсів порівняно з іншими рішеннями.

2.2.3. Python (Django)

Python – це інтерпретована мова програмування високого рівня з простим та зрозумілим синтаксисом. Вона широко використовується у веб-розробці, науці про дані, автоматизації та багатьох інших сферах.

Django – це потужний веб-фреймворк для Python, який дотримується парадигми "батарея включена". Він пропонує вбудовані засоби для створення веб-додатків, включаючи маршрутизацію, обробку запитів, адміністративну панель, систему аутентифікації тощо.

Переваги:

1. Простота у вивченні та використанні: Python вважається однією з найбільш зрозумілих мов програмування, що полегшує розробку та супровід проекту.
2. Гнучкість: Django надає широкі можливості для створення веб-додатків, включаючи підтримку різних баз даних та можливість швидко розробляти прототипи.
3. Велике спільнота: Python має велику та активну спільноту розробників, що забезпечує багато ресурсів та підтримки.

Недоліки:

1. Відносно нижча продуктивність: порівняно з іншими технологіями, такими як Node.js або Java, продуктивність Python може бути нижчою.
2. Обмежені можливості масштабування: Django може мати обмеження в масштабуванні для дуже великих проектів.

2.2.4. Результати аналізу

Після ретельного аналізу Node.js з Express та TypeScript виглядає найкращим вибором для розробки серверної частини онлайн-платформи для бронювання місць у закладах харчування. Цей стек забезпечує високу продуктивність, масштабованість, швидку розробку та сумісність з

клієнтською частиною на React. Крім того, він має величезну екосистему готових модулів та пакетів, а також активну спільноту розробників.

Хоча Java (Spring Boot) та Python (Django) також є потужними варіантами з власними перевагами, Node.js краще відповідає вимогам даного проекту. Він дозволить ефективно створити веб-платформу, яка зможе обробляти великі навантаження та масштабуватися в міру зростання кількості користувачів та функціональності.

2.3. Вибір технологій для клієнтської частини

Вибір правильного стеку технологій для клієнтської частини веб-додатку є критично важливим рішенням, яке впливає на ефективність розробки, продуктивність, підтримку та масштабованість проекту. На ринку існує кілька провідних рішень, таких як React [17], Angular [18] та Vue.js [19], кожне з яких має свої переваги та недоліки.

Головним завданням клієнтської частини онлайн-платформи для бронювання місць у закладах харчування є забезпечення зручного та інтуїтивно зрозумілого користувацького інтерфейсу, що дозволить користувачам легко знаходити заклади, переглядати наявні місця та здійснювати бронювання. Крім того, важливою вимогою є масштабованість та можливість підтримки додавання нових функцій у майбутньому.

2.3.1. React (Vite з TypeScript)

React, створений Facebook, є однією з найпопулярніших бібліотек JavaScript для створення користувацьких інтерфейсів. Він базується на компонентній архітектурі, що робить код модульним і полегшує повторне використання компонентів. Завдяки використанню віртуального DOM [20], React забезпечує високу продуктивність, особливо при частих оновленнях інтерфейсу. Крім того, React має величезну спільноту розробників та багату

екосистему пакетів і бібліотек, що спрощує розробку та інтеграцію додаткових функцій.

Для підвищення ефективності розробки з React можна використовувати Vite [21] – сучасний інструмент для розробки, який забезпечує швидке перезавантаження та оптимізований вихідний код завдяки використанню ES модулів [22]. TypeScript, у свою чергу, додає статичну типізацію до JavaScript, що допомагає виявляти помилки на етапі компіляції та покращує читабельність коду, що особливо корисно для великих проектів.

Переваги:

1. Ефективність: використання Vite забезпечує швидке завантаження та перезавантаження під час розробки, що дозволяє зосередитися на написанні коду без затримок.
2. Широка підтримка: React має велику та активну спільноту, яка забезпечує безліч готових рішень та компонентів, що прискорює процес розробки.
3. Гнучкість: React дозволяє будувати складні інтерфейси з використанням компонентної архітектури, що полегшує розробку та підтримку коду.
4. Статична типізація: TypeScript додає статичну типізацію до JavaScript, що покращує безпеку та надійність коду.

Недоліки:

1. Відсутність вбудованих рішень: React є бібліотекою, а не повноцінним фреймворком, тому для багатьох задач потрібне використання додаткових бібліотек, що може ускладнити налаштування проекту.

2.3.2. Angular

Angular, розроблений Google, є повнофункціональним фреймворком для створення веб-додатків. Він включає в себе багато вбудованих

можливостей, таких як маршрутизація, система модулів і залежностей, що полегшує розробку та підтримку додатків. Однак Angular має більшу криву навчання та жорсткішу структуру проекту порівняно з React.

Переваги:

1. Повнота: Angular має вбудований набір інструментів, таких як маршрутизація, форми та модулі, що полегшує розробку та підтримку додатків.
2. Строго типізований: Angular використовує TypeScript за замовчуванням, що забезпечує статичну типізацію та покращує безпеку коду.
3. Офіційна документація: Angular має велику та добре організовану офіційну документацію, яка допомагає розробникам швидко вирішувати проблеми.

Недоліки:

1. Складність: Angular є великим і комплексним фреймворком, що може збільшити час навчання та розробки.
2. Перевантаженість функціоналом: велика кількість вбудованих функцій може бути зайвою для менших проектів, що може негативно вплинути на продуктивність.

2.3.3. Vue

Vue.js – це прогресивний фреймворк для створення користувацьких інтерфейсів. Він легкий у вивченні та має гнучку структуру, що дозволяє використовувати його як для невеликих додатків, так і для великих SPA [23]. Vue.js пропонує швидку продуктивність завдяки використанню віртуального DOM та має активну спільноту розробників. Однак він може мати обмежену підтримку від великих компаній порівняно з React чи Angular.

Переваги:

1. Простота: Vue має простий та легкий у використанні синтаксис, що полегшує розробку та розуміння коду.
2. Гнучкість: Vue дозволяє розробляти додатки будь-якої складності, використовуючи компонентну архітектуру та вбудовані функції.
3. Прогресивність: Vue може починати як простий скрипт для веб-сторінок та поступово переходити до розробки складних SPA.

Недоліки:

1. Менша екосистема: хоча спільнота Vue активно зростає, вона все ще менша порівняно з React, що може обмежити доступ до готових рішень та компонентів.
2. Обмежена корпоративна підтримка: Vue менш поширений у великих корпораціях порівняно з Angular та React, що може вплинути на підтримку та розвиток проекту.

2.3.4. Результати аналізу

Беручи до уваги вимоги проекту, потужність React, переваги Vite та TypeScript, а також величезну спільноту та екосистему, React з Vite та TypeScript можна вважати найкращим вибором для створення клієнтської частини онлайн-платформи для бронювання місць у закладах харчування. Це рішення забезпечить високу продуктивність, швидку розробку, масштабованість та надійність, необхідні для успішної реалізації даного проекту.

2.4. Вибір додаткових технологій

Для забезпечення оптимальної роботи та функціональності онлайн-платформи для бронювання місць у закладах харчування необхідно ретельно підібрати стек технологій, який найкращим чином відповідатиме поставленим вимогам. Окрім основних технологій, фреймворків та бібліотек, доцільно розглянути додаткові інструменти та рішення, які

дозволять ефективно інтегрувати різні компоненти системи, забезпечити масштабованість, високу продуктивність та безперебійну роботу додатку.

Для об'єднання всіх обраних технологій в єдине ціле та забезпечення їх злагодженої взаємодії найкращим рішенням буде використання системи Docker [24] – потужної платформи контейнеризації. Вона дозволить упакувати додаток разом з усіма необхідними залежностями в ізольоване середовище, гарантуючи портативність, відтворюваність, ефективне використання ресурсів та підвищену безпеку.

Для реалізації інтерактивної взаємодії між користувачами та менеджерами закладів в режимі реального часу буде задіяно технологію веб-сокетів Socket.IO [25]. Ця бібліотека забезпечить надійний двосторонній зв'язок, автоматичне відновлення з'єднання та можливість масштабування, створюючи зручний інтерфейс для спілкування та оновлення інформації в реальному часі.

Для керування навантаженням, розподілу трафіку та забезпечення надійності рекомендовано використати Nginx [26] – потужний веб-сервер та проксі-сервер зі зворотним проксі.

Також для ефективного зберігання, обробки та доставки медіаконтенту (зображень) буде використано хмарну платформу Cloudinary. Вона забезпечить автоматичну оптимізацію медіафайлів, трансформації зображень, масштабованість та високу доступність зберігання та доставки контенту через розподілену CDN-мережу.

2.4.1. Docker

Docker – це платформа для створення, розгортання та запуску додатків у контейнерах. Контейнери забезпечують ізольоване середовище для додатків, що спрощує процес розгортання та масштабування.

Використання Docker для даного проєкту принесе наступні переваги:

1. Портативність: контейнери Docker можуть працювати на будь-якій платформі, що підтримує Docker (Linux, Windows, macOS), забезпечуючи портативність та уникаючи конфліктів залежностей.
2. Легке розгортання: Docker спрощує процес розгортання додатків, оскільки всі залежності та конфігурації вже включені в контейнер. Це забезпечує однакове середовище для розробки та тестування платформи.
3. Масштабованість: Docker дозволяє легко масштабувати додатки шляхом запуску додаткових контейнерів. Це особливо корисно для платформи, яка з часом може зростати в обсягах трафіку та навантаження.
4. Ефективність ресурсів: контейнери Docker використовують менше ресурсів, ніж повноцінні віртуальні машини, що робить їх більш ефективними та економічними.
5. Модульність: Docker сприяє модульному підходу до розробки додатків, де кожен компонент може бути упакований у окремий контейнер, що полегшує розробку, тестування та обслуговування.

2.4.2. Socket.IO

Socket.IO – це JavaScript бібліотека для реалізації двостороннього зв'язку в режимі реального часу між клієнтом та сервером. Для даної платформи Socket.IO може бути використаний для наступних цілей:

1. Оновлення в реальному часі: Socket.IO дозволить оновлювати інформацію про нові бронювання та інші дані в режимі реального часу для користувачів та менеджерів.
2. Спілкування в чатах: на платформі передбачені чати для спілкування між користувачами та менеджерами закладів. Socket.IO ідеально підходить для реалізації функціональності чату в реальному часі.

3. Сповіщення про події: використовуючи Socket.IO, можна надсилати сповіщення про нові бронювання, скасування бронювань або інші важливі події в режимі реального часу.
4. Двосторонній зв'язок: Socket.IO забезпечує двосторонній зв'язок між клієнтом та сервером, що дозволяє як клієнтам, так і серверу ініціювати та отримувати повідомлення, що є важливим для сучасних веб-додатків.

2.4.3. Nginx

Nginx – це високопродуктивний веб-сервер та проксі-сервер, який може бути використаний для керування навантаженням, розподілу трафіку та забезпечення надійності онлайн-платформи для бронювання місць у закладах харчування. Використання Nginx принесе наступні переваги:

1. Балансування навантаження: Nginx може виступати в ролі зворотного проксі-серверу, розподіляючи трафік між кількома серверами або контейнерами додатку. Це забезпечить рівномірне розподілення навантаження та підвищить продуктивність платформи.
2. Кешування: Nginx має вбудовані можливості для кешування статичного контенту, такого як зображення, CSS та JavaScript файли. Це зменшить навантаження на сервери додатку та прискорить завантаження сторінок для користувачів.
3. Безпека: Nginx надає різноманітні функції безпеки, такі як захист від DDoS-атак, фільтрування трафіку та обмеження швидкості передачі даних. Це допоможе захистити платформу від шкідливої активності.
4. Гнучка конфігурація: Nginx має зрозумілий та потужний конфігураційний файл, що дозволяє легко налаштовувати поведінку сервера відповідно до таких вимог як, перенаправлення трафіку, встановлення заголовків та інше.
5. Моніторинг стану: Nginx може відстежувати стан підключених серверів додатку та автоматично видаляти недоступні сервери з

балансування навантаження, забезпечуючи безперебійну роботу платформи.

2.4.4. Cloudinary

Cloudinary – це хмарна платформа для управління та доставки медіаконтенту. Її використання в проекті забезпечить ефективне зберігання, обробку та доставку зображень, відео та інших медіафайлів. Основні переваги Cloudinary:

1. Оптимізація медіаконтенту: Cloudinary автоматично оптимізує зображення для кращої продуктивності завантаження та відображення на різних пристроях та з'єднаннях.
2. Масштабованість та висока доступність: Cloudinary має розподілену архітектуру з кешуванням контенту на крайових серверах, забезпечуючи високу доступність та масштабованість.
3. Трансформації зображень: Платформа надає широкий спектр інструментів для обробки зображень, таких як зміна розмірів, обрізання, додавання фільтрів тощо.
4. Зберігання та доставка: Контент зберігається в хмарному сховищі Cloudinary та доставляється через їхню розподілену мережу доставки контенту (CDN).
5. Інтеграція та API: Cloudinary інтегрується з популярними фреймворками та CMS, а також надає потужні API для взаємодії з платформою.

2.4.5. Результати аналізу

Для забезпечення оптимальної роботи та функціональності онлайн-платформи для бронювання місць у закладах харчування рекомендовано використати додаткові технології Docker, Socket.IO та Nginx. Docker забезпечить портативність, легке розгортання, масштабованість та ефективне використання ресурсів через

контейнеризацію додатку. Socket.IO дозволить реалізувати взаємодію в режимі реального часу, включаючи оновлення даних, чати та сповіщення для користувачів і менеджерів закладів. Nginx виступатиме в ролі зворотного проксі-серверу, забезпечуючи балансування навантаження, кешування, безпеку та моніторинг стану серверів додатку. Застосування цих технологій створить надійну основу для високопродуктивної та безперебійної роботи платформи.

2.5. Висновки до розділу 2

Після ретельного аналізу та оцінки різних варіантів технологій для клієнтської, серверної частин та СКБД, найкращим комплексним рішенням для розробки онлайн-платформи бронювання місць у закладах харчування є використання React (з Vite та TypeScript) для клієнтської частини, Node.js (Express з TypeScript) для серверної частини та MongoDB як системи керування базами даних.

Для клієнтської частини React, поєднаний з Vite та TypeScript, забезпечить високу продуктивність, швидку розробку та надійність користувацького інтерфейсу. React пропонує компонентну архітектуру та великі спільноту і екосистему, що полегшить розробку та підтримку складного інтерфейсу платформи. Використання Vite прискорить процес розробки, а TypeScript додасть статичну типізацію для кращої підтримки великих проектів.

На стороні серверної частини Node.js з Express та TypeScript є ідеальним вибором завдяки своїй високій продуктивності, масштабованості та інтеграції з клієнтською частиною на JavaScript. Сполучення Express та TypeScript забезпечить зручність розробки API та обробки маршрутів із перевагами типізації, що зробить код надійнішим та легшим для підтримки.

Для зберігання та керування даними MongoDB є найбільш підходящою системою керування базами даних. Будучи

документо-орієнтованою NoSQL базою даних, MongoDB надасть необхідну гнучкість схеми даних для зміни структури інформації про заклади, бронювання та користувачів без переробки основної схеми. Крім того, MongoDB забезпечить високу продуктивність, масштабованість та підтримку операцій реального часу, що є критичними вимогами для онлайн-платформи бронювання.

Обраний стек технологій React, Node.js та MongoDB створить потужне та ефективне рішення для розробки онлайн-платформи бронювання місць у закладах харчування. Він забезпечить високу продуктивність, масштабованість, швидку розробку та гнучкість, необхідні для успішної реалізації проекту, а також можливість легко додавати нові функції та розширення в майбутньому.

Таким чином, комплексний вибір React (Vite, TypeScript) для клієнтської частини, Node.js (Express, TypeScript) для серверної частини та MongoDB як СКБД є оптимальним рішенням для створення масштабованої, продуктивної та гнучкої онлайн-платформи бронювання місць у закладах харчування.

3. СТРУКТУРНА ОРГАНІЗАЦІЯ ПРОГРАМНИХ ЗАСОБІВ

3.1. Загальна структура онлайн-платформи

Онлайн-платформа для бронювання місць у закладах харчування є багатокомпонентним веб-додатком, який складається з серверної та клієнтської частин. Для забезпечення оптимальної продуктивності, масштабованості та зручності розробки було ретельно підібрано стек сучасних технологій та фреймворків, що дозволяють створити надійну та ефективну систему.

Серверна частина платформи реалізована з використанням Node.js та фреймворку Express.js. Node.js забезпечує асинхронну обробку запитів, що дозволяє ефективно використовувати ресурси сервера та обробляти велику кількість з'єднань одночасно. Express.js надає зручний інструментарій для створення RESTful API, маршрутизації запитів та управління міддлверами.

Для зберігання даних використовується база даних MongoDB, яка є неструктурованою та масштабованою. Взаємодія з MongoDB реалізована за допомогою об'єктно-документної бібліотеки маппінгу Mongoose, що спрощує роботу з базою даних та забезпечує зручний інтерфейс для визначення схем даних та моделей.

Клієнтська частина розроблена з використанням бібліотеки React та мови TypeScript. React забезпечує декларативний підхід до створення користувацького інтерфейсу, що полегшує розробку та підтримку коду, а TypeScript додає статичну типізацію до JavaScript, покращуючи читабельність та виявлення помилок на етапі розробки.

Для реалізації взаємодії в режимі реального часу, наприклад, для чатів та оновлення інформації про бронювання, використовується технологія Socket.IO. Вона забезпечує двосторонній зв'язок між клієнтами та сервером та дозволяє ефективно передавати дані в реальному часі.

Для контейнеризації та полегшення розгортання всіх компонентів системи використовується Docker. Він інкапсулює додаток та всі його

залежності в ізольованих контейнерах, забезпечуючи портативність, масштабованість та ефективне використання ресурсів.

Інтеграція Nginx у стек технологій платформи дозволить ефективно керувати трафіком, забезпечить масштабованість, продуктивність та надійність онлайн-платформи для бронювання місць у закладах харчування.

Загальна структура програмного забезпечення побудована на принципах модульності та розділення відповідальностей, що забезпечує гнучкість та зручність розробки, тестування та підтримки коду.

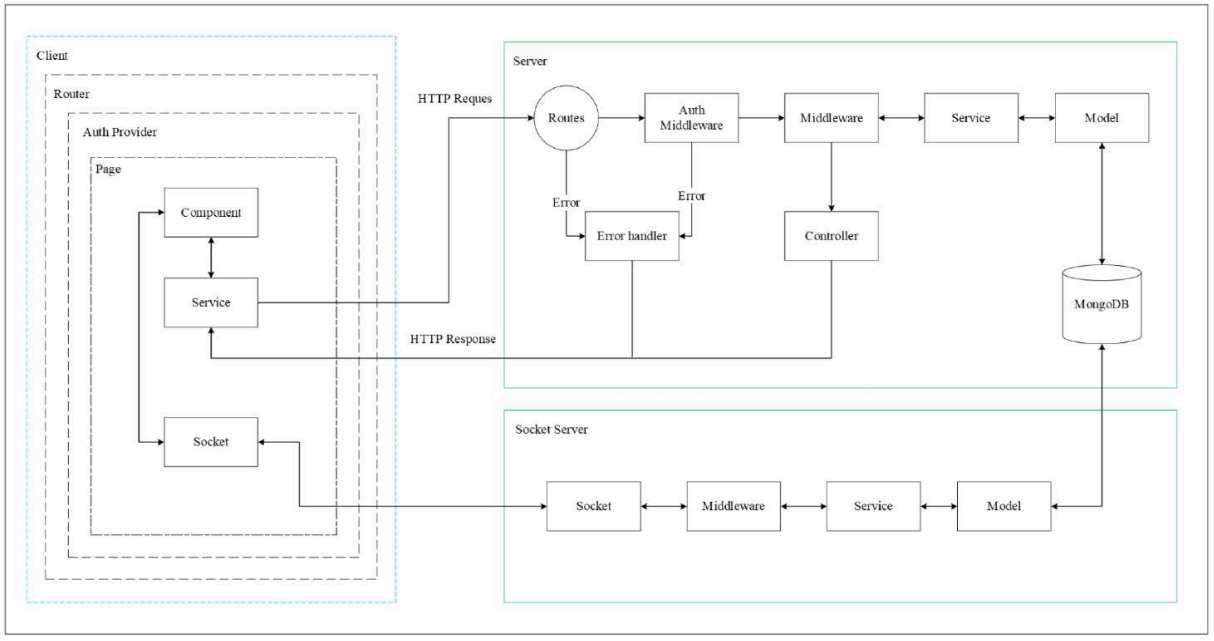


Рис. 3.1. Загальна структура онлайн-платформи

3.1.1. Загальна структура серверної частини

Серверна частина онлайн-платформи для бронювання місць у закладах харчування відіграє ключову роль у забезпеченні основної функціональності системи. Вона відповідає за обробку запитів від клієнтів, взаємодію з базою даних, реалізацію бізнес-логіки та забезпечення безпеки даних.

Основна функціональність серверної частини:

1. Аутентифікація: процес реєстрації та входу користувачів у систему

реалізований на стороні сервера. Це забезпечує безпеку даних та запобігає несанкціонованому доступу до платформи.

2. Розподіл ролей: система підтримує різні ролі користувачів, такі як звичайні користувачі, менеджери закладів та адміністратори платформи. Серверна частина відповідає за перевірку ролей та надання відповідних дозволів для виконання певних дій.
3. Хешування паролів: для захисту конфіденційних даних користувачів, таких як паролі, використовується односторонній алгоритм хешування. Це унеможливлює відновлення вихідного паролю навіть у разі витоку хешованих даних.
4. CRUD операції: серверна частина забезпечує виконання операцій створення (Create), читання (Read), оновлення (Update) та видалення (Delete) даних у базі даних. Ці операції стосуються різних сутностей, таких як користувачі, заклади, бронювання, новини тощо.

Структура серверної частини розподілена на модулі для забезпечення модульності, гнучкості та зручності розробки та підтримки коду.

Основними модулями є:

- Сервіси (Services): реалізують основну бізнес-логіку додатку, такі як обробка бронювань, управління закладами та користувачами.
- Контролери (Controllers): відповідають за обробку HTTP-запитів, валідацію вхідних даних та делегування запитів до відповідних сервісів.
- Міддлвари (Middlewares): виконують додаткову обробку запитів, такі як перевірка аутентифікації, парсинг тіла запиту тощо.
- Роутери (Routes): визначають маршрути для HTTP-запитів та пов'язують їх з відповідними контролерами.
- Схеми документів (Schema): визначають структуру даних, які зберігаються в базі даних MongoDB.

- Інтерфейси (Interfaces): описують контракти для різних компонентів системи, забезпечуючи типізацію та полегшуючи розробку.
- Валідатори (Validators): відповідають за перевірку та валідацію вхідних даних відповідно до заданих правил.
- Представники (Presenters): форматують та перетворюють дані з бази даних у зручний для клієнтів формат.
- Шаблони електронних листів (Email-templates): містять HTML-шаблони для генерації електронних листів, які надсилаються користувачам системи.
- Сокети (Socket): реалізують функціональність взаємодії в режимі реального часу за допомогою технології Socket.IO.
- Config (Конфігурація): зберігає конфігураційні налаштування додатку, такі як налаштування бази даних, секретні ключі, URL-адреси сервісів тощо.

3.1.2. Загальна структура клієнтської частини

Клієнтська частина онлайн-платформи для бронювання місць у закладах харчування відповідає за надання зручного та інтерактивного користувацького інтерфейсу, а також за обробку взаємодії користувача з додатком. Вона побудована з використанням бібліотеки React та TypeScript, що забезпечує модульність, гнучкість та типову безпеку коду.

Основна функціональність клієнтської частини:

1. Аутентифікація та авторизація: клієнтська частина відповідає за процеси реєстрації, входу та виходу користувачів з системи. Також реалізовано перевірку прав доступу користувачів до певних розділів, даних чи функцій додатку.
2. Розподіл ролей та дозволів: система підтримує різні ролі користувачів, такі як звичайні користувачі, менеджери закладів та адміністратори платформи. Клієнтська частина забезпечує

відображення відповідного інтерфейсу та функціональних можливостей залежно від ролі користувача.

3. Управління бронюваннями: користувачі можуть переглядати доступні заклади, вибирати дати та часи, а також створювати, редагувати чи скасовувати бронювання місць.
4. Шифрування та розшифрування повідомлень: для забезпечення приватності спілкування, повідомлення в чатах шифруються на стороні клієнта перед відправкою на сервер та розшифровуються при отриманні.
5. Робота в офлайн-режимі: деякі функціональності додатку, такі як перегляд закладів, новин або історії бронювань, можуть бути доступні в офлайн-режимі за допомогою кешування даних на стороні клієнта.
6. Локалізація та інтернаціоналізація: Клієнтська частина підтримує відображення інтерфейсу та даних на різних мовах, відповідно до налаштувань користувача.
7. Взаємодія в режимі реального часу: клієнтська частина забезпечує функціональність чатів для спілкування між користувачами та менеджерами закладів у режимі реального часу. Також реалізовано оновлення даних про бронювання, новини та інші зміни в реальному часі.
8. Адаптивний дизайн: інтерфейс користувача розроблений з урахуванням адаптивного дизайну, що забезпечує оптимальний досвід використання додатку на різних пристроях та розмірах екрану.

Структура клієнтської частини побудована відповідно до сучасних принципів організації веб-додатків та складається з наступних ключових модулів:

1. Компоненти (Components): це основні будівельні блоки інтерфейсу користувача, які відповідають за візуалізацію даних та обробку

подій користувача. Вони можуть бути атомарними (кнопки, поля введення тощо) або складними (сторінки, модальні вікна тощо).

2. Сторінки (Pages): представляють логічні розділи додатку та містять композицію компонентів для відображення відповідної функціональності.
3. Маршрути (Routes): визначають маршрути навігації між сторінками додатку та забезпечують передачу відповідних даних між ними.
4. Сервіси (Services): реалізують взаємодію з серверною частиною через HTTP-запити або веб-сокети. Вони відповідають за отримання, відправлення та обробку даних від сервера.
5. Контексти (Contexts): Забезпечують глобальне управління станом додатку та передачу даних між компонентами без необхідності пропсів.
6. Хуки (Hooks): реалізують повторно використовувану логіку, яка може бути застосована в різних компонентах, наприклад, для взаємодії з сервісами або управління станом.
7. Утиліти (Utilities): містять допоміжні функції та інструменти, такі як форматери, валідатори, шифрування/розшифрування даних тощо.
8. Стилі (Styles): визначають стилі та оформлення компонентів користувацького інтерфейсу.
9. Public: містить статичні файли, такі як зображення, шрифти, файли локалізації. Вона забезпечує зручне місце для зберігання всіх статичних ресурсів, необхідних для функціонування React-додатку. Під час збірки проекту, React статично копіює ці файли до кореневої директорії збірки, роблячи їх доступними для використання в додатку.

3.2. Структура бази даних

База даних відіграє ключову роль в онлайн-платформі для бронювання місць у закладах харчування, забезпечуючи зберігання та організацію всіх необхідних даних. Для зберігання даних використовується неструктурована та маштабована база даних MongoDB, яка дозволяє ефективно працювати з великими обсягами різноманітної інформації.

Структура бази даних ретельно спроектована з урахуванням вимог та специфіки системи. Вона складається з кількох взаємопов'язаних колекцій, кожна з яких відповідає певній сутності додатку. Зв'язки між колекціями відображені на ER-діаграмі (див. Додаток 1). Детальний опис полів кожної схеми, описаних нижче, допоможе краще зрозуміти структуру бази даних, типи даних, які зберігаються, та взаємозв'язки між різними колекціями.

Колекція users (користувачі) – профілі зареєстрованих користувачів:

- _id: унікальний ключ користувача, який автоматично генерується при створенні нового запису в колекції;
- name: ім’я користувача в системі;
- email: електронна пошта користувача, яка є унікальним значенням та використовується для авторизації в системі;
- password: пароль користувача, який зберігається в захешованому вигляді та використовується для підтвердження особи користувача при вході у систему;
- uniqueIndicator: унікальний ідентифікатор, який користувач може створити для можливості спілкування з іншими користувачами.
- status: відповідає за роль користувача (звичайний користувач, менеджер закладу, адміністратор платформи), який є дійсним тільки після перевірки відповідних записів, крім звичайного користувача, у інших колекціях і чи є дані підтверджені адміністратором платформи;
- dOB: дата народження користувача;
- phone: номер мобільного телефону користувача;

- avatar: зберігає посилання на зображення профілю користувача для відображення у чатах;
- isActivated: відповідає чи був профіль користувача підтверджений за допомогою електронної пошти;
- phoneVerify: відповідає чи був профіль користувача підтверджений за допомогою мобільного номеру;
- verifyCode: код, який використовується для підтвердження мобільного номеру;
- activationLink: посилання, яке використовується для підтвердження електронної пошти;
- registerBy: тип реєстрації користувача в системі, чи була здійснена реєстрація за допомогою наступних методів: email + password;
- blocked: містить наступні поля:
  - isBlocked: відповідає за те, чи був профіль користувача заблокований;
  - whyBlock: вказує причину, через що профіль користувача був заблокований;
- createdAt: дата коли користувач зареєструвався у системі;
- updatedAt: дата коли було останній раз змінено дані користувача.

Колекція establishments (заклади) – заклади, містить всю необхідну інформацію про заклад:

- _id: унікальний ключ закладу, який автоматично генерується при створенні нового запису в колекції;
- title: назва закладу;
- pictures: зображення закладу;
- workSchedule: відповідає за робочі дні та години закладу, а також показує вихідні дні;
- location: місцезнаходження закладу за координатами;
- place: населений пункт та адреса де знаходиться заклад;
- type: тип закладу;

- description: опис закладу;
- cuisine: тип кухні закладу;
- contacts: контакти закладу;
- tags: теги закладу;
- verify: відповідає чи був заклад перевірений адміністратором платформи;
- verifyBy: зовнішній ключ, який посилається на адміністратора, який було перевірено заклад;
- rating: рейтинг закладу;
- averageCheck: середній чек закладу;
- features: особливості закладу;
- reviewsLength: кількість відгуків закладу;
- newsLength: кількість новин закладу;
- commentsLength: кількість коментарів закладу;
- createdBy: зовнішній ключ, який посилається на менеджера закладу;
- seats: зовнішній ключ, який посилається на інформацію про місця закладу;
- viewsNumber: кількість переглядів закладу користувачами на платформі, від одного користувача може бути тільки один перегляд;
- visits: кількість відвідувань закладу користувачами на платформі, від одного користувача може бути тільки один перегляд протягом дня;
- sendNotifications: показує чи надсилати сповіщення користувачам про нову новину закладу, за умови, що користувачі підписані на розсилку;
- createdAt: дата коли було додано заклад;
- updatedAt: дата коли було останній раз змінено дані закладу.

Колекція establishmentNews (новини закладу) – новини, містить всю необхідну інформацію про новину:

- _id: унікальний ключ новини, який автоматично генерується при створенні нового запису в колекції;
- title: назва новини;
- pictures: зображення новин;
- place: місцезнаходження, населений пункт та адреса де знаходиться заклад;
- category: категорія новини;
- description: опис новини;
- status: відповідає чи є новина публічною;
- createdBy: зовнішній ключ, який посилається на менеджера закладу;
- viewsNumber: кількість переглядів новини користувачами на платформі, від одного користувача може бути тільки один перегляд;
- visits: кількість відвідувань новини користувачами на платформі, від одного користувача може бути тільки один перегляд протягом дня;
- publishAt: вказує чи опублікувати новину і дату коли має відбутися публікація;
- dateEvent: вказує коли буде дата події, якщо така є;
- createdAt: дата коли було додано новину;
- updatedAt: дата коли було останній раз змінено дані новини.

Колекція oauths (сесії користувача) – зберігає інформацію, пов'язану з аутентифікацією користувача. Вона містить токени доступу, токени оновлення, а також деталі про пристрій і браузер, з якого був здійснений вхід:

- _id: унікальний ключ сесії, який автоматично генерується при створенні нового запису в колекції;

- userId: зовнішній ключ, який посилається на користувача;
- access_token: токен доступу;
- refresh_token: токен оновлення;
- userAgent: об'єкт, що містить інформацію про браузер і пристрій, з якого здійснювався вхід:
  - browser: деталі про браузер:
    - name: назва браузера;
    - version: повна версія браузера;
    - major: основна версія браузера;
  - device: деталі про пристрій:
    - model: модель пристрою;
    - type: тип пристрою;
    - vendor: виробник пристрою;
  - engine: деталі про рушій браузера:
    - name: назва рушія браузера;
    - version: версія рушія браузера;
  - os: деталі про операційну систему:
    - name: назва операційної системи;
    - version: версія операційної системи;
- createdAt: дата коли було додано новину;
- updatedAt: дата коли було останній раз змінено дані новини.

Колекція capls (резервування) – зберігає інформацію, пов'язану з резервуванням:

- _id: унікальний ключ резервування, який автоматично генерується при створенні нового запису в колекції;
- user: зовнішній ключ, який посилається на користувача;
- manager: зовнішній ключ, який посилається на менеджера закладу;
- establishment: зовнішній ключ, який посилається на заклад;
- fullName: ім’я користувача на яке створено резервування;
- eventType: подія, для якої створюється резервування;

- date: дата на коли створено резервування;
- comment: коментар користувача, щодо резервування;
- writeMe: відповідає за те, чи може заклад написати користувачеві, для уточнення деталей;
- desiredAmount: приблизна сума витрат, на яку розраховує користувач;
- numberPeople: кількість гостей;
- whoPay: ПІБ особи, хто буде платити;
- userRejected: скасування резервування користувачем:
  - isRejected: чи було скасовано резервування;
  - reasonRefusal: причина скасування;
- establishmentStatus: статус закладу про прийняття резервування:
  - value: значення статусу;
  - freeDateFor: вільні дати у разі відмови від бронювання;
  - reasonRefusal: причина відмови;
- isActive: вказує чи є резервування ще дійсним:
  - status: значення статусу;
  - updatedBy: вказує чи статус було оновлено автоматично чи його змінив менеджер закладу;
- isClientAppeared: вказує чи з'явився користувач до закладу, щодо резервування;
- createdAt: дата коли було додано резервування;
- updatedAt: дата коли було останній раз змінено дані резервування.

3.3. Синхронізація даних

Для забезпечення безпечної та ефективної взаємодії з базою даних використовуються різні технології та підходи. Зокрема, реалізовано синхронізацію даних у режимі реального часу за допомогою веб-сокетів та технології Socket.IO. Це дозволяє відображати актуальну інформацію про

бронювання, новини та інші зміни для всіх підключених користувачів без затримок.

Крім того, для забезпечення безпеки та аутентифікації користувачів використовується JSON Web Token (JWT). Після успішної аутентифікації користувачеві видається JWT-токен, який містить актуальні дані про користувача. Цей токен використовується для перевірки автентифікації при кожному наступному запиті до серверної частини системи. При кожному новому запиті з дійсним JWT-токеном сервер отримує оновлені дані про користувача, гарантуючи, що інформація про нього завжди є актуальною. Це забезпечує безпечний доступ до даних лише авторизованим користувачам та дозволяє зберігати точну інформацію про їх стан та налаштування.

Ретельно продумана структура бази даних у поєднанні з технологіями синхронізації даних та аутентифікації на основі JWT забезпечує надійне та ефективне зберігання, доступ та обмін даними в рамках онлайн-платформи для бронювання місць у закладах харчування. Актуальні дані про користувачів завжди доступні завдяки оновленню цієї інформації при кожному запиті з дійсним JWT-токеном.

3.4. Опис алгоритмів платформи

Онлайн-платформа для бронювання місць у закладах харчування реалізує низку алгоритмів для забезпечення основної функціональності, такої як реєстрація та авторизація користувачів, перегляд інформації про заклади, резервування місць та взаємодія між користувачами й менеджерами закладів. У цьому розділі будуть детально описані ключові алгоритми, що лежать в основі роботи платформи.

Алгоритм реєстрації користувача:

1. Користувач вводить необхідні дані для реєстрації (ім'я, електронну пошту, пароль тощо) на сторінці реєстрації.

2. Клієнтська частина додатку відправляє ці дані на сервер для перевірки та створення нового облікового запису.
3. Сервер перевіряє, чи не існує вже користувача з наданою електронною поштою в базі даних.
4. Якщо користувач з такою електронною поштою не знайдений, пароль хешується та зберігається в базі даних разом з іншими даними користувача.
5. Користувача перенаправляє на сторінку авторизації.
6. Клієнтська частина додатку відображає підтвердження успішної реєстрації користувачеві.
7. Користувача перенаправляє на сторінку авторизації.

Алгоритм бронювання місць:

1. Авторизований користувач переглядає список доступних закладів та обирає той, де він бажає забронювати місце.
2. Користувач вказує бажану дату та час бронювання, кількість місць, а також інші необхідні дані.
3. Клієнтська частина додатку відправляє запит на сервер для перевірки доступності бронювання на вказану дату та час та кількість людей.
4. Сервер перевіряє наявність вільних місць у закладі на вказаний період часу, якщо місця доступні, сервер зберігає нове бронювання в базі даних, відправляє відповідь про створене бронювання, як користувачеві так і менеджеру закладу.
5. Клієнтська частина додатку відображає підтвердження успішного створення бронювання користувачеві та оновлює дані в реальному часі за допомогою Socket.IO.
6. Менеджер закладу отримує інформацію про нове резервування місць, перевіряє його, за відсутності проблем приймає резервування.

7. Клієнтська частина відправляє на сервер запит про підтвердження резервування, сервер відправляє користувачеві відповідь про успішне підтвердження резервування.

Алгоритм взаємодії в чаті:

1. Користувач або менеджер закладу ініціює новий чат або відкриває існуючий.
2. Клієнтська частина додатку встановлює з'єднання Socket.IO із сервером.
3. При введенні нового повідомлення клієнтська частина відправляє його на сервер через Socket.IO.
4. Сервер отримує повідомлення та зберігає його в базі даних, пов'язуючи з відповідним чатом.
5. Сервер відправляє нове повідомлення через Socket.IO всім клієнтам, підключеним до цього чату.
6. Клієнтська частина додатку отримує повідомлення та відображає його в інтерфейсі чату в реальному часі.

3.5. Висновки до розділу 3

Онлайн-платформа для резервування місць у закладах харчування поєднує сучасні технології та передові підходи до розробки програмного забезпечення. Ретельно спроектована архітектура системи забезпечує продуктивність, масштабованість, надійність та безпеку.

Серверна частина, побудована на Node.js, Express.js та MongoDB, відповідає за обробку запитів, взаємодію з базою даних, реалізацію бізнес-логіки та забезпечення безпеки даних. Клієнтська частина, створена з використанням React та TypeScript, надає зручний та інтерактивний користувацький інтерфейс, забезпечуючи швидку та плавну роботу.

Структура бази даних ретельно продумана та складається з взаємопов'язаних колекцій, що зберігають інформацію про користувачів, заклади, бронювання, новини та інші сутності. Синхронізація даних у

режимі реального часу реалізована за допомогою веб-сокетів та технології Socket.IO, забезпечуючи відображення актуальної інформації для всіх підключених користувачів.

Ключові алгоритми, такі як реєстрація користувачів, бронювання місць та взаємодія в чатах, детально описані та забезпечують безперебійну роботу основної функціональності платформи. Використання передових технологій, таких як Docker для контейнеризації та Nginx для керування навантаженням, гарантує ефективне розгортання та масштабування системи.

Cloudinary, потужний сервіс для хмарного зберігання та обробки зображень, інтегрований у платформу для забезпечення надійного та оптимізованого управління візуальним контентом. Він дозволяє безпечно зберігати зображення закладів, новин та аватарів користувачів, автоматично оптимізуючи їх для різних розмірів екранів та пристроїв. Крім того, Cloudinary надає зручні інструменти для редагування, обрізання, додавання ефектів та фільтрів до зображень, покращуючи візуальний досвід для користувачів платформи.

Загалом, онлайн-платформа для бронювання місць у закладах харчування є сучасним і добре спроектованим веб-додатком, який відповідає високим стандартам якості та забезпечує зручний та безпечний досвід для користувачів.

4. АНАЛІЗ РОЗРОБЛЕНОЇ ПЛАТФОРМИ

4.1. Особливості реалізації платформи

Платформа для онлайн-резервування місць у закладах харчування була розроблена з акцентом на зручність користування, повнофункціональність та відповідність сучасним вимогам до веб-додатків. Під час реалізації були впроваджені численні особливості та функціональні можливості, які забезпечують безперешкодний і приємний досвід взаємодії з платформою для всіх зацікавлених сторін.

Користувацький інтерфейс платформи був ретельно спроектований з урахуванням принципів інтуїтивного та зрозумілого дизайну. Завдяки продуманій навігації та зручним елементам керування, користувачі можуть легко орієнтуватися у функціоналі додатку та ефективно виконувати необхідні дії.

Одним із пріоритетів під час розробки стало забезпечення високої інтерактивності та оновлення даних у режимі реального часу. Це дозволяє всім учасникам процесу бронювання отримувати актуальну інформацію без затримок, підвищуючи ефективність комунікації та прийняття рішень.

Крім базової функціональності, пов'язаної з переглядом закладів та створенням бронювань, платформа пропонує низку додаткових можливостей, спрямованих на підвищення зручності та персоналізацію досвіду користувача. Детальний опис цих особливостей наведено нижче.

4.1.1. Система сповіщень

Платформа інтегрує систему сповіщень, яка дозволяє надсилати повідомлення користувачам у різних ситуаціях, пов'язаних з бронюванням місць. Для зручного перегляду та взаємодії зі сповіщеннями, не актуальні можна додати до корзини, після чого остаточно видалити. Сповіщення для користувачів генеруються у таких випадках:

1. Новина закладу: коли заклад публікує нову новину, система відправляє сповіщення користувачам, які підписалися на розсилку сповіщень від закладу.
2. Нове бронювання: після успішного створення бронювання користувач і менеджер закладу отримують відповідні сповіщення з деталями резервування.
3. Зміна статусу бронювання: при зміні статусу існуючого бронювання (підтвердження, скасування тощо) задіяні сторони отримують повідомлення про це.

Крім згаданих раніше випадків генерування сповіщень, система також реалізує функціональність нагадування про бронювання. За годину до забронюваного часу відвідування закладу, користувачам надсилається нагадувальне сповіщення на електронну пошту та у веб-інтерфейс платформи.

Ця функціональність реалізована за допомогою бібліотеки node-schedule, яка дозволяє планувати виконання завдань у Node.js за встановленим розкладом.

Процес нагадування відбувається наступним чином:

1. Під час створення нового бронювання, інформація про його час, користувача та електронну адресу зберігається у базі даних.
2. За допомогою node-schedule створюється запланована задача, яка виконується за годину до забронюваного часу.
3. У цій задачі відбувається пошук відповідного бронювання у базі даних та вилучення необхідних даних користувача.
4. Генерується тіло нагадувального сповіщення з деталями бронювання.
5. Сповіщення надсилається на зареєстровану електронну адресу користувача за допомогою інтегрованої служби електронної пошти NodeMailer.

6. Паралельно сповіщення також відображається у веб-інтерфейсі платформи після авторизації користувача.

Node-schedule надає зручний спосіб створення запланованих завдань у Node.js, використовуючи кронові вирази або об'єкти Date для визначення часу виконання. Це дозволяє легко налаштувати відправку нагадувань за вказаний період до події бронювання.

Впровадження цієї функціональності забезпечує своєчасне інформування користувачів про їх заплановані відвідування закладів та допомагає уникнути пропущених чи забутих бронювань.

4.1.2. Управління збереженими закладами та новинами

Платформа надає користувачам зручні інструменти для управління вибраними закладами та новинами. Кожен зареєстрований користувач має персональний профіль, у якому можна:

1. Додавати вибрані заклади та новини до окремого списку для швидкого доступу.
2. Видаляти заклади та новини зі списку вибраного, якщо вони більше не представляють інтерес.
3. Додавати нові назви та описи до вибраних закладів та новин для полегшення пошуку.

4.1.3. Відображення закладів на карті

Для забезпечення зручності пошуку та бронювання місць платформа інтегрована з інтерактивною картою. Ця функціональність дозволяє:

1. Відображати місцезнаходження всіх доступних закладів на карті.
2. Переглядати детальну інформацію про заклад при натисканні на його маркер.
3. Застосовувати фільтри для відображення лише закладів певного типу чи кухні.

4. Отримувати маршрут до обраного закладу від поточного місцезнаходження користувача.

4.1.4. Спілкування в чатах у режимі реального часу

Одна з ключових особливостей платформи – можливість спілкуватися у режимі реального часу між користувачами та менеджерами закладів через вбудовану систему чатів. Ця функціональність реалізована з використанням технології Socket.IO і дозволяє:

1. Створювати окремі чати для кожної пари: “Користувач – Менеджер закладу”, “Користувач – Користувач”, “Користувач – Менеджер закладу, щодо конкретного резервування”.
2. Обмінюватися текстовими повідомленнями у режимі реального часу.
3. Надсилати мультимедійні файли (фото) в рамках чату.
4. Отримувати сповіщення про нові повідомлення в активних чатах

4.2. Тестування платформи

Для забезпечення належної якості та стабільної роботи платформи для онлайн-резервування місць, було проведено комплексне тестування. Основною метою тестування було виявлення та усунення потенційних помилок, дефектів та невідповідностей у функціонуванні системи.

Під час тестування було застосовано переважно ручні методики перевірки функціоналу, які передбачали детальне відтворення дій користувача через веб-інтерфейс платформи. Ця стратегія дозволила ретельно протестувати всі аспекти програми, від процесів реєстрації, авторизації та керування обліковими записами до створення, редагування та видалення елементів, таких як заклади, бронювання, новини тощо.

Особливу увагу було приділено тестуванню критично важливих компонентів платформи, зокрема системи сповіщень і нагадувань про бронювання, а також функціональності чатів і відображення закладів на

картах. Ці модулі відіграють ключову роль у забезпеченні безперебійної взаємодії між користувачами та закладами, а також зручності використання платформи.

Під час тестування було не лише перевірено очікувані результати в користувацькому інтерфейсі, але й гарантовано правильність запису та збереження даних у базі даних MongoDB та коректність обробки даних на стороні сервера Node.js. Такий підхід дозволив отримати комплексну оцінку роботи додатку та виявити потенційні проблеми, що потребували вдосконалення або виправлення.

4.2.1. Аутентифікація та авторизація

Ця критично важлива частина платформи забезпечує безпечний доступ користувачів до їхніх облікових записів та відповідних функцій залежно від ролі (користувач чи менеджер закладу). Комплексне тестування процесів аутентифікації та авторизації було проведено для гарантування належного захисту даних та функціонування системи.

- Тестовий випадок 1.1 (реєстрація нового користувача):
  - опис:
    - користувач на сторінці реєстрації вводить свої особисті дані: повне ім'я, електронну пошту, номер телефону, дату народження, пароль та вибирає роль (користувач чи менеджер);
    - користувач натискає кнопку "Зареєструватись", система перевіряє коректність введених даних, включаючи формат електронної пошти та відповідність пароля вимогам до складності;
    - якщо всі дані введені правильно, система створює новий обліковий запис користувача в базі даних;
  - очікуваний результат:

        - користувач успішно зареєстрований, про що сповіщає відповідне повідомлення в інтерфейсі;
        - у базі даних створено новий запис з інформацією про користувача.
- Тестовий випадок 1.2 (авторизація існуючого користувача):
    - опис:
        - користувач вводить свої облікові дані (електронну пошту та пароль) на сторінці авторизації;
        - система перевіряє правильність введених даних, порівнюючи з записом в базі даних;
        - у випадку успішної перевірки користувачу надається доступ до свого облікового запису та відповідних функцій;
    - очікуваний результат:
        - користувач успішно авторизований і отримує доступ до функціоналу платформи згідно своєї ролі;
        - у локальному сховищі браузера зберігається інформація користувача та токени доступу для подальшого керування.
- Тестовий випадок 1.3 (неправильні облікові дані):
    - опис:
        - користувач вводить невірні облікові дані (неіснуюча електронна пошта або неправильний пароль) під час спроби авторизації;
    - очікуваний результат:
        - система відхиляє спробу авторизації та відображає відповідне повідомлення про помилку для користувача.

Кожен з цих тестових випадків був ретельно перевірений як для ролі користувача, так і для ролі менеджера закладу. Особливу увагу було приділено тестуванню вимог до пароля, коректності обробки введених даних, а також дотриманню принципів безпеки, зокрема захисту від атак перебору паролів.

Тестування підтвердило належне функціонування системи аутентифікації та авторизації на платформі, забезпечуючи надійний захист облікових даних користувачів та доступ до функцій відповідно до визначених ролей.

4.2.2. Створення та редагування бронювань

Ключовою функціональністю платформи є можливість створення, перегляду, редагування та скасування бронювань місць у закладах харчування. Тестування цього модуля було критично важливим для забезпечення належної роботи основних бізнес-процесів системи.

- Тестовий випадок 2.1 (створення нового бронювання, з боку користувача):
  - опис:
    - авторизований користувач переходить до сторінки вибраного закладу та натискає кнопку "Забронювати місце". Він вводить необхідні дані: бажану дату та час бронювання, кількість місць, додаткові вимоги або коментар;
    - після підтвердження система перевіряє доступність вказаного часу та, у разі успіху, створює нове бронювання;
  - очікуваний результат:
    - нове бронювання успішно створене та відображається в особистому кабінеті користувача;
    - користувач та менеджер закладу отримують сповіщення про бронювання;
    - у базі даних створено новий запис про бронювання.
- Тестовий випадок 2.2 (редагування бронювання, з боку менеджера закладу):
  - опис:
    - менеджер авторизованого закладу переглядає список активних бронювань у адмін-панелі;

        - він вибирає потрібне бронювання та змінює певні деталі, як от кількість місць або додаткові вимоги;
        - після збереження змін система оновлює інформацію про бронювання;
    - очікуваний результат:
        - дані бронювання успішно оновлені, про що сповіщає відповідне повідомлення;
        - користувач отримує сповіщення про зміну деталей бронювання;
        - у базі даних оновлено запис бронювання.
- Тестовий випадок 2.3 (скасування бронювання, з боку користувача):
    - опис:
        - користувач на сторінці свого облікового запису переглядає список активних бронювань;
        - вибирає бронювання, яке необхідно скасувати, та натискає відповідну кнопку;
        - система вимагає підтвердження та після нього скасовує бронювання;
    - очікуваний результат:
        - бронювання успішно скасоване, клієнт та заклад отримують сповіщення;
        - запис про бронювання видаляється з бази даних або позначається як скасований.

Під час тестування особливу увагу було приділено перевірці валідації вхідних даних, обробці конфліктів бронювань, а також тестуванню сценаріїв із паралельними діями користувачів та менеджерів над одним бронюванням. Також було перевірено відповідність відображених даних бронювань записам у базі даних.

Комплексне тестування підтвердило стабільність та коректність функціонування модулів створення, редагування та скасування бронювань на платформі, дозволяючи користувачам та менеджерам закладів ефективно керувати процесом резервування місць.

4.2.3. Система сповіщень та нагадувань

Система сповіщень та нагадувань відіграє ключову роль у забезпеченні зручної взаємодії між користувачами та закладами на платформі. Тестування цього модуля було спрямоване на перевірку своєчасної доставки повідомлень у різних сценаріях та гарантування їх коректного відображення.

- Тестовий випадок 3.1 (сповіщення про нову новину закладу):
  - опис:
    - менеджер авторизованого закладу створює нову новину в адмін-панелі та публікує її;
    - система має згенерувати та надіслати сповіщення всім користувачам, які підписалися на розсилку новин цього закладу;
  - очікуваний результат:
    - підписані користувачі отримують сповіщення з короткою інформацією про нову новину у веб-інтерфейсі платформи;
    - повний текст новини доступний для перегляду у відповідному розділі.
- Тестовий випадок 3.2 (сповіщення про нове бронювання):
  - опис:
    - користувач створює нове бронювання для певного закладу;
    - система має згенерувати сповіщення як для користувача, так і для менеджера цього закладу з деталями бронювання;
  - очікуваний результат:

        - користувач і менеджер закладу отримують відповідні сповіщення про успішне створення бронювання у веб-інтерфейсі платформи.
- Тестовий випадок 3.3 (нагадування про бронювання):
    - опис:
        - за встановлений період часу (наприклад, за 1 годину) до забронюваного часу система має згенерувати нагадувальне сповіщення про бронювання та надіслати його користувачу;
    - очікуваний результат:
        - користувач отримує нагадувальне сповіщення про своє майбутнє бронювання на зареєстровану електронну пошту та у веб-інтерфейсі платформи.

Під час тестування приділялася особлива увага перевірці своєчасності доставки сповіщень, коректності відображення їх вмісту, а також тестуванню різних сценаріїв генерування сповіщень (новини, бронювання, зміни статусів). Окремо було протестовано функціонування планувальника завдань (node-schedule) для забезпечення відправки нагадувань у потрібний час.

Тестування системи сповіщень та нагадувань підтвердило її належну роботу, забезпечуючи ефективну комунікацію між учасниками процесу бронювання та високий рівень зручності для користувачів платформи.

4.3. Рекомендації щодо подальшого вдосконалення

Архітектура веб-платформи для онлайн-бронювання місць була ретельно спроектована з акцентом на гнучкість та можливість масштабування. Кожен компонент і модуль відокремлені та незалежні, що дозволяє додавати нові функції без необхідності значних змін в існуючому коді або змінювати окремі модулі на альтернативні рішення.

Такий підхід до проектування забезпечує платформі високу адаптивність та відкриває шлях для майбутніх вдосконалень та розширень

функціоналу. Після аналізу відгуків перших користувачів, експертних оцінок та огляду сучасних тенденцій в галузі, були виокремлені кілька перспективних напрямків для подальшого вдосконалення платформи, які могли б підвищити її корисність, зручність та привабливість для користувачів.

4.3.1. Додавання меню

Інтеграція можливості переглядати повне меню закладу перед бронюванням столика. Це дозволить користувачам заздалегідь ознайомитися з пропозиціями закладу, типами кухні та цінами. Меню має відображатися у зручному та інтерактивному вигляді з фільтрами за категоріями страв, можливістю сортування та пошуку.

4.3.3. Система персоналізованих рекомендацій

Впровадження системи персоналізованих рекомендацій закладів для користувачів може значно підвищити зручність використання платформи. Ця система використовуватиме алгоритми на основі машинного навчання для аналізу історії переглядів, розташування та минулих вподобань користувачів і надаватиме релевантні пропозиції нових гарних закладів поблизу.

4.3.3. Історія переглядів

Збереження детальної історії переглянутих закладів та новин для кожного облікового запису користувача є корисною функціональністю для полегшення повторного пошуку місць чи цікавого контенту в майбутньому. Історія має мати зручний інтерфейс з фільтрами за різними параметрами, наприклад, типом закладу, кухнею, датою перегляду тощо.

4.3.4. Ділитися бронюваннями

Додавання функціональності ділитися інформацією про бронювання з іншими користувачами платформи може стати зручним засобом для сумісного планування відвідувань закладів, наприклад, з членами сім'ї чи друзями. Користувач повинен мати можливість легко надіслати посилання на бронювання зі всіма деталями резервування (заклад, час, кількість місць тощо).

4.4. Висновки до розділу 4

У даному розділі було детально проаналізовано особливості реалізації платформи для онлайн-резервування місць у закладах харчування. Було розглянуто ключові функціональності, такі як система сповіщень, управління збереженими закладами та новинами, відображення закладів на карті та спілкування в чатах у режимі реального часу. Крім того, було описано процес комплексного тестування платформи, зокрема тестування аутентифікації та авторизації, створення та редагування бронювань, а також системи сповіщень та нагадувань.

Результати тестування підтвердили стабільну роботу та правильне функціонування всіх модулів платформи, забезпечуючи високу якість користувацького досвіду та ефективну взаємодію між користувачами та закладами.

Проте, для подальшого вдосконалення платформи та підвищення її привабливості для користувачів були запропоновані кілька рекомендацій. Серед них – вдосконалення користувацького інтерфейсу з урахуванням нових тенденцій та потреб різних груп користувачів, додавання можливості переглядати меню закладів, впровадження системи персоналізованих рекомендацій на основі машинного навчання, збереження історії переглядів користувачів та функціональність ділитися інформацією про бронювання з іншими користувачами.

Впровадження цих удосконалень може значно підвищити зручність використання платформи, розширити її функціональність та забезпечити кращий користувацький досвід для всіх зацікавлених сторін.

ВИСНОВКИ

Метою даного дипломного проєкту було створити онлайн-платформу для полегшення процесу бронювання місць у закладах харчування, таких як ресторани, кафе та бари. Результатом роботи стала онлайн-платформа, доступний через Інтернет.

У процесі виконання проєкту проаналізовано предметну галузь програмних рішень для онлайн-резервування місць у закладах харчування, виділено ряд проблем, з якими стикаються відвідувачі, досліджено наявні програмні рішення, висвітлено їх переваги та недоліки. Зібрано вимоги до системи та обрано інструменти для її реалізації: React з TypeScript для клієнтської частини, Node.js з Express.js та Socket.IO для серверної частини, MongoDB як систему керування базами даних. Порівнявши існуючі аналоги, виявлено, що жоден із сервісів не вирішує всі проблеми користувачів та не задовольняє їхні потреби повною мірою, тому розробка даної онлайн-платформи є актуальним завданням.

Розроблення виконано у повному обсязі відповідно до висунутих вимог, описаних у документації та технічному завданні. Тестування продукту здійснено згідно із затвердженою програмою та методикою тестування.

Платформа дозволяє реєструватися як звичайним користувачам, так і менеджерам закладів харчування. Користувачі можуть переглядати список доступних закладів, їх інформацію та новини, додавати заклади до вибраного списку. Крім того, реалізовано можливість спілкуватися з менеджерами закладів через вбудовану систему чатів у режимі реального часу.

Менеджери закладів мають можливість створювати, редагувати та видаляти інформацію про свої заклади та оновлення у вигляді новин. Вони також можуть взаємодіяти з користувачами через чати та керувати бронюваннями.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Reservble – пошук та резерв стола у закладах міста [Електронний ресурс]. — Режим доступу: https://www.reservble.com/ — (23.01.2024).
2. Restoran.ua – ресторани України [Електронний ресурс]. — Режим доступу: https://restoran.ua/ — (25.01.2024).
3. Резерв столика онлайн в ресторане, кафе, пабе [Електронний ресурс]. — Режим доступу: https://reston.ua/ — (28.01.2024).
4. SQL Introduction [Електронний ресурс]. — Режим доступу: https://www.w3schools.com/sql/sql_intro.asp — (02.02.2024).
5. PostgreSQL: About [Електронний ресурс]. — Режим доступу: https://www.postgresql.org/about/ — (03.02.2024). https://www.oracle.com/mysql/what-is-mysql/
6. What Is MySQL? | Oracle [Електронний ресурс]. — Режим доступу: https://www.postgresql.org/about/ — (04.02.2024).
7. Introduction to NoSQL [Електронний ресурс]. — Режим доступу: https://www.geeksforgeeks.org/introduction-to-nosql/ — (06.02.2024).
8. What Is MongoDB? | MongoDB [Електронний ресурс]. — Режим доступу: https://www.mongodb.com/company/what-is-mongodb — (08.02.2024).
9. Introduction to Node.js [Електронний ресурс]. — Режим доступу: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs — (10.02.2024).
10. Express.js [Електронний ресурс]. — Режим доступу: https://expressjs.com/ — (11.02.2024).
11. Documentation – TypeScript for JavaScript Programmers [Електронний ресурс]. — Режим доступу: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html — (13.02.2024).

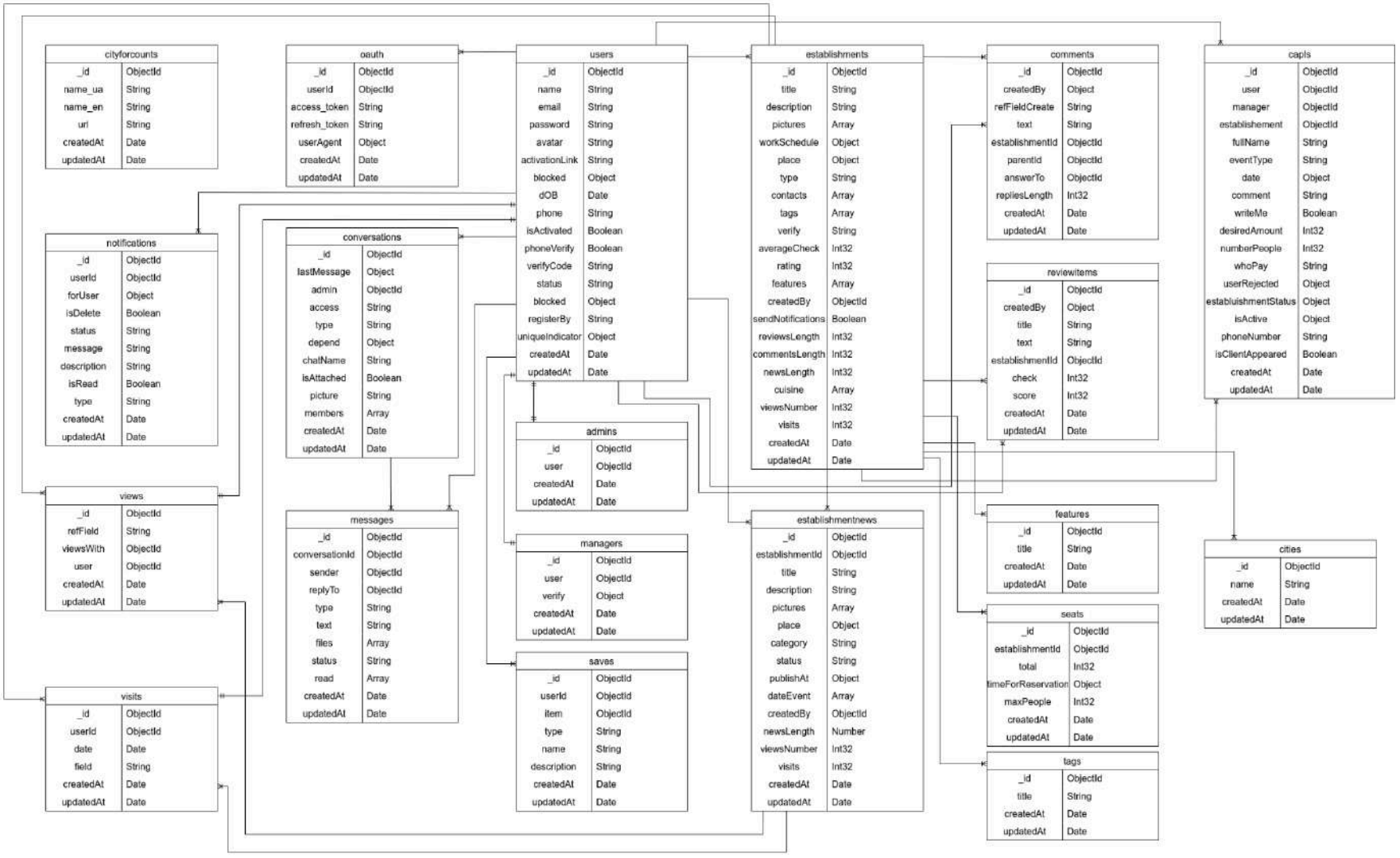
12. What Is Java? [Електронний ресурс]. — Режим доступу: https://www.ibm.com/topics/java — (13.02.2024).
13. Spring Boot [Електронний ресурс]. — Режим доступу: https://spring.io/projects/spring-boot — (13.02.2024).
14. About Python [Електронний ресурс]. — Режим доступу: https://www.python.org/about/ — (14.02.2024).
15. Django: The web framework for perfectionists with deadlines [Електронний ресурс]. — Режим доступу: https://www.djangoproject.com/ — (14.02.2024).
16. The V8 JavaScript Engine [Електронний ресурс]. — Режим доступу: https://nodejs.org/en/learn/getting-started/the-v8-javascript-engine — (16.02.2024).
17. React [Електронний ресурс]. — Режим доступу: https://react.dev// — (20.02.2024).
18. Angular – What is Angular? [Електронний ресурс]. — Режим доступу: https://v17.angular.io/guide/what-is-angular — (22.02.2024).
19. Introduction | Vue.js [Електронний ресурс]. — Режим доступу: https://vuejs.org/guide/introduction — (24.02.2024).
20. Understanding Virtual DOM in React | Refine [Електронний ресурс]. — Режим доступу: https://refine.dev/blog/react-virtual-dom/ — (25.02.2024).
21. Vite | Next Generation Frontend Tooling [Електронний ресурс]. — Режим доступу: https://vitejs.dev/ — (28.02.2024).
22. ECMAScript modules | Node.js v22.2.0 Documentation [Електронний ресурс]. — Режим доступу: https://nodejs.org/api/esm.html — (01.03.2024).
23. SPA (Single-page application) – MDN Web Docs Glossary [Електронний ресурс]. — Режим доступу: https://developer.mozilla.org/en-US/docs/Glossary/SPA — (04.03.2024).

24. Docker overview [Електронний ресурс]. — Режим доступу: https://docs.docker.com/get-started/overview/ — (07.03.2024).
25. Introduction | Socket.IO [Електронний ресурс]. — Режим доступу: https://socket.io/docs/v4/ — (10.03.2024).
26. What is Nginx: Everything You Need to Know [Електронний ресурс]. — Режим доступу: https://www.papertrail.com/solution/guides/nginx/ — (12.03.2024).

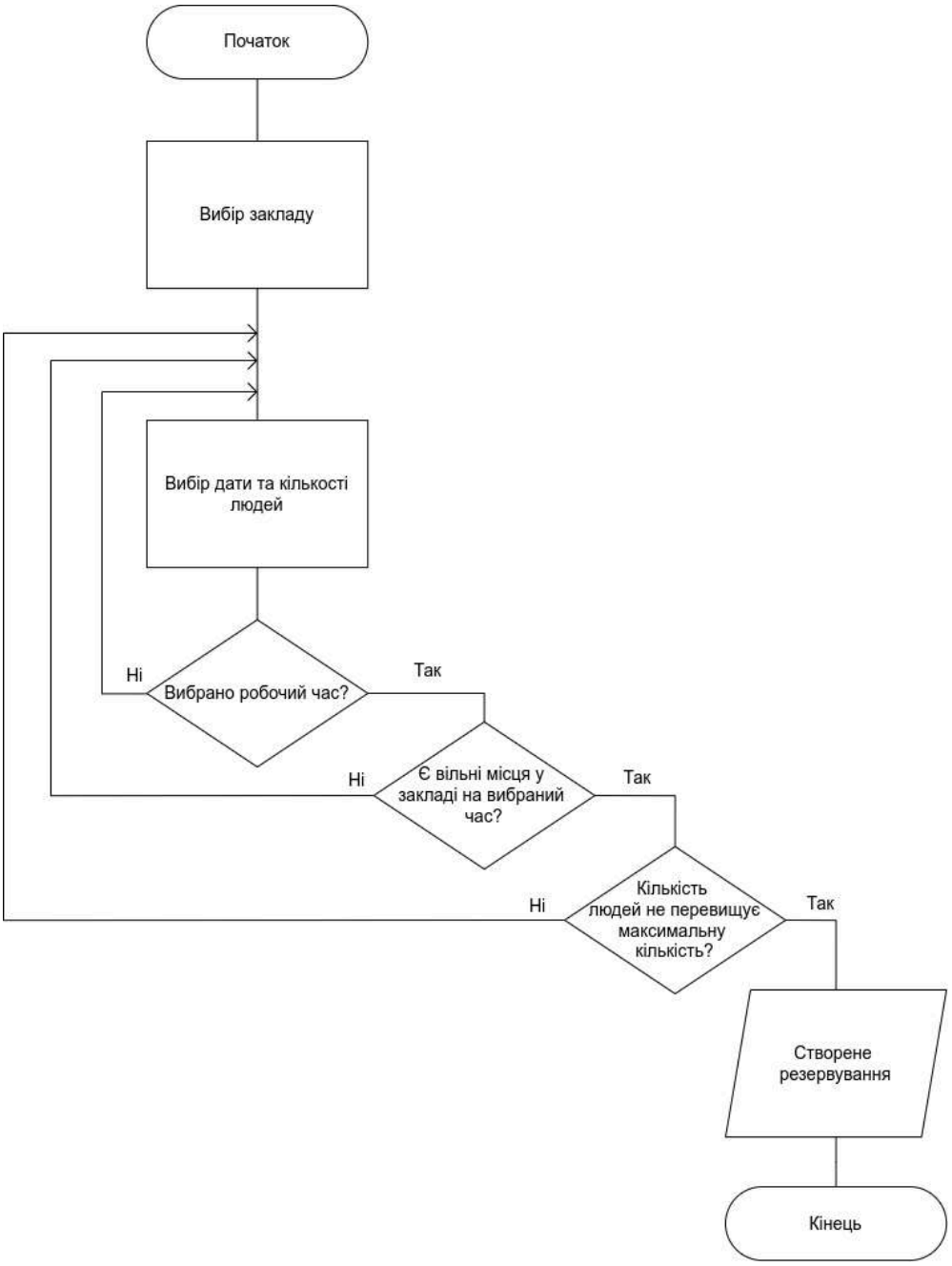
ДОДАТКИ

Додаток 1

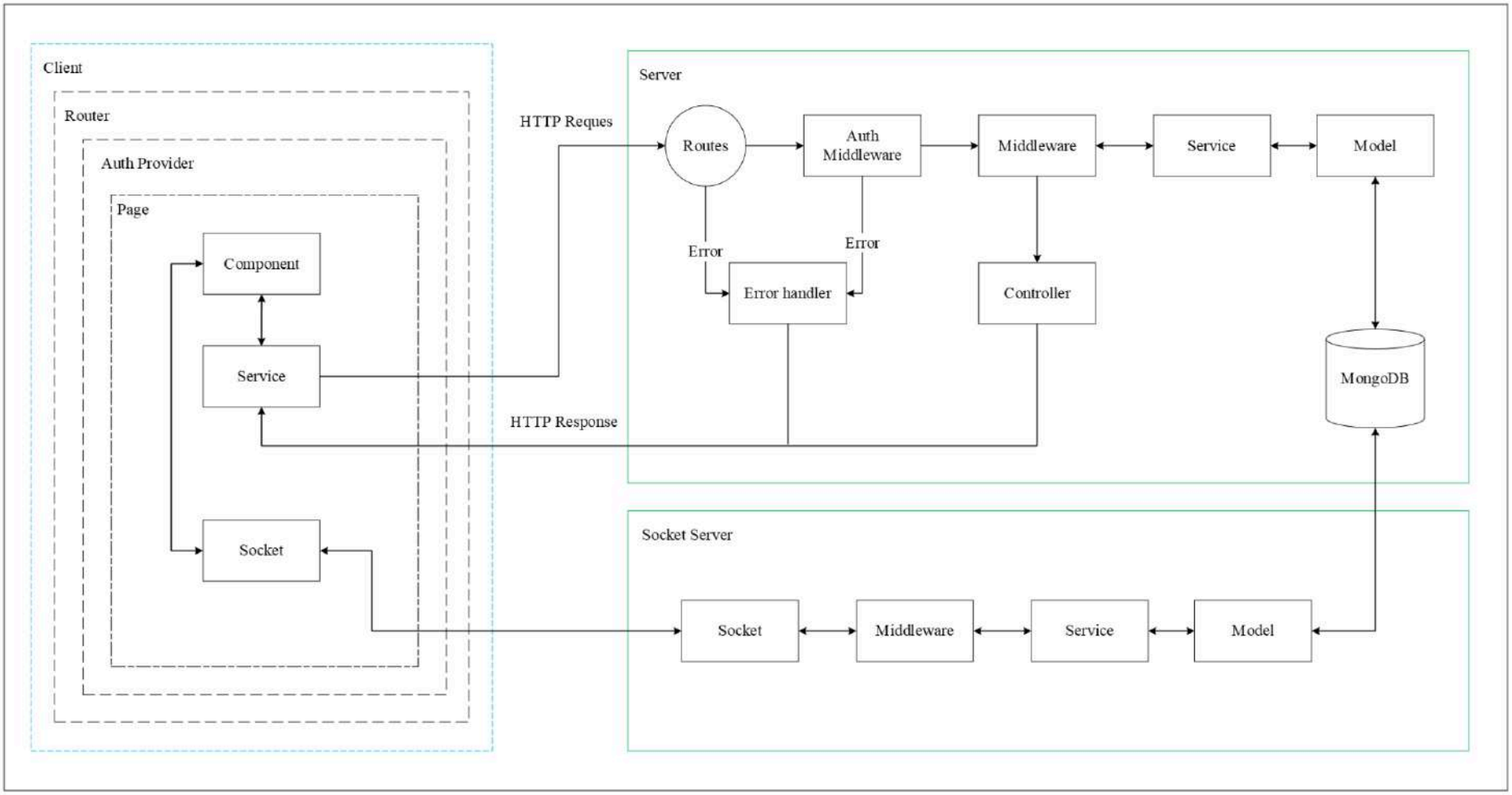
Копії графічних матеріалів



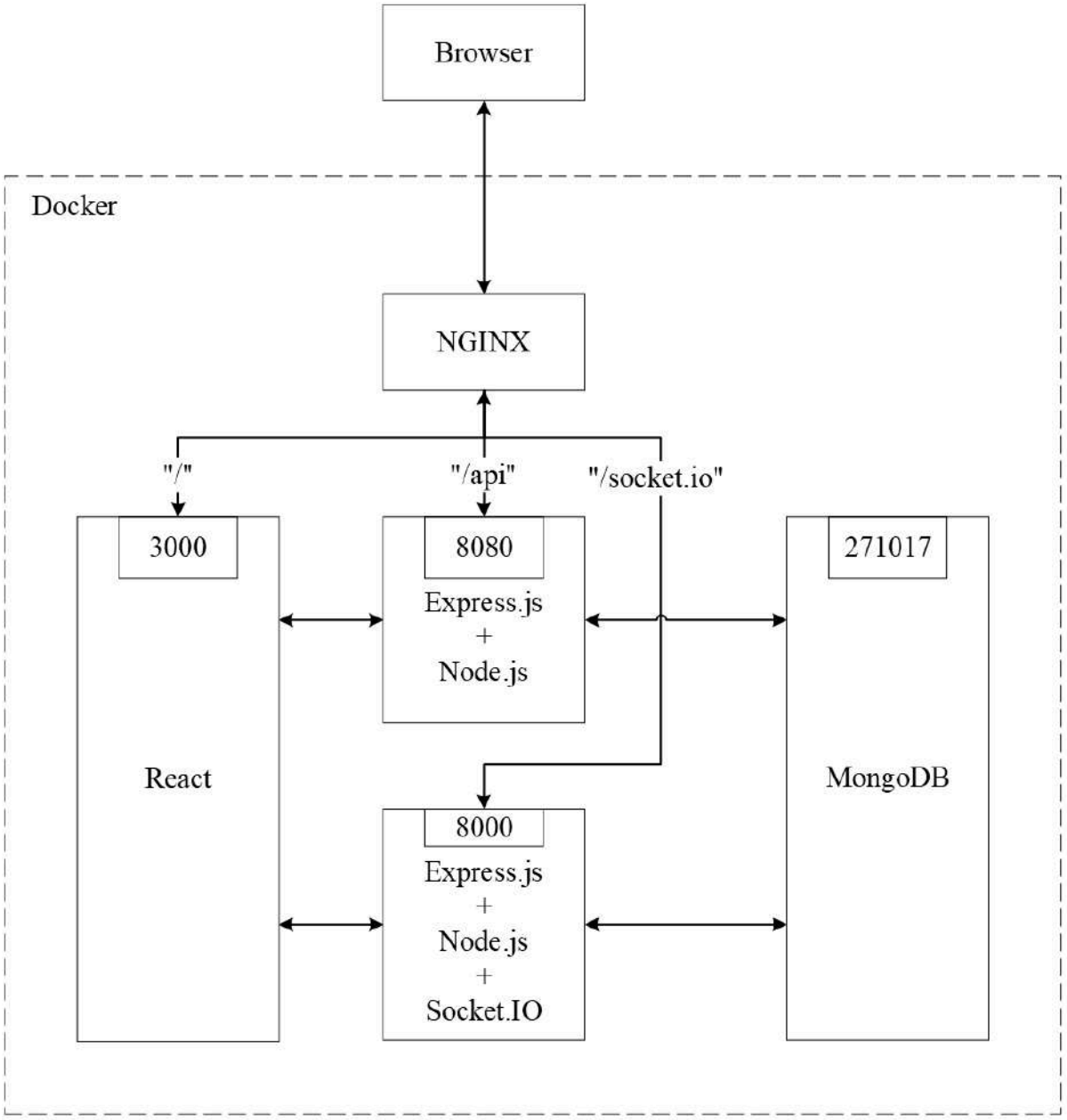
ДП.045440-06-99. Платформа для онлайн-резервування місць у закладах харчування. Структура бази даних. ER-діаграма



ДП.045440-07-99. Платформа для онлайн резервування місць у закладах харчування.
Алгоритм перевірки вільних місць.
Блок-схема алгоритму



Архітектура платформи
Якобчук Р.В., група КП-02



Інфраструктура платформи
Якобчук Р.В., група КП-02

Додаток 2

Лістинг програми

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1"/>
    <meta name="theme-color" content="#000000"/>
    <meta name="apple-mobile-web-app-capable" content="yes">
    <meta name="mobile-web-app-capable" content="yes">
    <meta
            name="description"
            content="CAPL"
    />
    <link rel="icon" href="./favicon.ico">
    <link rel="apple-touch-icon" href="./favicon.ico">
    <link rel="stylesheet"
href="https://unpkg.com/react-spring-bottom-sheet/dist/style.css"
crossorigin="anonymous">
    <link

href="https://fonts.googleapis.com/css2?family=Plus+Jakarta+Sans:wght@200;300;400;
500;600;700;800&family=Poppins:wght@100;200;300;400;500;600;700;800;900&display=sw
ap"
            rel="stylesheet"
    />
    <meta name="google-signin-client_id"

content={495583971120-ki01714kbouu8nciat4fpfo86g5e3ed2.apps.googleusercontent.com}
>
    <meta name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1">
    <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@100;200;300;400;500
;600;700;800;900&display=swap"
            rel="stylesheet"
    />
    <style>
        html {
            width: 100%;
            height: 100%;
            /*overflow: hidden;*/
        }
        body {
            overflow: hidden;
            width: 100%;
            height: 100%;
        }
        #root {
            height: 100%;
        }
        ul.MuiList-root.MuiMenu-list li.MuiMenuItem-root.Mui-selected {
            background-color: rgb(183 160 160 / 49%)
        }
    </style>
    <title>
        Capl
    </title>
    <script
            src="https://accounts.google.com/gsi/client"
            async
            defer
    ></script>
```

```
    <script src="https://apis.google.com/js/platform.js?onload=init" async
defer></script>
    <script src="https://accounts.google.com/gsi/client" async></script>
    <script src="https://apis.google.com/js/platform.js?onload=onLoadCallback"
async defer></script>
    <script src="https://accounts.google.com/gsi/client"></script>
    <script
src="https://maps.googleapis.com/maps/api/directions/json?origin=43.65077%2C-79.37
8425&destination=43.63881%2C-79.42745&key=495583971120-ki01714kbouu8nciat4fpfo86g5
e3ed2.apps.googleusercontent.com"></script>
    <script src="https://connect.facebook.net/en_US/sdk.js" async defer></script>
</head>
<body>
<div id="root"></div>
<script type="module" src="./src/main.tsx"></script>
</body>
</html>

import React from "react";
import {createRoot} from "react-dom/client";

import App from "./App";
import "./i18n";
import {Loading} from "./components";
import "./main.css";

const container = document.getElementById("root") as HTMLElement;
const root = createRoot(container);

root.render(
      <React.Suspense fallback={<Loading/>}>
         <App/>
      </React.Suspense>
);

import {RefineKbar, RefineKbarProvider} from "@refinedev/kbar";
import {Refine, I18nProvider} from "@refinedev/core";
import {BrowserRouter} from "react-router-dom";

import React from "react";

import {accessControlProvider} from "./accessControlProvider";
import {
   RefineSnackbarProvider,
} from "@refinedev/mui";
import {useNotificationProvider} from "@refinedev/antd"
import {CssBaseline, GlobalStyles} from "@mui/material";
import routerBindings, {
   UnsavedChangesNotifier,
} from "@refinedev/react-router-v6";
import dataProvider from "@refinedev/simple-rest";
import {GoogleOAuthProvider} from "@react-oauth/google";
import {useTranslation} from "react-i18next";
```

```
import {ColorModeContextProvider} from "./contexts";
import {authProvider, axiosInstance} from "./authProvider";
import {SchemaProvider} from "./settings/schema";
import {VariantProvider} from "./settings/variantEstablishment";
import {resources} from "./resources";
import {CommentCreatorDataProvider} from "./contexts/CommentCreatorDataContext";
import AppRoutes from "@/appRoutes";

const API_URL = import.meta.env.VITE_APP_API;
const STATISTIC_API_URL = import.meta.env.VITE_APP_STATISTIC_API;
const SOCKET_API_URL = import.meta.env.VITE_APP_SOCKET_API;
const clientId = `${import.meta.env.VITE_APP_GOOGLE_CLIENT_ID}`;

function App() {
    const {t, i18n} = useTranslation();

    const notificationProvider = useNotificationProvider();

    const i18nProvider: I18nProvider = {
        translate: (key: string, params: object) => t(key, params),
        changeLocale: (lang: string) => i18n.changeLanguage(lang),
        getLocale: () => i18n.language,
    };

    return (
        <BrowserRouter>
            <RefineKbarProvider>
                <ColorModeContextProvider>
                    <SchemaProvider>
                            <GoogleOAuthProvider clientId={clientId}>
                                <VariantProvider>
                                    <CssBaseline/>
                                    <GlobalStyles styles={{html: {WebkitFontSmoothing: "auto"}}}/>
                                    <RefineSnackbarProvider>
                                        <Refine
                                            dataProvider={{
                                                default: dataProvider(`${API_URL}/api/v1`, axiosInstance as
any),
                                                statistics:
dataProvider(`${STATISTIC_API_URL}/statistics_api/v1`, axiosInstance as any),
                                                socket: dataProvider(`${SOCKET_API_URL}/socket.io/api/v1`,
axiosInstance as any)
                                            }}
                                            notificationProvider={notificationProvider}
                                            resources={resources}
                                            accessControlProvider={accessControlProvider}
                                            routerProvider={routerBindings}
                                            authProvider={authProvider}
                                            i18nProvider={i18nProvider}
```

```
                                options={{
                                    syncWithLocation: true,
                                    mutationMode: 'undoable',
                                    liveMode: 'auto',
                                    warnWhenUnsavedChanges: true,
                                    useNewQueryKeys: true,
                                }}
                            >
                                <CommentCreatorDataProvider>
                                    <AppRoutes/>
                                </CommentCreatorDataProvider>
                                <RefineKbar/>
                                <UnsavedChangesNotifier/>
                            </Refine>
                        </RefineSnackbarProvider>
                    </VariantProvider>
                </GoogleOAuthProvider>
            </SchemaProvider>
        </ColorModeContextProvider>
    </RefineKbarProvider>
</BrowserRouter>
);
}

export default App;

import axios, {AxiosInstance, InternalAxiosRequestConfig} from "axios";
import {AuthProvider} from "@refinedev/core";

import {parseJwt} from "./utils";
import {IData, IGetIdentity, ProfileProps} from "./interfaces/common";
import {
    ACCESS_TOKEN_KEY,
    REFRESH_TOKEN_KEY
} from "./config/const";
import {clearUserAllData} from "@/hook/useUserLogout";

export const baseURL = `${import.meta.env.VITE_APP_API}/api/v1`;
export const axiosInstance: AxiosInstance = axios.create({
    baseURL, headers: {
        'Access-Control-Allow-Origin': `${import.meta.env.VITE_APP_API}`,
        'Content-Type': 'application/json;charset=UTF-8'
    }
});

function isAccessTokenExpired(access_token: string) {
    const token_a = parseJwt(access_token);
    const dateNow = new Date();
    if (token_a?.exp) {
        return token_a?.exp * 1000 > dateNow.getTime();
```

```
    } else {
        return false
    }
}

let _isRefreshing = false;
let _refreshSubscribers: any[] = [];

const subscribeTokenRefresh = (cb: any) => {
    _refreshSubscribers.push(cb);
};

const onTokenRefreshed = () => {
    _refreshSubscribers.map((cb) => cb());
};

axiosInstance.defaults.headers.common['Authorization'] =
localStorage.getItem(ACCESS_TOKEN_KEY);
axiosInstance.interceptors.request.use(async (config: InternalAxiosRequestConfig) => {
        const access_token = localStorage.getItem(ACCESS_TOKEN_KEY);
        if (access_token && isAccessTokenExpired(access_token)) {
            if (config.headers) {
                config.headers["Authorization"] = access_token;
            }
        }
        return config;
    },
    (error) => Promise.reject(error)
);

axiosInstance.interceptors.response.use(
    (response) => {
        return response;
    },
    async (error) => {
        if (error.response) {
            const config = error?.config;
            if (error?.response?.status === 401 && !config._retry) {
                if (!_isRefreshing) {
                    config._retry = true;
                    _isRefreshing = true;

                    try {

                        const refresh_token = localStorage.getItem(REFRESH_TOKEN_KEY);
                        if (!refresh_token) {
                            window.location.reload();
                            return Promise.reject(error)
                        }
                        const response = await axios.post(`${baseURL}/auth/refreshToken`, {
```

```
                    refresh_token,
                });

                config.headers.authorization = response?.data?.access_token;
                axiosInstance.defaults.headers.common['Authorization'] =
response?.data?.access_token;
                localStorage.setItem(ACCESS_TOKEN_KEY,
response?.data?.access_token);
                localStorage.setItem(REFRESH_TOKEN_KEY,
response?.data?.refresh_token);
                localStorage.setItem("user", response?.data?.user);

                onTokenRefreshed();
                config._retry = true;

                return await axios.request(config);
            } catch (error: any) {
                if (
                    error?.response?.data?.code === 401 ||
                    error?.response?.data?.code === "401"
                ) {
                    clearUserAllData();
                    return;
                }
                return {};
            } finally {
                _isRefreshing = false;
            }
        } else {
            return new Promise((resolve) => {
                subscribeTokenRefresh(() => {
                    resolve(axios.request(config));
                });
            });
        }
    }
    return Promise.reject(error);
  }
);

export const authProvider: AuthProvider = {
  login: async ({user, access_token, refresh_token}: IData) => {
    if (user) {
      const profileObj = user ? parseJwt(user) : null;
      if (profileObj) {
        localStorage.setItem(
          "user",
          JSON.stringify(user)
        );
```

```
            localStorage.setItem(ACCESS_TOKEN_KEY, access_token)
            localStorage.setItem(REFRESH_TOKEN_KEY, refresh_token)

            return {
                success: true,
            }
        } else {
            return {
                success: false
            }
        }
    }
    return {
        success: false
    };
},
logout: async () => {
    clearUserAllData();
    axios.defaults.headers.common = {};
    return {
        success: true,
    };
},
onError: async (error) => {
    console.log(error?.response?.data)
    return {error: error?.response?.data || error}
},
check: async () => {
    const access_token = localStorage.getItem(ACCESS_TOKEN_KEY);
    const refresh_token = localStorage.getItem(REFRESH_TOKEN_KEY);
    if (!access_token || !refresh_token) {
        return {
            authenticated: false,
            error: new Error("Not authenticated"),
            logout: true,
            success: false,
            // redirectTo: '/login'
        }
    } else if (access_token || refresh_token) {
        return {
            authenticated: true
        }
    }
    return {
        authenticated: false,
        // redirectTo: '/login',
        logout: true
    };

},
```

```
    getPermissions: async () => {
        const userToken = localStorage.getItem('user') as string;
        const user = userToken ? parseJwt(userToken) : null;
        if (user) {
            return user?._doc?.status || user?.status
        }
        return "user";
    },
    getIdentity: async (): Promise<IGetIdentity | any> => {
        const token = localStorage.getItem(ACCESS_TOKEN_KEY) as string;
        const refresh = localStorage.getItem(REFRESH_TOKEN_KEY) as string;
        // const favoritePlaces = JSON.parse(localStorage.getItem(localFavPlacesKey) as
string);
        const user = localStorage.getItem("user") as string;
        if (!token && (refresh && !isAccessTokenExpired(refresh))) {
            return {
                authenticated: false,
                error: new Error("Not authenticated"),
                logout: true,
                redirectTo: '/login'
            };
        }
        const data = user ? parseJwt(user) : null;

        if (!data) return null;

        return {
            user: data?._doc || data as ProfileProps,
            // favoritePlaces: favoritePlaces as string[]
        }
    },
};

import express, {Response, NextFunction} from 'express';
import mongoose from  'mongoose';
import expressFileUpload from  'express-fileupload';
import bodyParser from  'body-parser';
import cors from  "cors";
import dotenv from  "dotenv";
import compression from  "compression";


import {scheduler, reserve} from "./services";

import {
    authRouter,
    userRouter,
    establishmentRouter,
    newsRouter,
    reviewRouter,
```

```
    commentRouter,
    cityRouter,
    menuRouter,
    managerRouter,
    conversationRouter,
    messageRouter,
    caplRouter,
    subscribeNotificationRouter,
    fullTextSearchRouter,
    tagRouter, featureRouter,
    saveRouter
} from './routes';

import {configs} from './configs';
import {CustomError} from "./errors";
import {CustomRequest} from "./interfaces/func";
import notificationRouter from "./routes/notification.router";
dotenv.config({path: `./environments/.env`});

const app = express();


app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: true}));

app.use(function (req, res, next) {
    res.setHeader('Content-Type', 'application/json;charset=UTF-8')
    res.setHeader('Access-Control-Allow-Credentials', '*')
    res.setHeader("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type,
Accept")
    next();
})
app.use(cors(
    {
        origin: [configs.CLIENT_URL, configs.STAT_URL],
    }
));
app.use(compression());

app.use(expressFileUpload({
    useTempFiles: true,
    tempFileDir: '/tmp/'
}));

app.use('/api/v1/ping', (req, res) => res.json('THE SERVER WORKS STABLY'));

app.use('/api/v1/auth', authRouter);
app.use('/api/v1/users', userRouter);
app.use('/api/v1/managers', managerRouter);
app.use('/api/v1/conversation', conversationRouter);
```

```
app.use('/api/v1/message', messageRouter);
app.use('/api/v1/establishment', establishmentRouter);
app.use('/api/v1/news', newsRouter);
app.use('/api/v1/review', reviewRouter);
app.use('/api/v1/comment', commentRouter);
app.use('/api/v1/city', cityRouter);
app.use('/api/v1/menu', menuRouter);
app.use('/api/v1/capl', caplRouter);
app.use('/api/v1/subscribe', subscribeNotificationRouter);
app.use('/api/v1/notification', notificationRouter);
app.use('/api/v1/search', fullTextSearchRouter);
app.use('/api/v1/tag', tagRouter);
app.use('/api/v1/feature', featureRouter);
app.use('/api/v1/save', saveRouter);

app.use('*', (req, res) => {
    res.status(404).json('Route not found');
});

app.use((err: any, _: any, res: Response, __: NextFunction) => {
    console.log(err)
    res
        ?.status(err?.status || 500)
        ?.send({
            error: err?.message || 'Unknown Error',
            code: err?.status || 500
        });
});

mongoose.Promise = Promise;
mongoose.connect(configs.DB_URI as string).then(async () => {
    console.log("|----------------------------------------")
    console.log('| Connect: success')
    let db = mongoose.connection.db;
    app.listen(Number(configs.PORT)!, configs.HOST!, () => {
        console.log(`| Started on port http://localhost:${configs.PORT}`);
        console.log("|________________________________________________")
    });
}).catch(err => {
    console.log(err)
    console.log('connect: error')
})
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ПЛАТФОРМА ДЛЯ ОНЛАЙН-РЕЗЕРВУВАННЯ МІСЦЬ У ЗАКЛАДАХ ХАРЧУВАННЯ

Виконав: Якобчук Роман Валерійович

Керівник: доцент кафедри ПЗКС, к.т.н., Катін Павло Юрійович

Київ – 2024

ПОСТАНОВКА ЗАДАЧІ

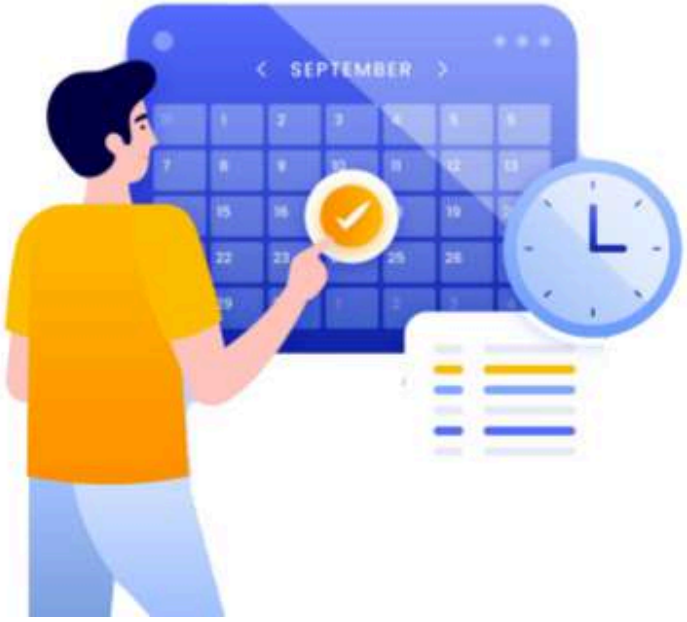
Мета проекту: створити зручну та функціональну онлайн-платформу, яка упростить процес бронювання місць у ресторанах, кафе та інших закладах харчування для користувачів, а також забезпечить ефективне управління бронюваннями для власників закладів.

Завдання:

1. Провести аналіз існуючих рішень
2. Порівняти аналоги з розроблюваною платформою
3. Створити перелік вимог до платформи
4. Обрати технологій розробки
5. Програмно реалізувати платформу
6. Провести тестування готової платформи

АКТУАЛЬНІСТЬ

- За даними досліджень, кількість онлайн-користувачів зростає на 10% щорічно.
- З розвитком технологій зростає попит на зручні сервіси онлайн-резервування місць.
- Клієнти цінують можливість швидко та зручно зарезервувати місця.



АНАЛОГІЧНІ РІШЕННЯ

Reservble

Reston

Restoran

ФУНКЦІОНАЛЬНІСТЬ РОЗРОБЛЮВАНОЇ ПЛАТФОРМИ

1. Авторизація користувачів.
2. Перегляд, створення, редагування, видалення закладів.
3. Перегляд, створення, редагування, видалення новин.
4. Підтримка багатомовності.
5. Пошук за різними фільтрами.
6. Перегляд закладів на карті.
7. Адаптивний користувацький інтерфейс.
8. Система сповіщень.
9. Створення, підтвердження, відхилення нових резервувань.

ОБРАНІ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБЛЕННЯ

React

TypeScript

Socket.IO

Node.js

Express.js

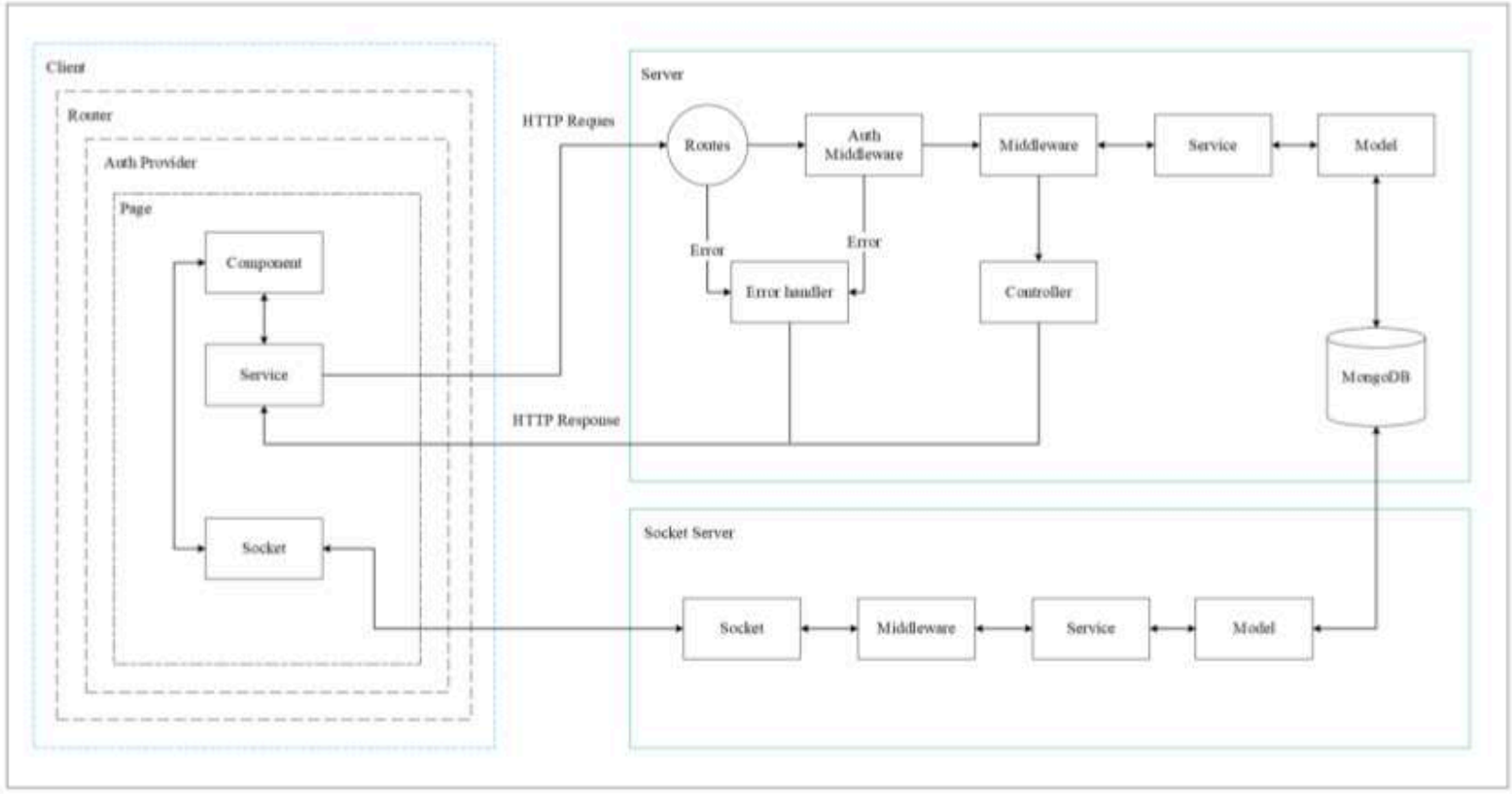
MongoDB

Cloudinary

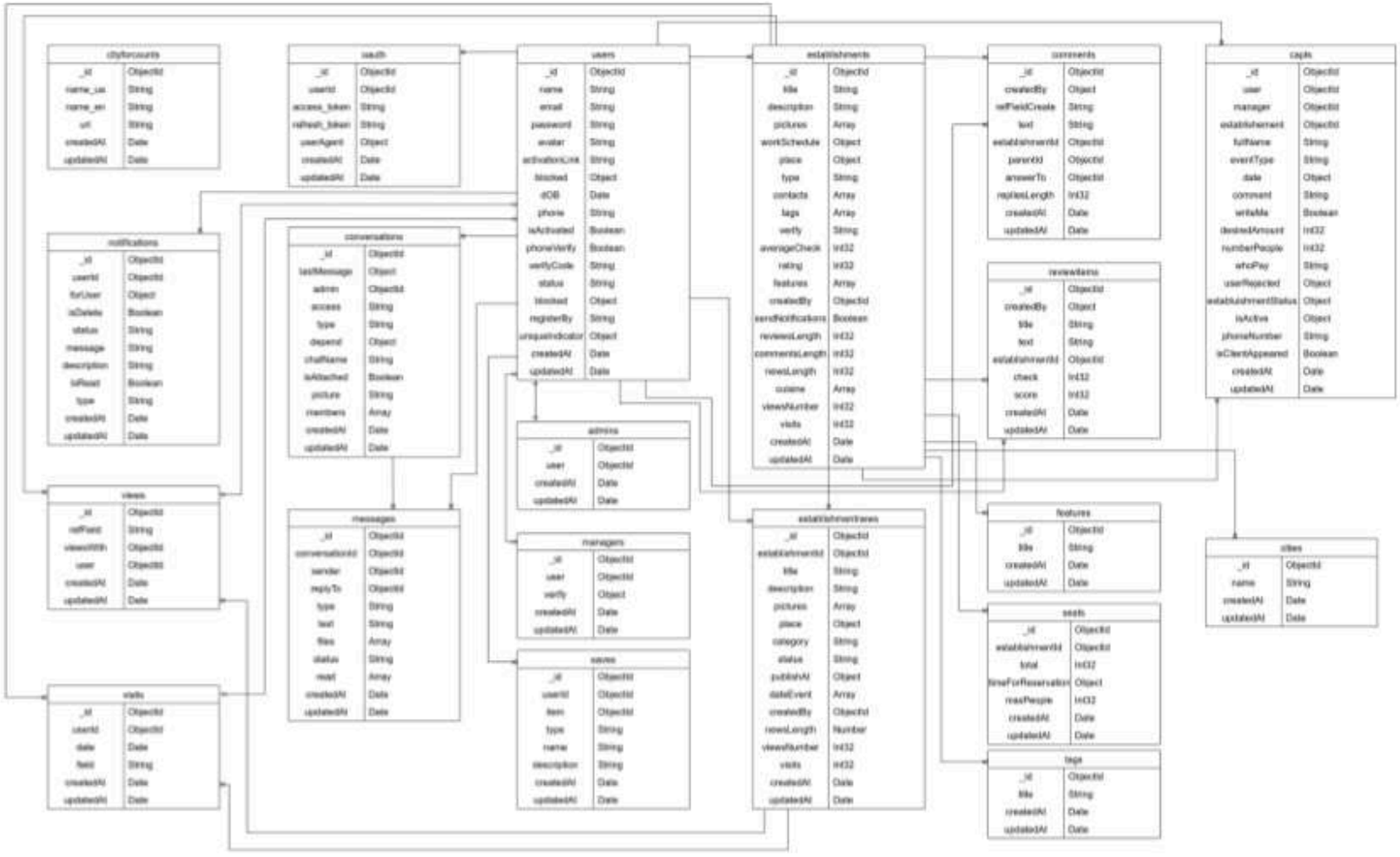
Docker

Nginx

АРХІТЕКТУРА



СТРУКТУРА БАЗИ ДАНИХ

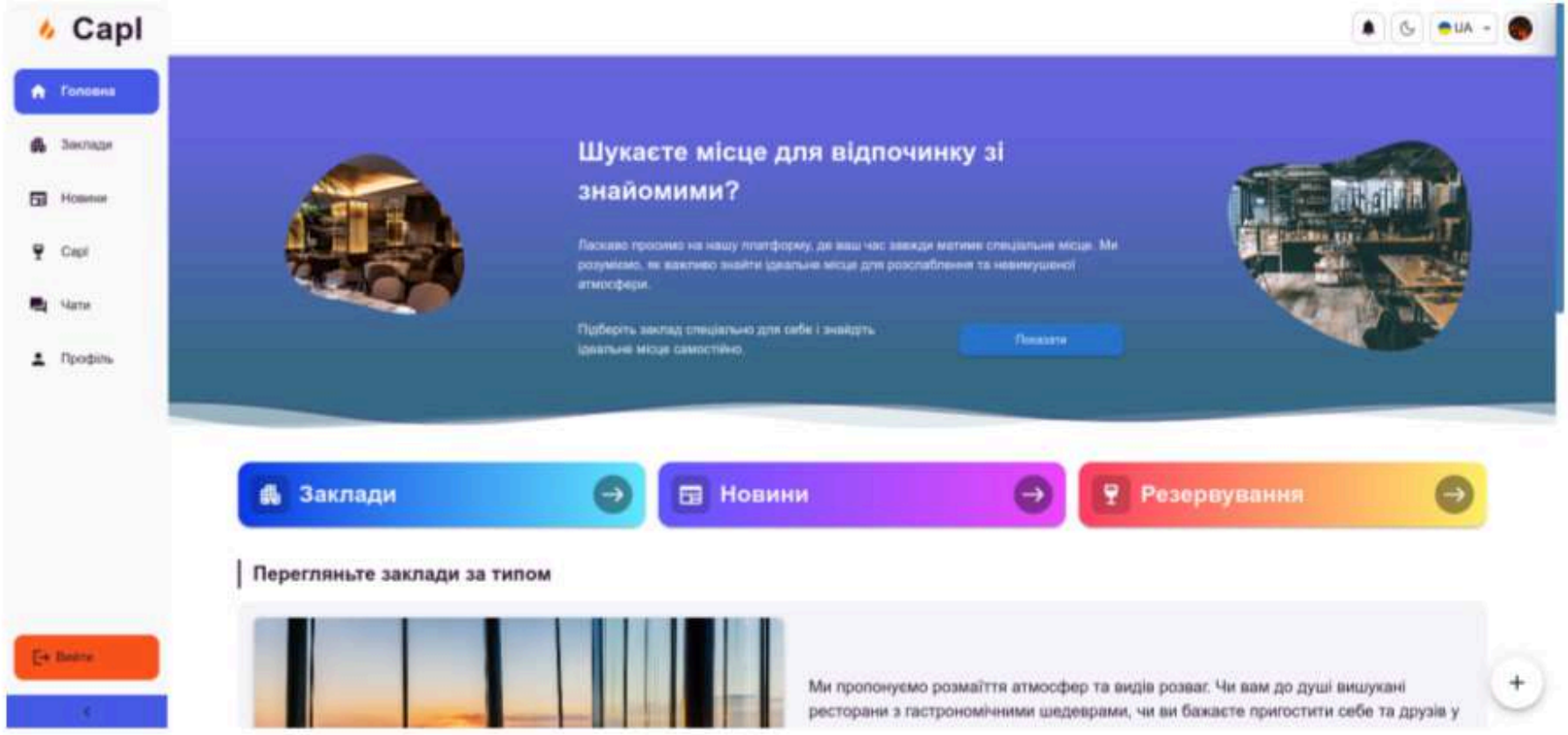


ПОРІВНЯННЯ З АНАЛОГАМИ

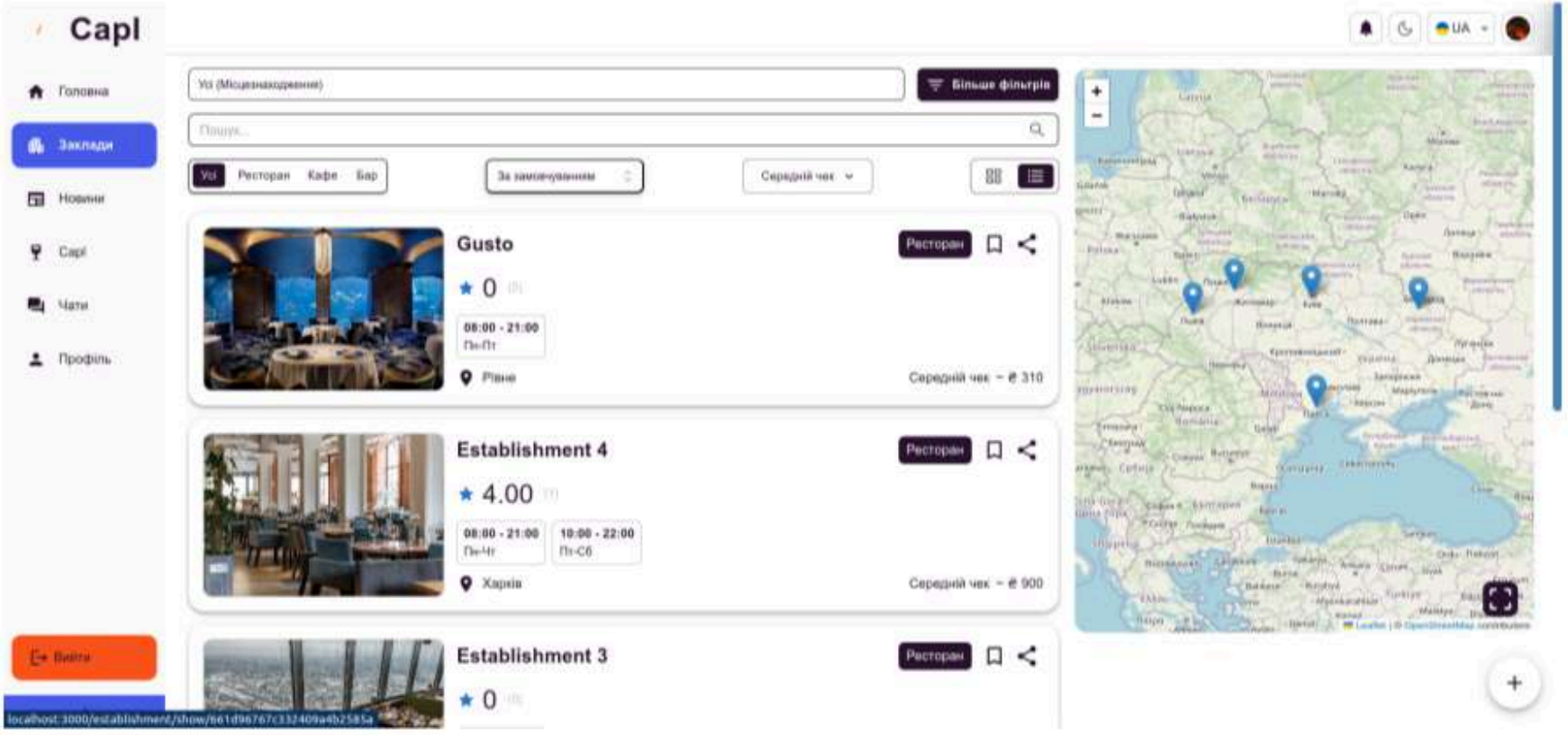
Під час порівняння платформи з конкурентними рішеннями було виявлено, що платформа має такі переваги:

1. Інтуїтивно зрозумілий інтерфейс.
2. Географічне охоплення.
3. Зручність пошуку та фільтрування.
4. Можливості онлайн-бронювання.
5. Система відгуків та рейтингів.
6. Картографічна візуалізація.
7. Багатомовний інтерфейс.
8. Пошук закладів поблизу.
9. Інтерактивна комунікація з закладами.

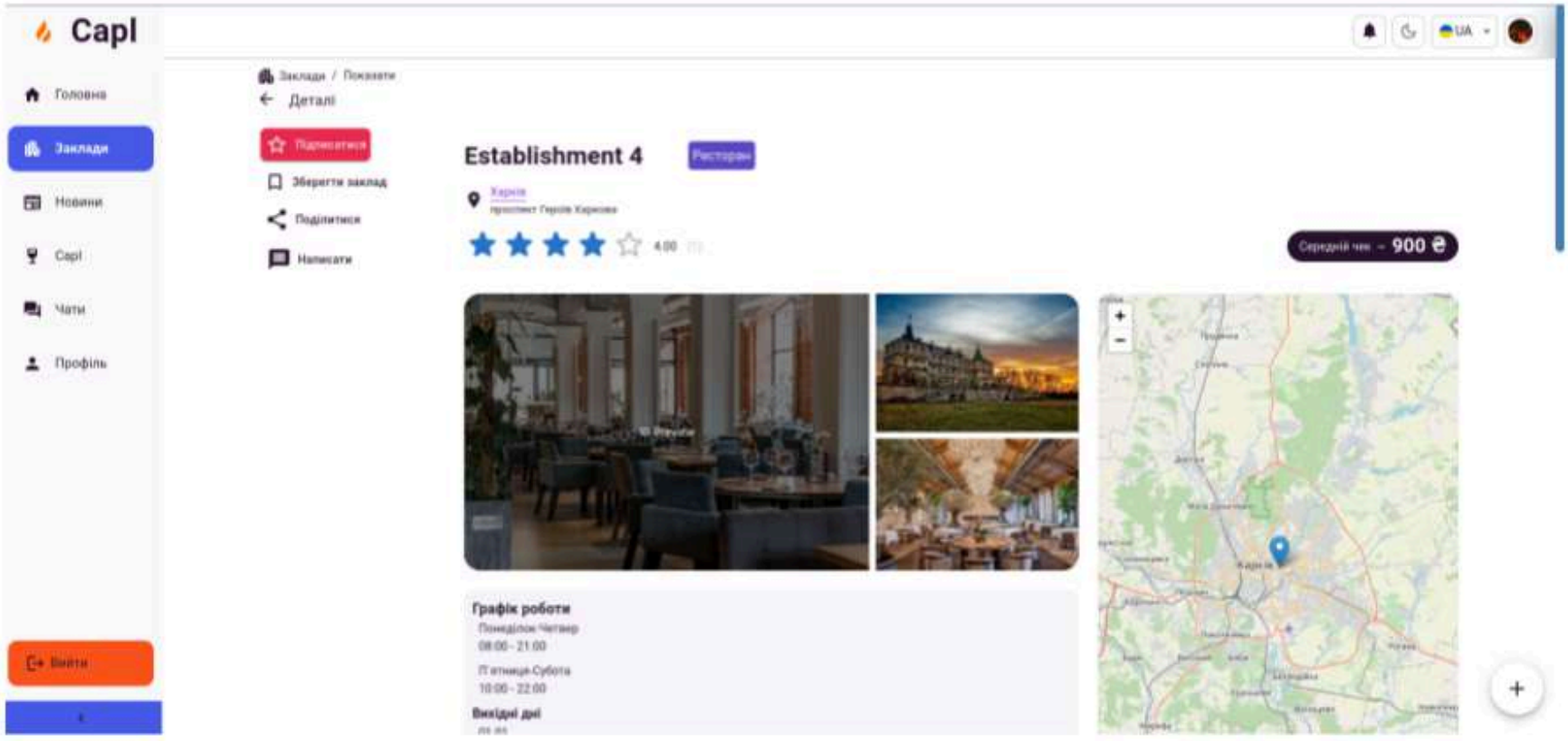
ІНТЕРФЕЙС КОРИСТУВАЧА. ГОЛОВНА СТОРІНКА



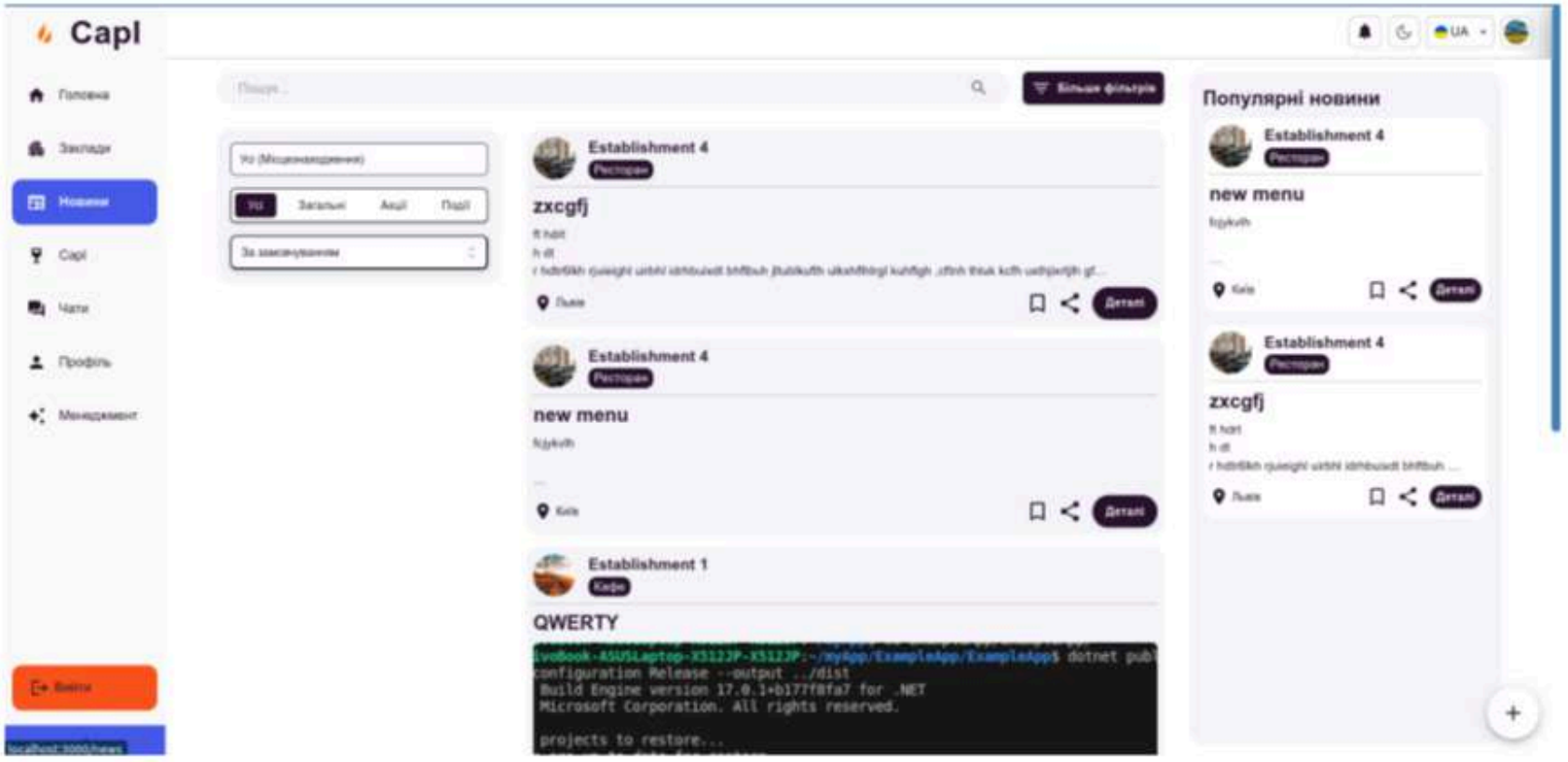
ІНТЕРФЕЙС КОРИСТУВАЧА. СТОРІНКА ЗАКЛАДІВ



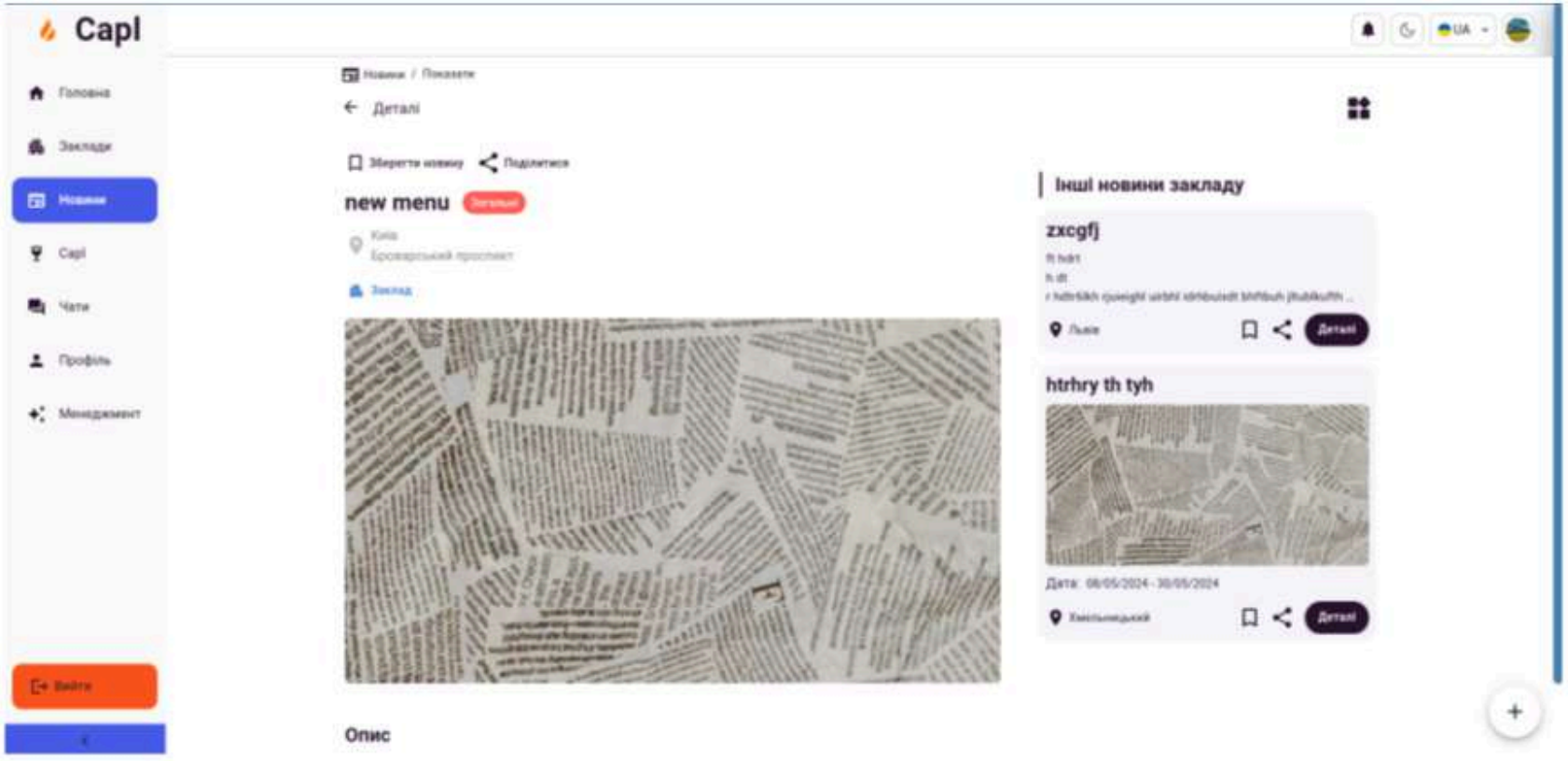
ІНТЕРФЕЙС КОРИСТУВАЧА. ПЕРСОНАЛЬНА СТОРІНКА ЗАКЛАДУ



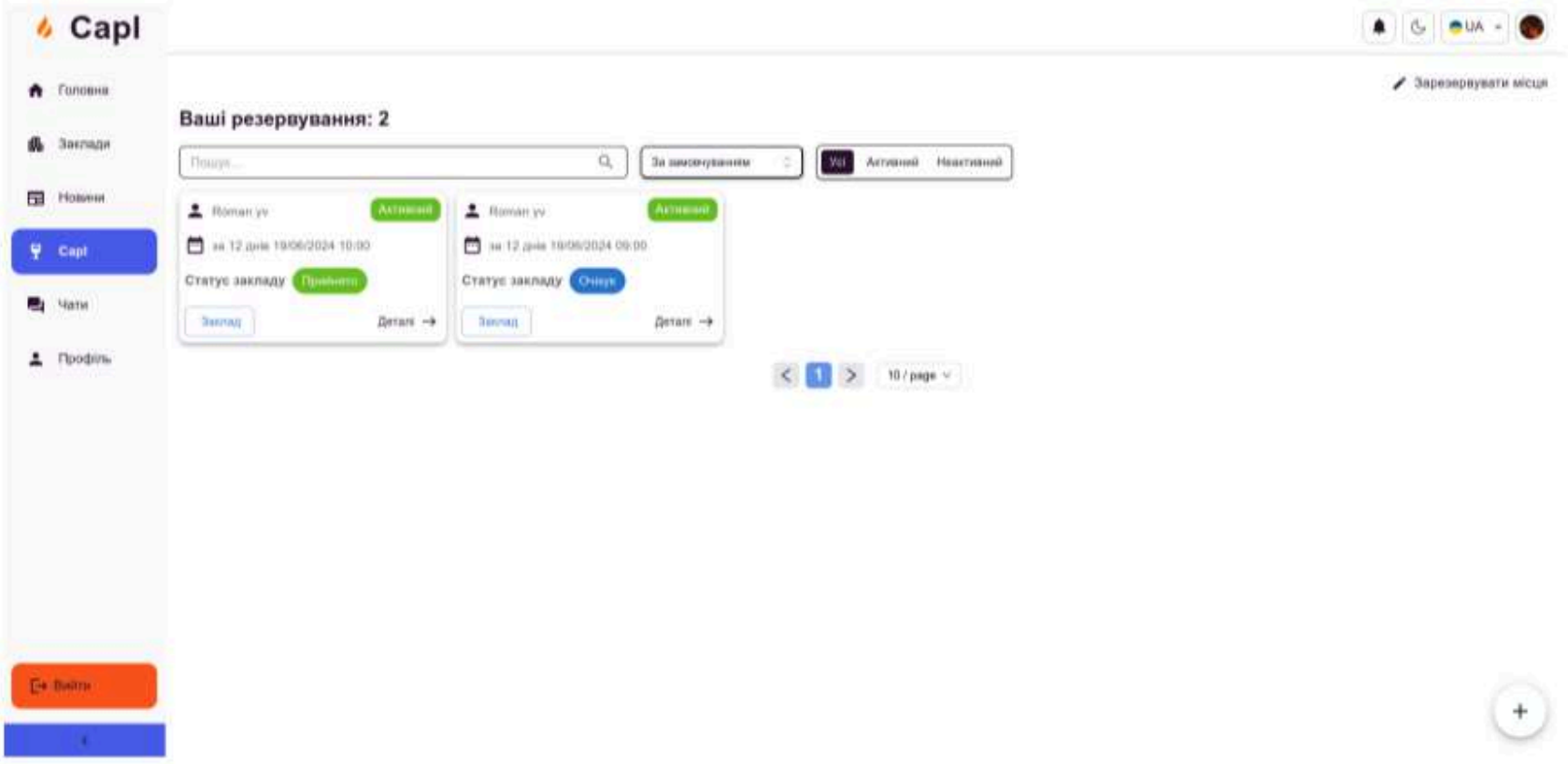
ІНТЕРФЕЙС КОРИСТУВАЧА. СТОРІНКА НОВИН



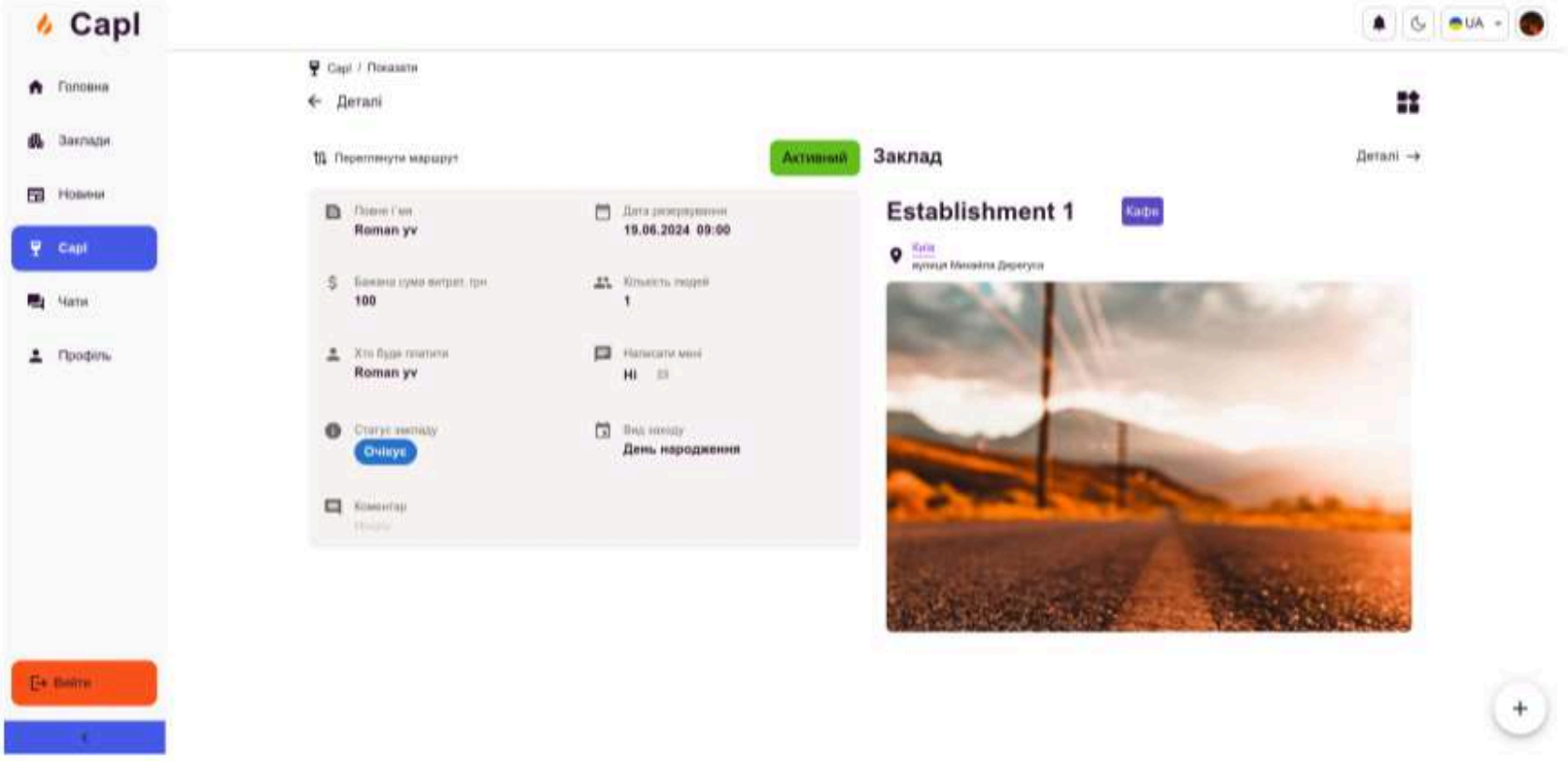
ІНТЕРФЕЙС КОРИСТУВАЧА. ПЕРСОНАЛЬНА СТОРІНКА НОВИН



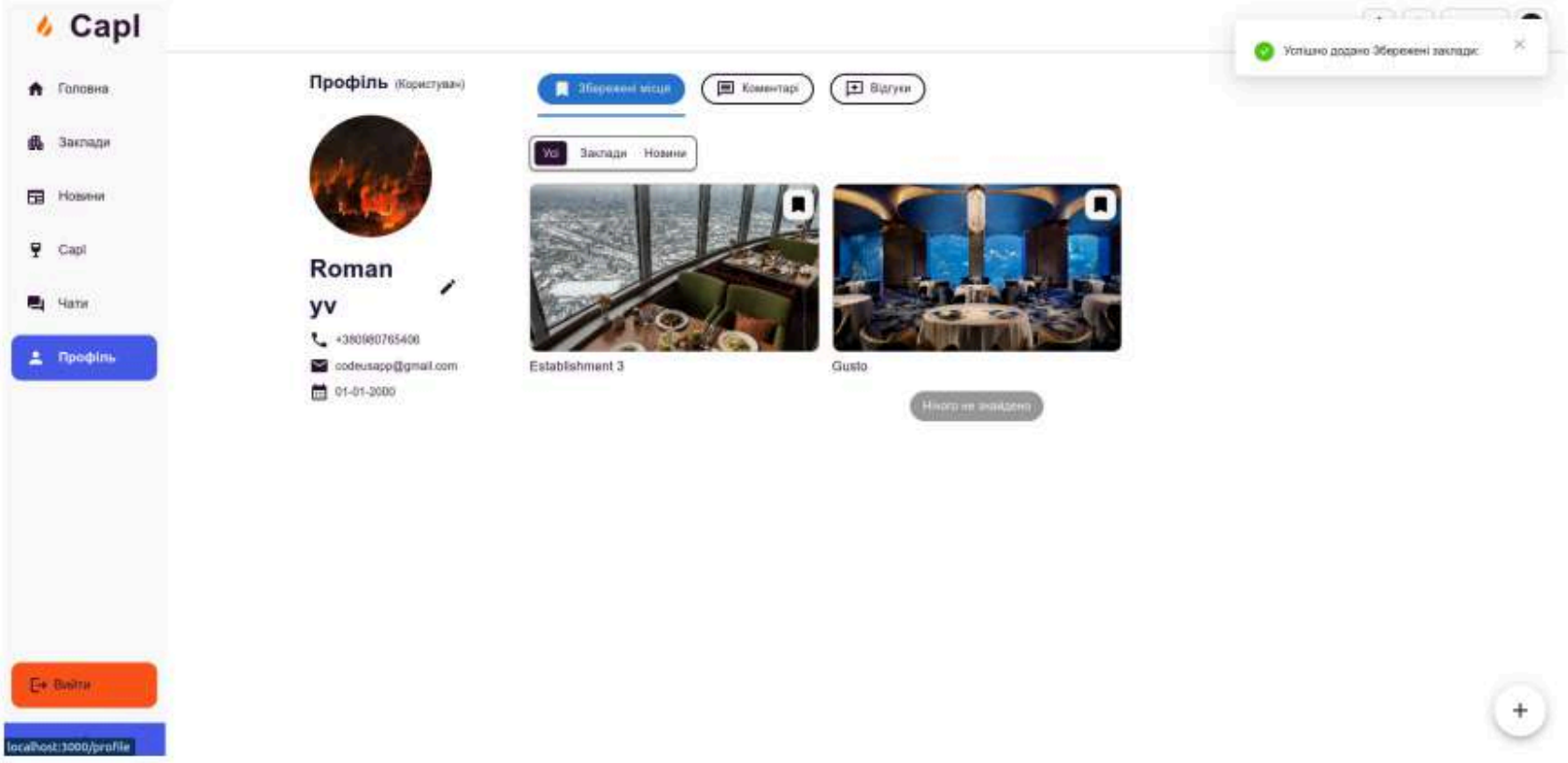
ІНТЕРФЕЙС КОРИСТУВАЧА. СТОРІНКА РЕЗЕРВУВАНЬ



ІНТЕРФЕЙС КОРИСТУВАЧА. ДЕТАЛЬНА ІНФОРМАЦІЯ РЕЗЕРВУВАННЯ



ІНТЕРФЕЙС КОРИСТУВАЧА. ПРОФІЛЬ КОРИСТУВАЧА



РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Працездатність платформи було перевірено шляхом:

- тестування інтерфейсу;
- тестування у різних браузерах;
- тестування на відповідність функціональним вимогам;
- тестування введення даних.

ШЛЯХИ ПОДАЛЬШОГО РОЗВИТКУ

Історія переглядів

Меню

Персоналізовані рекомендації

Вдосконалення користувацького інтерфейсу

ВИСНОВКИ

1. Проведено аналіз існуючих рішень, наведено їх переваги та недоліки.
2. Проведено порівняння засобів розробки.
3. Описано загальну структуру та архітектуру платформи.
4. Спроєктовано базу даних платформи.
5. Розроблено платформу відповідно до переліку вимог та вибраних технологій.
6. Проведено тестування платформи та виправлено помилки.
7. Сформовано рекомендації щодо подальшого розвитку платформи.

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2023 р.

ПЛАТФОРМА ДЛЯ ОНЛАЙН-РЕЗЕРВУВАННЯ МІСЦЬ У ЗАКЛАДАХ ХАРЧУВАННЯ

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Павло КАТІН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Роман ЯКОБЧУК

2023

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Платформа для онлайн-резервування місць у закладах харчування, який являє собою веб-додаток, серверну частину якого створено за допомогою технології Node.js з мовою програмування TypeScript, а клієнтську – бібліотекою React з використанням Typescript.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) функціональна працездатність елементів сторінок платформи;
2) наявність доступу до бази даних закладів харчування;
3) отримання сповіщень та електронних листів про наближення події резервування місць;
4) забезпечення належного рівня безпеки даних;
5) зручність роботи з платформою;
6) відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Black Box Testing. Перевіряється безпосередньо програмний продукт на відповідність функціональним вимогам, не заглиблюючись у внутрішню структуру коду. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1) функціональне тестування, зокрема на рівні Use Case Testing (тестування на основі використання);
2) тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);
3) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність платформи перевіряється шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування платформи в різних веб-браузерах;
5) тестування при максимальному навантаженні;
6) тестування стабільності роботи при різних умовах;
7) тестування зручності використання;
8) тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Завідувач кафедри

_____ Євгенія СУЛЕМА

"___" _____ 2024 р.

ПЛАТФОРМА ДЛЯ ОНЛАЙН-РЕЗЕРВУВАННЯ МІСЦЬ У ЗАКЛАДАХ ХАРЧУВАННЯ

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проєкту:

_____ Павло КАТІН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Роман ЯКОБЧУК

2024

ЗМІСТ

1. Опис структури платформи

Платформа для онлайн-резервування місць у закладах харчування складається із однієї статичної веб-сторінки та веб-сторінок, вміст яких формується динамічно. Платформа є двомовною, тому всі статичні веб-сторінки продубльовані українською та англійською мовами. Вміст динамічних сторінок залежить від мови, вибраної користувачем.

Однією статичною сторінкою є “Вітаємо”, на яку користувач потрапляє в першу чергу коли відвідує сайт.

Динамічна частина платформи включає наступні сторінки:

1. Сторінки аутентифікації та авторизації.
2. Головна сторінка після авторизації.
3. Сторінка закладів.
4. Сторінка новин.
5. Сторінка резервувань.
6. Персональні сторінки закладів.
7. Персональні сторінки новин.
8. Персональні сторінки резервувань.
9. Сторінка профілю користувача.

Платформа реалізована у вигляді багатьох блоків. Головними блоками є “шапка” сайту, бокова панель та “тіло” сайту платформи, де динамічно змінюються сторінки. Шапка містить такі елементи як, посилання на сторінку зі сповіщеннями, кнопку зміни світлої та темної теми платформи, кнопку зміни мови платформи для статичного тексту, а також посилання на профіль. Посилання на сторінки сповіщень та профілю доступні лише авторизованим користувач. Бічна панель доступна тільки авторизованим користувачам, вона містить основні посилання на сторінки платформи. Також для авторизованих користувачів, на кожній сторінці є кнопка “+”, яка розташована у правій нижній частині сайту. При наведені або натисненні кнопки відкривається меню з кнопок, яке розділене за ролями користувачів. Для менеджерів та адміністраторів доступні такі

кнопки: “Зарезервувати місце”, “Додати новину”, “Додати заклад”. Для звичайного користувача доступна тільки кнопка зарезервувати.

2. Опис вмісту статичних веб-сторінок

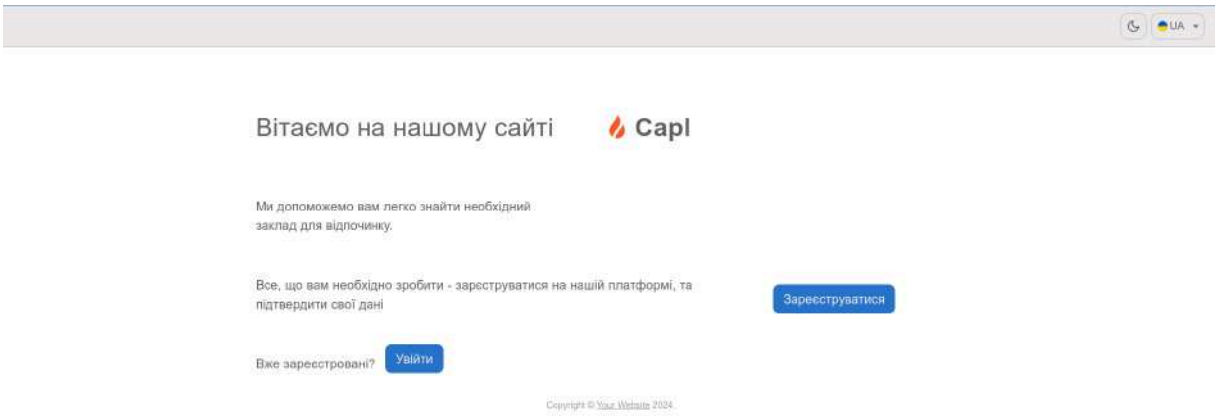


Рис. 1. Сторінка “Вітаємо”

Сторінка “Вітаємо” платформи (рис. 1) містить посилання для неавторизованих користувачів можливість увійти у власний профіль або зареєструватися у разі його відсутності.

3. Процедура авторизації користувача

Для доступу до платформи необхідно попередньо пройти процедуру реєстрації. Для цього на сторінці “Вітаємо” необхідно натиснути “Зареєструватися”. Після натиснення користувача перенаправить на сторінку реєстрації (рис. 2), де необхідно ввести номер телефону, вибрати дату народження, вибрати роль “Користувач” або “Менеджер”, також користувач повинен ввести ім’я, електронний адрес та пароль, для захисту власного профілю від несанкціонованого доступу. Після заповнення усіх полів, необхідно натиснути на “Зареєструватися”. У разі успішної реєстрації, користувач повинен отримати на електронний адрес лист для

підтвердження своїх даних. Лист містить посилання, при кліку на яке, користувача перенаправляє на сторінку входу.

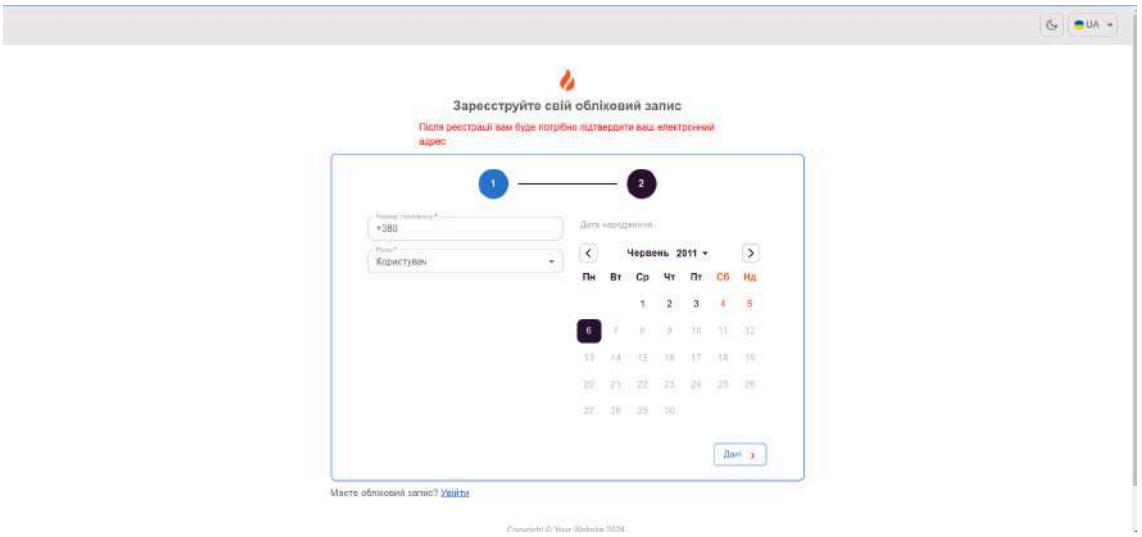


Рис. 2 Сторінка реєстрації

На сторінці входу (рис. 3) користувач повинен ввести електронний адрес та пароль, які він попередньо вводив на сторінці реєстрації. Без підтвердження електронної адреси користувач не матиме доступу до платформи. Після заповнення полів на сторінці входу, необхідно натиснути “Увійти”, в разі коректності введених даних, користувач автоматично перейде на головну сторінку сайту.

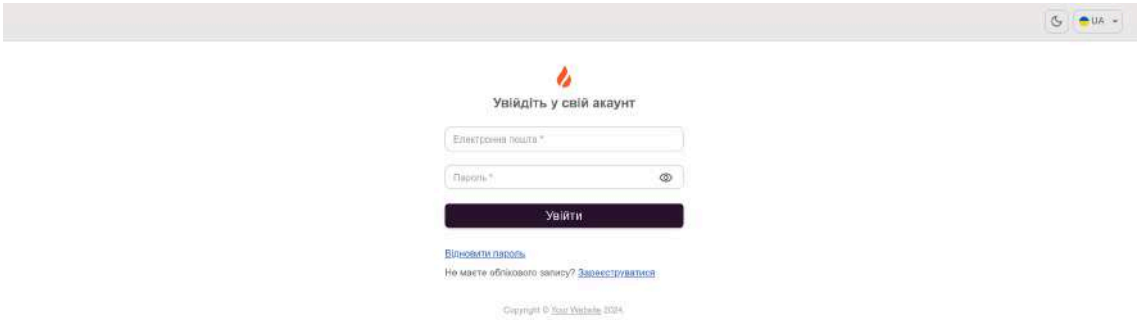


Рис. 3. Сторінка автентифікації

4. Головна сторінка

Головна сторінка складається з п’яти окремих блоків.

Перший блок (рис. 4.1) – вітання користувачів на даній платформі. Даний блок містить зображення закладів, заголовок, опис, а також посилання на сторінку закладів.

Другий блок (рис. 4.1) – містить посилання на головні сторінки платформи, які є важливими для користувача, при кліку на які, користувача буде перенаправлено та такі сторінки як “Заклади”, “Новини” та “Резервування”.

Третій блок (рис. 4.2) – містить посилання на сторінку закладів за типом та показує інформацію, скільки закладів за даним типом є на платформі.

Четвертий блок (рис. 4.3) – містить посилання на сторінку закладів за обласним центром України та показує інформацію, скільки закладів є в даному місті.

П’ятий блок (рис. 4.3) – показує популярні заклади за відвідуваннями, при кліку на які користувача перенаправляє на певну сторінку закладу.

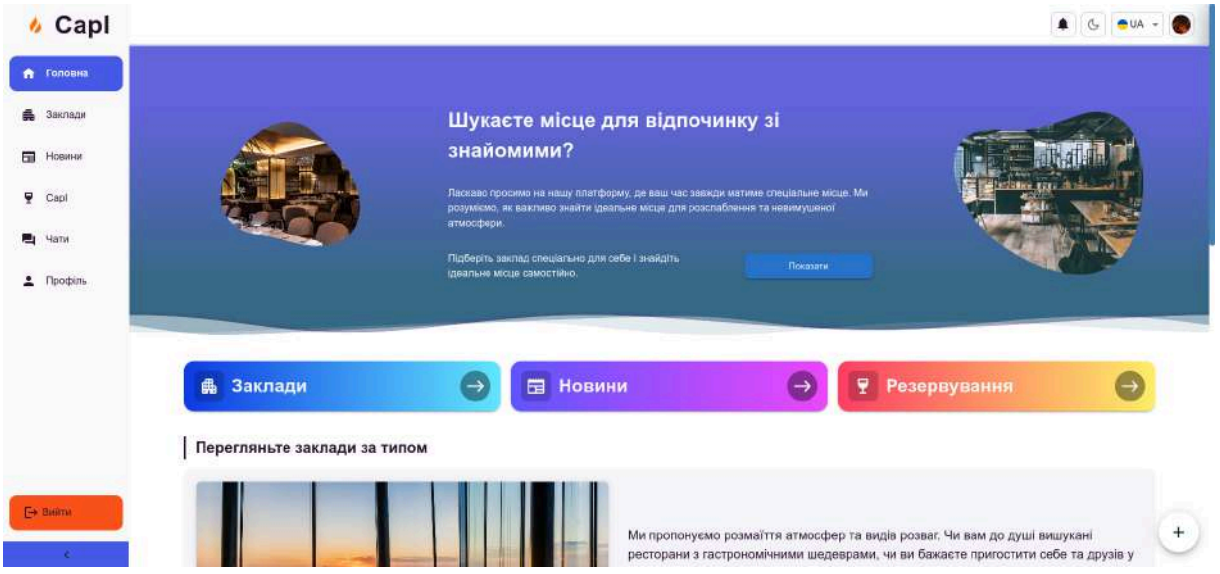


Рис. 4.1. Головна сторінка

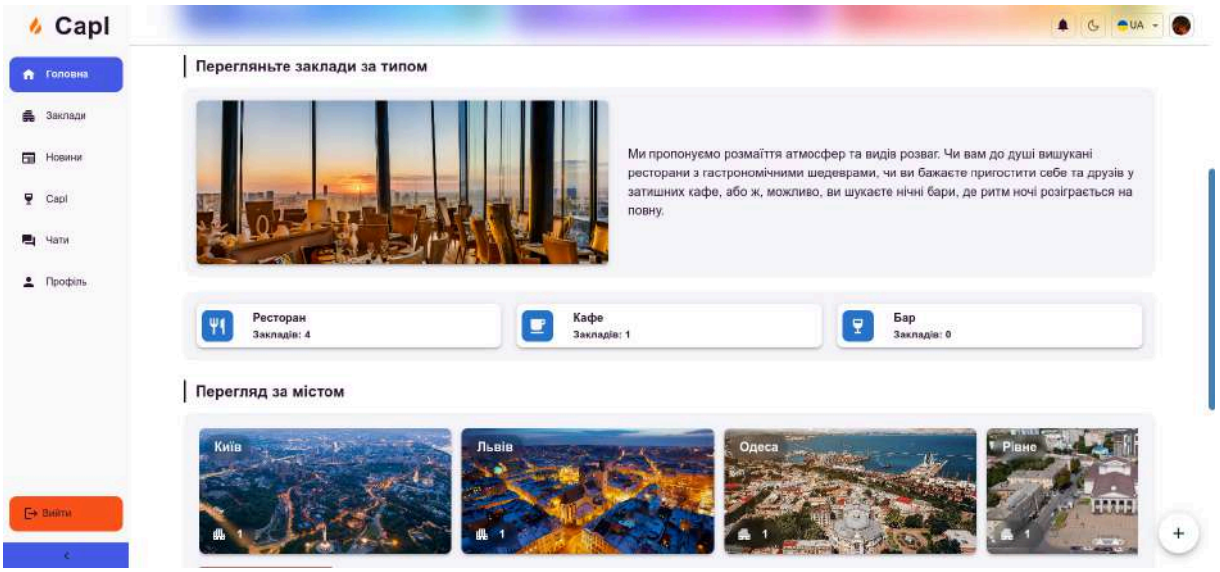


Рис. 4.2. Головна сторінка

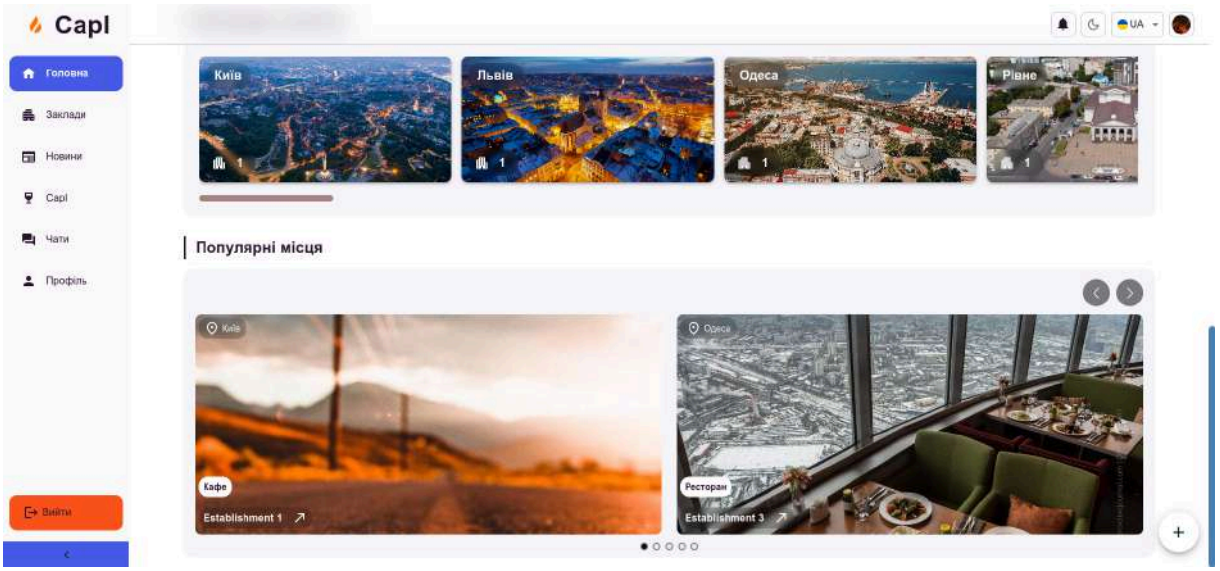


Рис. 4.3. Головна сторінка

5. Сторінка перегляду закладів

Сторінка перегляду закладів (рис. 5) складається з трьох основних блоків:

1. Фільтри, для пошуку закладів за різними категоріями.
2. Карта з маркерами закладів, що відображаються у списку.
3. Список закладів вибраних за фільтрами.

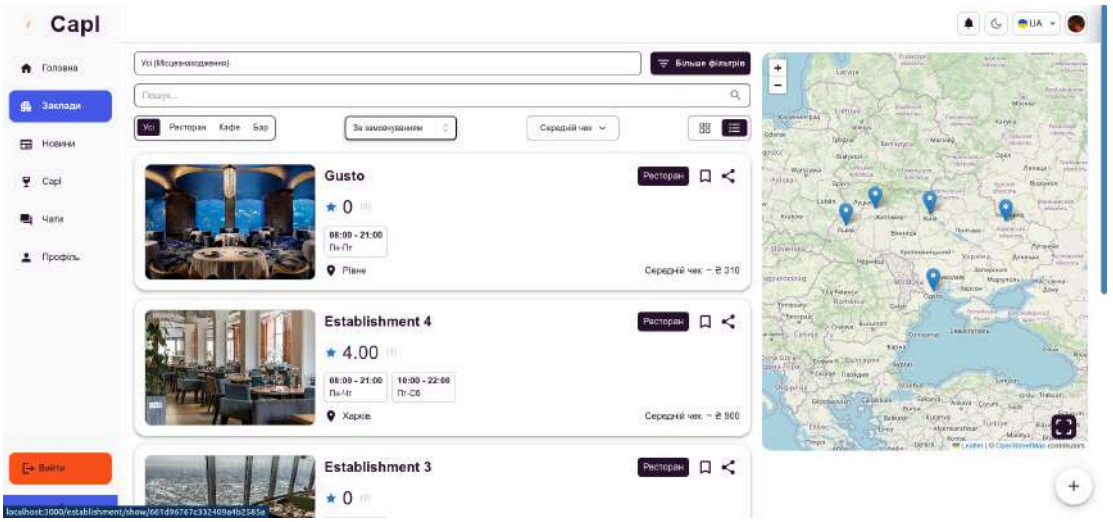


Рис. 5. Сторінка закладів

Сторінка поділена на дві частини, перша – відображає фільтри пошуку та список закладів, друга – відображає карту із закладами.

Основними фільтрами на сторінці є:

1. Вибір міста, де розташований заклад.
2. Кнопка “Більше фільтрів”, яка показує додаткові фільтри.
3. Поле пошуку, в якому можна ввести текст для пошуку закладів за такими полями: назва, опис, теги, особливості та адресою.
4. Поле вибору типу закладу: усі, ресторан, кафе, бар.
5. Поле вибору метода сортування закладів.
6. Поле вибору діапазону для пошуку закладів за середнім чеком.
7. Кнопка перемикання методу відображення закладів, списком чи блоками.

При відкриті додаткових фільтрів, користувач може вибрати місцезнаходження та радіус, для пошуку закладів поруч. Для цього необхідно натиснути кнопку “Додаткові фільтри”, після чого відкриється модальне вікно (рис. 6), де користувач повинен на карті вибрати місцезнаходження, де він хоче шукати заклади, та ввести радіус пошуку в метрах. Після чого необхідно натиснути кнопку “Зберегти”, потім натиснути кнопку “Пошук”, після чого фільтри буде застосовано.

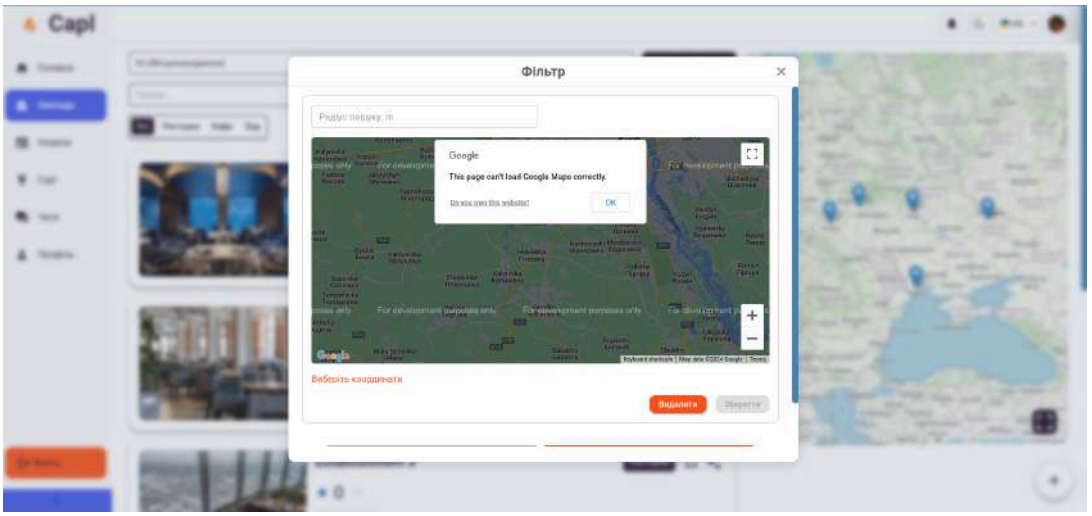


Рис. 6. Додаткові фільтри

На карті відображенні маркери за координатами закладів, а також кнопка для перемикання в повноекранний режим (рис. 7), для зручного перегляду. При натисканні на маркер сторінка закладів пролистує і показує заклад який відповідає за даний маркер. При кліку на кнопку повноекранного режиму, поверх сторінки закладів відкривається вікно з картою. В такому режимі, користувач також має змогу переглядати заклади та застосовувати фільтри пошуку.

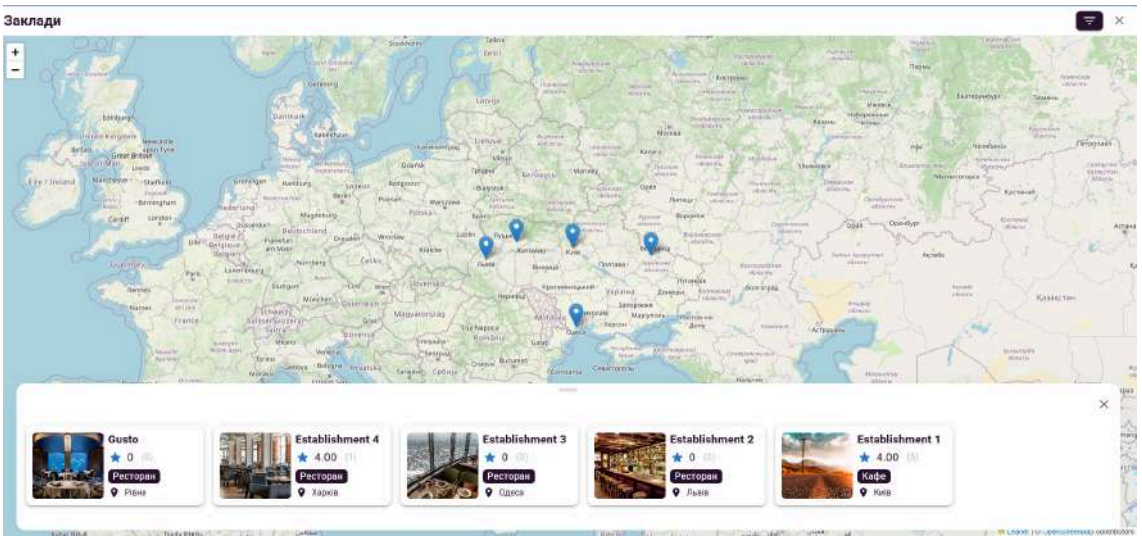


Рис. 7. Повноекранний режим карти закладів

У списку закладів, розташовані картки (рис. 8) даних закладів, при кліку на які користувача перенаправляє на сторінку вибраного закладу. Також для зручності користувачів, у правому верхньому куті картки розташовані кнопки додати у збереженні та поділитися.

Рис. 8. Картка закладу

Після натиснення кнопки “Зберегти” відкривається вікно (рис. 9), де користувач додатково може ввести назву та опис, для зручного пошуку закладів у збережених. Після чого необхідно натиснути кнопку “Зберегти” і заклад буде додано у збережені. Таким самим методом заклад можна видалити зі збережених.

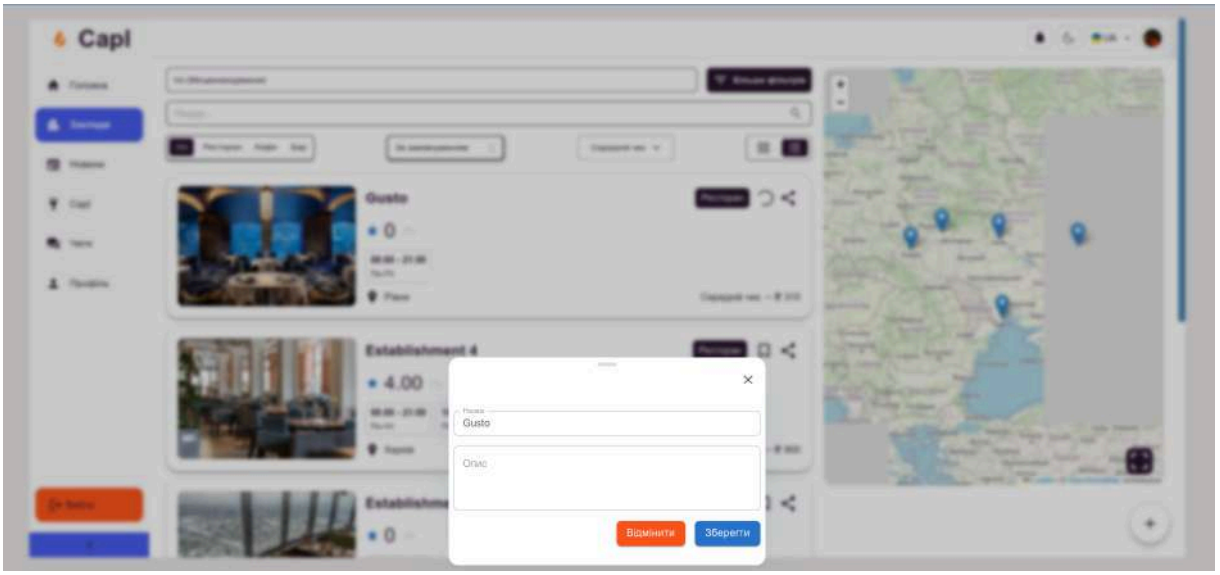


Рис. 9. Вікно додавання закладу у збереженні

Для того, щоб поділитися інформацією про заклад, необхідно спочатку натиснути кнопку “Поділитися”, після чого відкриється модальне вікно (рис. 10). У даному вікні можна скопіювати посилання на сторінку з інформацією про заклад або надіслати посилання у доступні соціальні мережі.

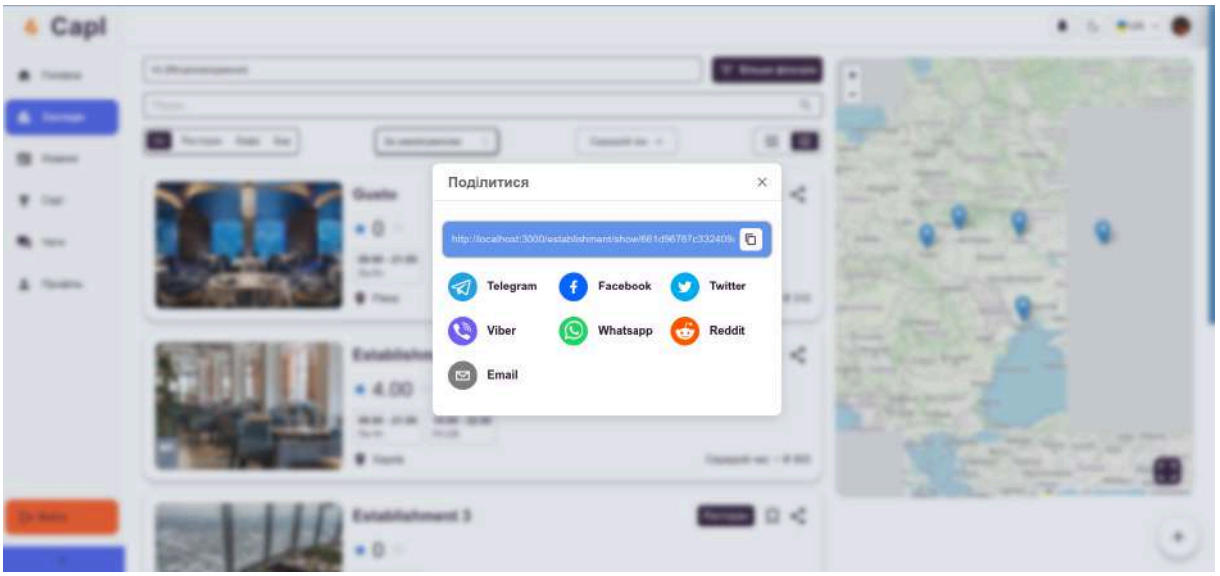


Рис. 10. Вікно “Поділитися” закладом

6. Персональна сторінка закладу

Щоб потрапити на сторінку певного закладу, необхідно натиснути на картку відповідного закладу на сторінці закладів, після чого користувача буде перенаправлено на персональну сторінку закладу з детальною інформацією (рис. 11-12). Зліва можна побачити список з кнопок: “Підписатися”, “Зберегти заклад”, “Поділитися”, “Написати”.

Кнопка “Підписатися” доступна тільки для звичайних користувачів або, якщо ви не є менеджером даного закладу. При кліку на кнопку користувач підписується на сповіщення від закладу, коли буде опубліковано новини. Кнопки “Зберегти заклад” та “Поділитися” виконують такі ж функції як і на картці закладу. Кнопка “Написати” відкриває чат (рис. 13) із закладом для спілкування в реальному часі.

На персональній сторінці закладу користувач може переглянути детальну інформацію про заклад, а саме:

- назва;
- тип закладу;
- місце розташування;
- рейтинг;
- зображення закладу;
- графік роботи;
- вихідні дні;
- опис;
- особливості;
- контакти.

Також на персональній сторінці закладу є кнопка “Зарезервувати місця”, при натисканні на яку, користувача перенаправляє на сторінку створення резервування з даним закладом. Для зручності користувач може переглянути список схожих закладів та той, на сторінці якого перебуває.

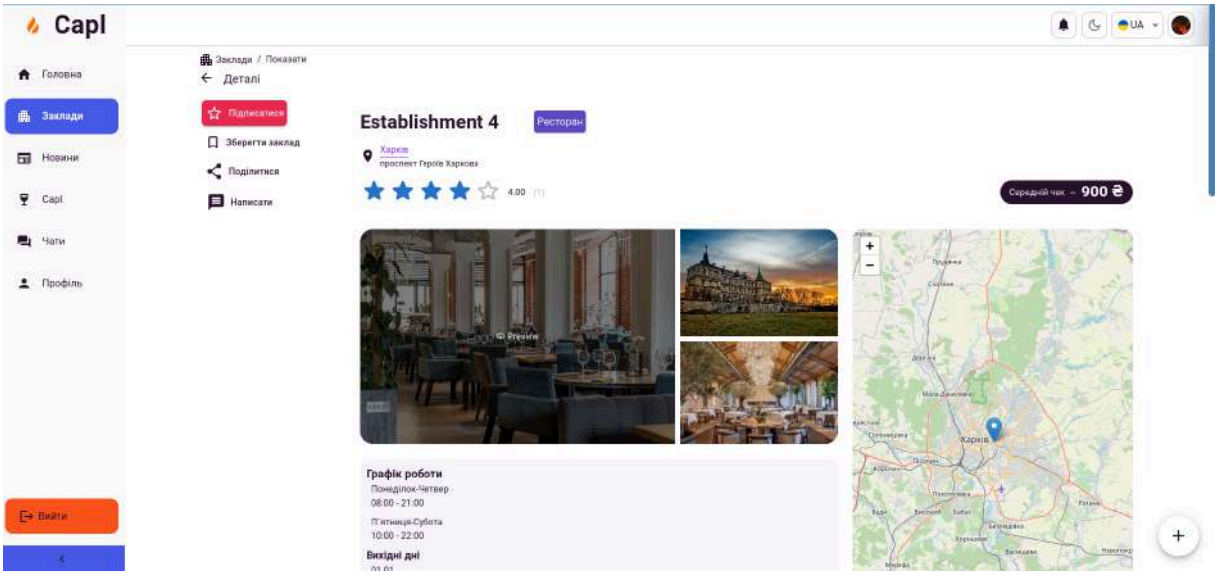


Рис. 11. Персональна сторінка закладу

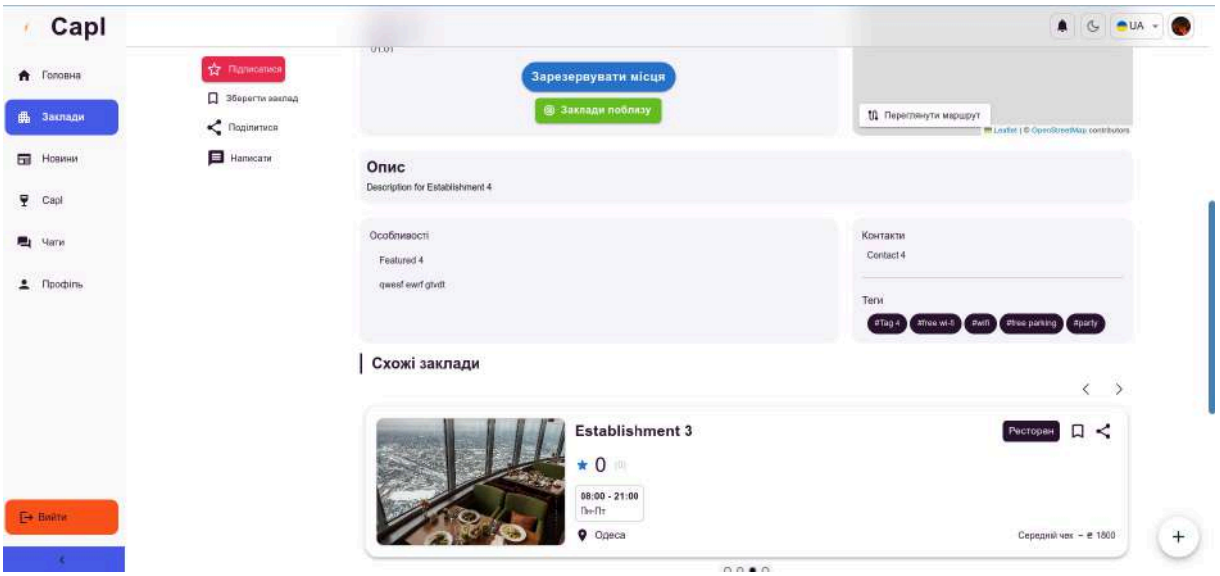


Рис. 12. Персональна сторінка закладу

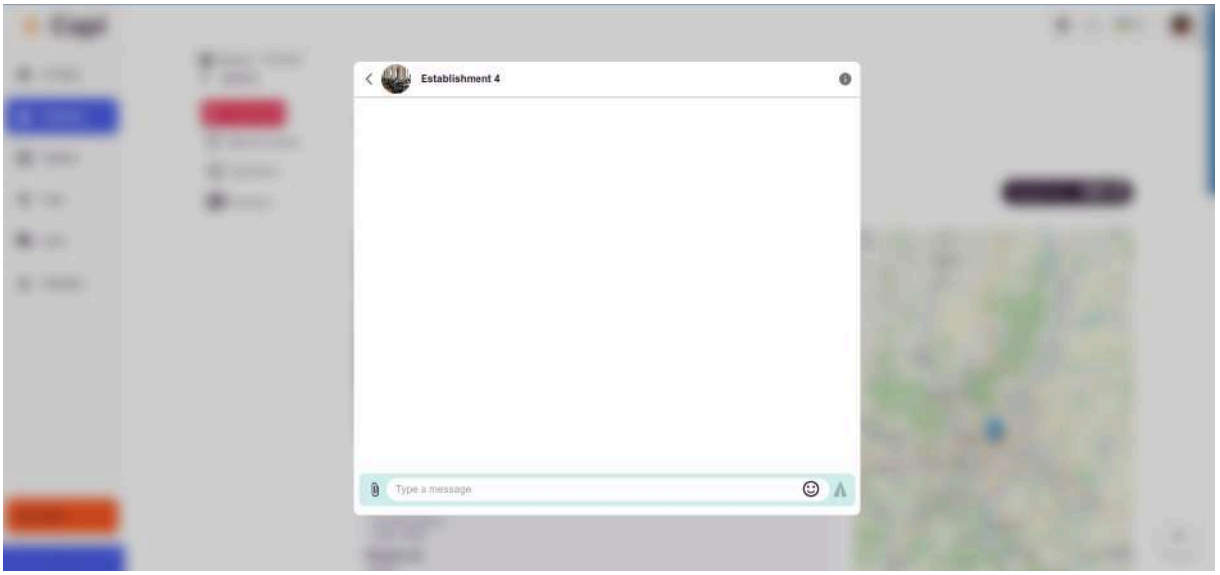


Рис. 13. Персональний чат із закладом

Нижче знаходиться блок з наступними відділами, які динамічно змінюються: “Відгуки”, “Новини” та “Коментарі”.

Відділ з новинами (рис. 14) показує опубліковані новини закладу, при натисканні на кнопку “Деталі”, яка знаходиться у правому нижньому куті картки, користувача перенаправить на персональну сторінку новини.

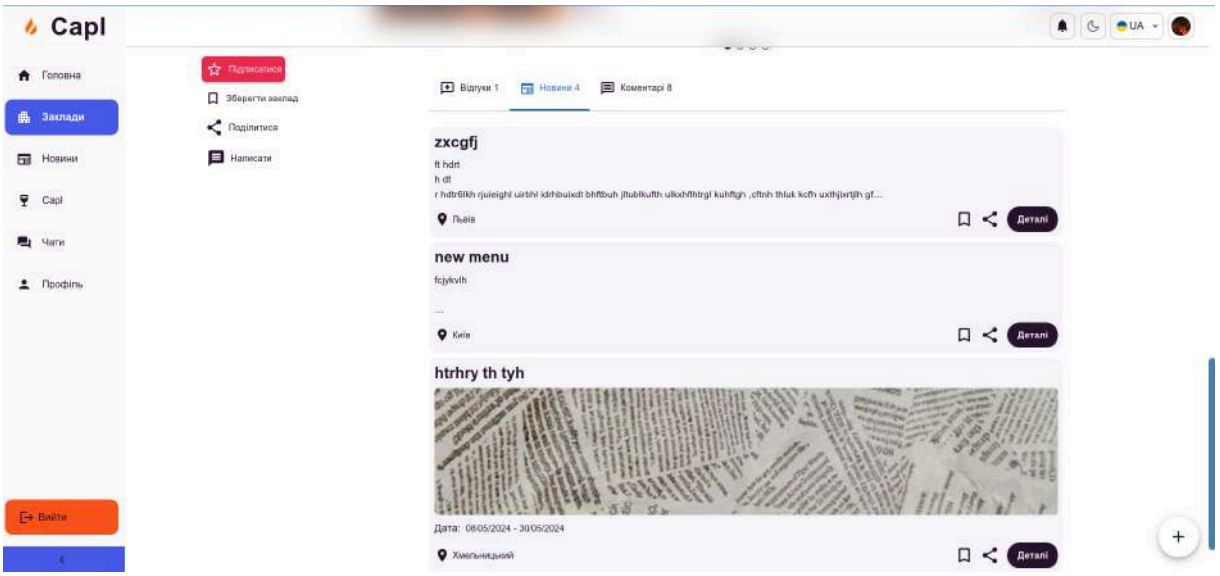


Рис. 14. Персональна сторінка закладу. Новини

Відділ відгуків (рис. 15) містить відгуки, які користувачі залишили відвідавши даний заклад. Щоб залишити відгук, необхідно вибрати оцінку від 1 до 5, далі ввести суму чеку, заголовок, а також опис до відгуку. Після чого натиснути кнопку “Залишити відгук” і відгук буде опубліковано. Важливо, користувач може залишити лише один відгук для закладу протягом доби.

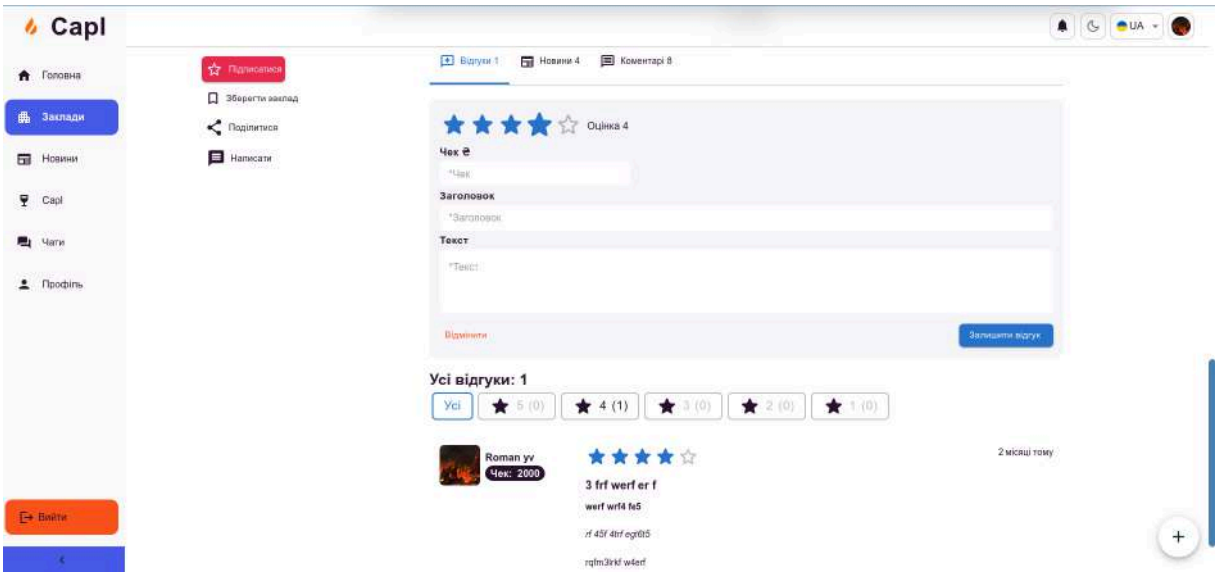


Рис. 15. Персональна сторінка закладу. Відгуки

Відділ коментарі (рис. 16) містить поле вводу нового коментаря, коментарі та питання, які користувачі можуть поставити закладу. Також можна відповідати на коментарі інших користувачів. Таким чином користувачі групуються у зручний вигляд для читання. На картці коментаря відображається наступне: фото профіля користувача, що залишив відгук, ім’я користувача, дата публікації коментаря, текст. Знизу картки є кнопки, щоб відповісти на коментар або переглянути відповіді на даний коментар. Справа зверху знаходиться меню, в якому є кнопка “Видалити”, якщо цей коментар належить користувачеві, він має змогу видалити його.

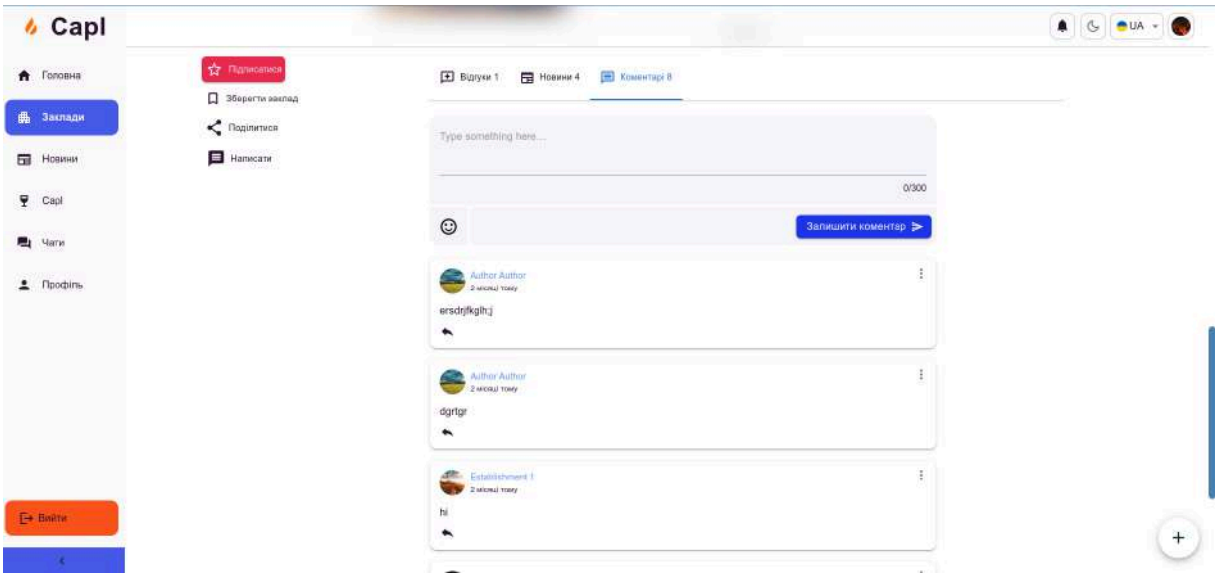


Рис. 16. Персональна сторінка закладу. Коментарі

7. Сторінка керування закладом

Щоб потрапити на сторінку керуванням інформації про заклад, необхідно мати роль “Менеджер”. Для цього необхідно перейти на персональну сторінку закладу (рис. 17), у правому верхньому куті буде відображатися кнопка меню, яка не доступна звичайному користувачеві. При натисканні на неї відкривається меню, де відображається кнопка “Редагувати”, при кліку на кнопку, менеджера перенаправляє на сторінку редагування закладу (рис. 18). Важливо, що менеджер може редагувати інформацію лише про власний заклад. На даній сторінці можна редагувати

всю інформацію про заклад, яка відображається на персональні сторінці закладу. Сторінка додавання містить ідентичний інтерфейс, що і сторінка редагування, і відрізняється лише кнопками “Зберегти” на сторінці редагування, та “Створити” на сторінці додавання нового закладу.

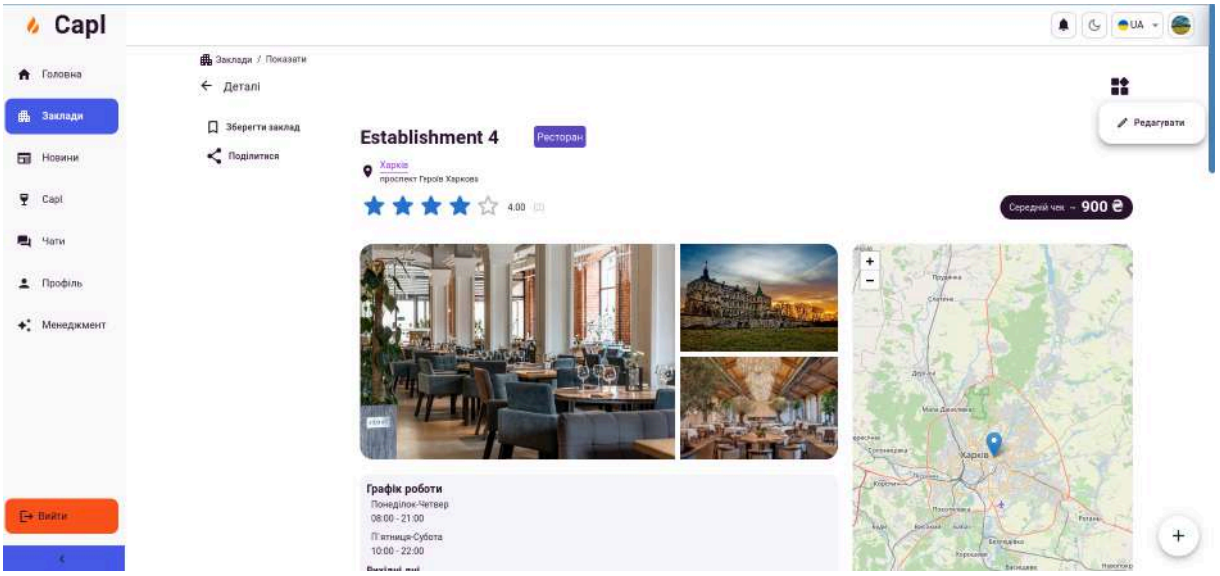


Рис. 17. Меню для редагування інформації про заклад

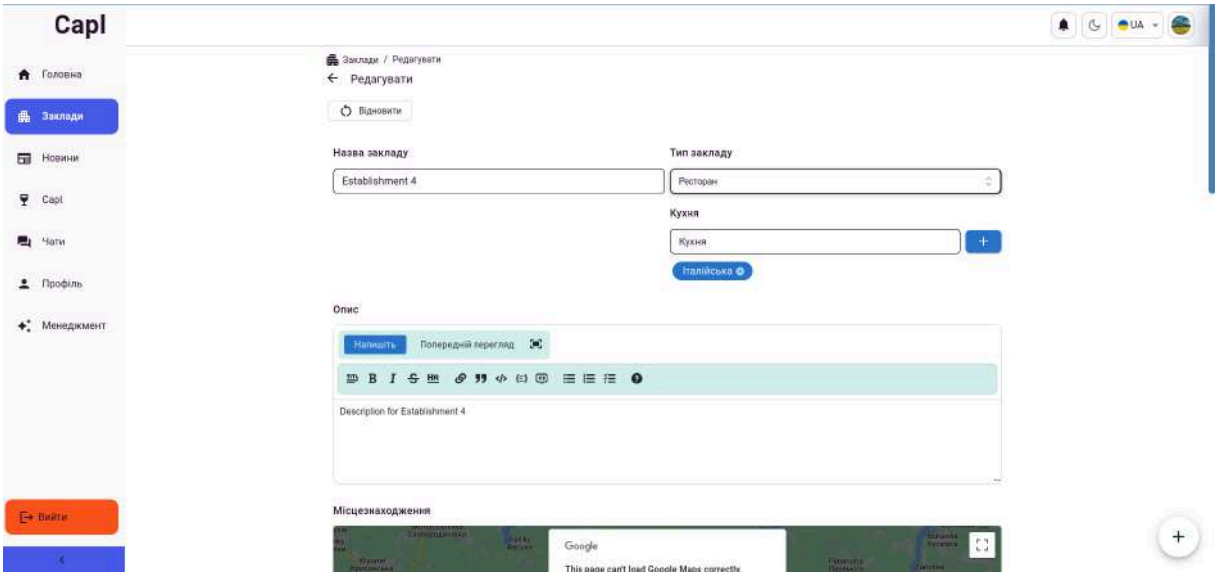


Рис. 18. Сторінка редагування закладу

8. Сторінка перегляду новин

Сторінка перегляду новин (рис. 19) складається з трьох основних блоків:

1. Фільтри, для пошуку новин за різними категоріями.
2. Список новин вибраних за фільтрами.
3. Список популярних новин за переглядами.

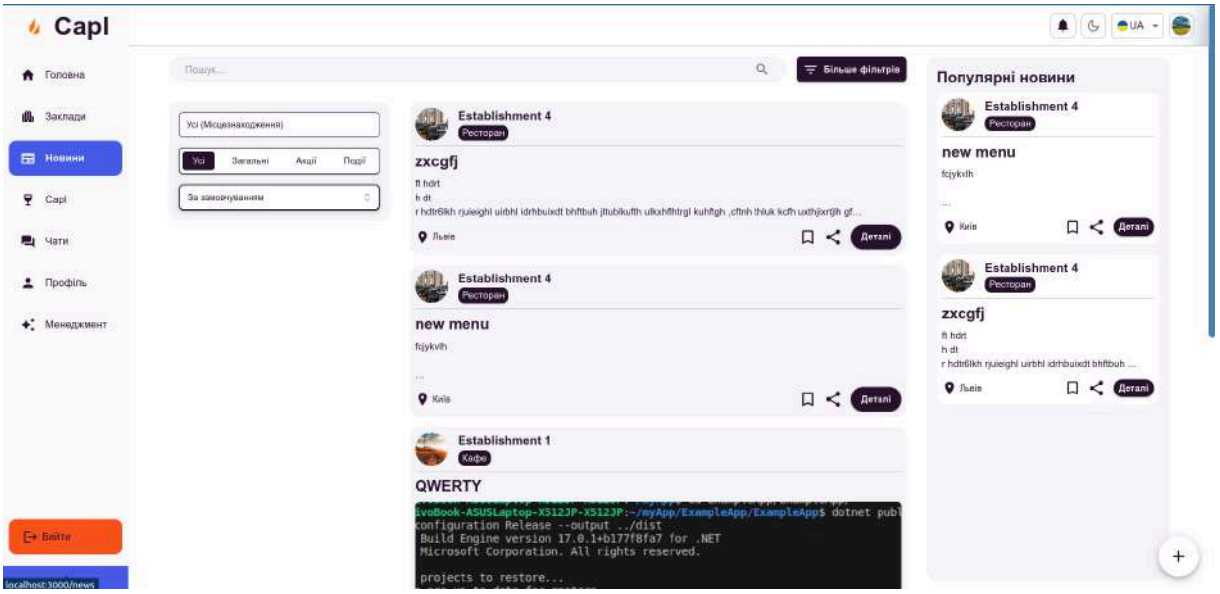


Рис. 19. Сторінка перегляду новин

Як і сторінка закладів, сторінка новин поділена на дві основні частини, перша – відображає фільтри пошуку та список новин, друга – відображає популярні новини.

Основними фільтрами на сторінці є:

1. Вибір міста, де розташований заклад.
2. Поле пошуку, в якому можна ввести текст для пошуку новин за такими полями: назва, опис.
3. Поле вибору типу закладу: загальні, акції, події.
4. Поле вибору метода сортування новин.

Список закладів складається з новин, на яких крім основної інформації про новину, також відображається мінімальна інформація про

заклад, а саме назва, зображення та тип закладу. При натисненні на блок з інформацією про заклад, користувача буде перенаправлено на персональну сторінку закладу.

У правому нижньому куті картки новин знаходиться кнопка “Деталі”, при натисканні на яку, користувача перенаправить на персональну сторінку новин.

9. Персональна сторінка новини

Щоб потрапити на сторінку певної новини, необхідно натиснути на кнопку “Деталі” на картці відповідної новини на сторінці новин, після чого користувача буде перенаправлено на персональну сторінку новини з детальною інформацією (рис. 20).

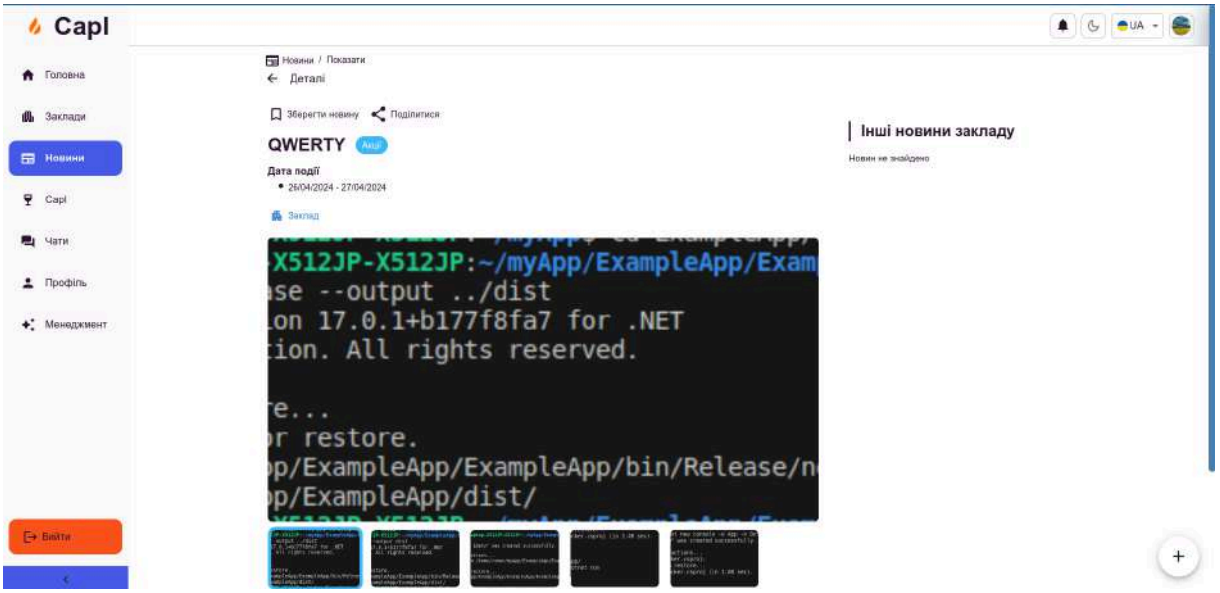


Рис. 20. Персональна сторінка новини

На персональній сторінці новини відображається наступна інформація:

- назва;
- опис;
- дата проведення події, за умови, що така існує;

- місце проведення події, за умови що така існує;
- зображення, за умови, що такі є;
- кнопка, для перенаправлення на сторінку закладу, якому належить новина.

Справа від інформації про новину, знаходиться список інших новин, що належать тому ж закладу, що і поточна новина (рис. 21).

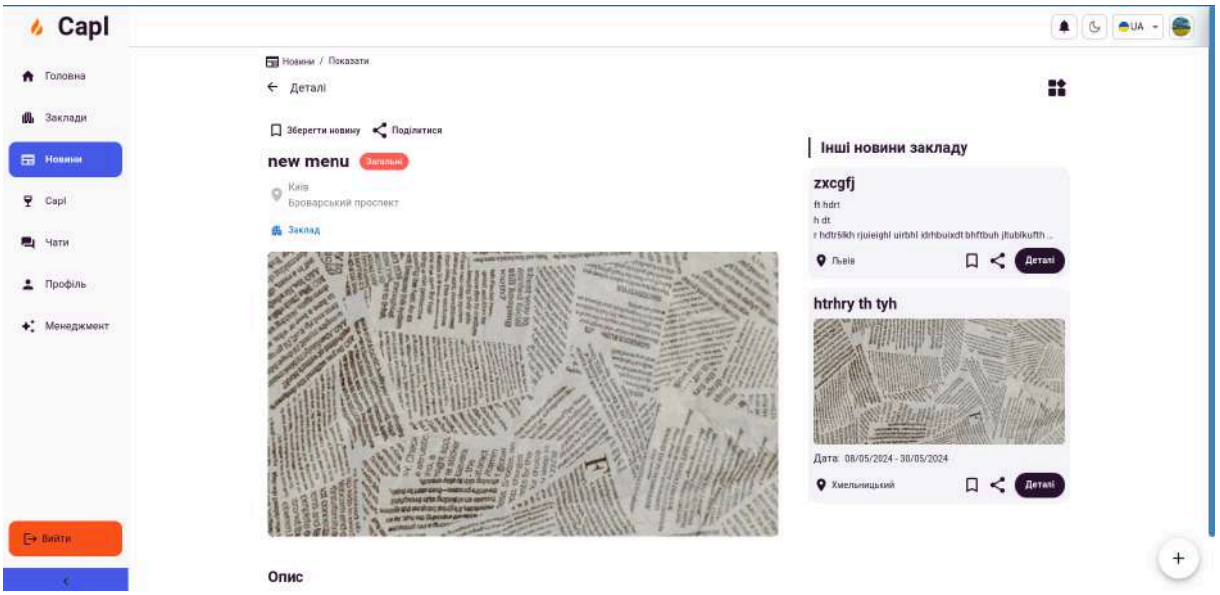


Рис. 21. Список інших новин закладу

10. Сторінка керування новиною

Щоб потрапити на сторінку керуванням інформації про новину, необхідно мати роль “Менеджер”. Для цього необхідно перейти на персональну сторінку новини (рис. 22), у правому верхньому куті буде відображатися кнопка меню, яка не доступна звичайному користувачеві. При натисканні на неї відкривається меню, де відображається кнопка “Редагувати”, при кліку на кнопку, менеджера перенаправляє на сторінку редагування новини (рис. 23). Важливо, що менеджер може редагувати інформацію лише про власні новини власних закладів. На даній сторінці можна редагувати всю інформацію про новину, яка відображається на персональні сторінці закладу. Сторінка додавання містить ідентичний

інтерфейс, що і сторінка редагування, і відрізняється лише кнопками “Зберегти” на сторінці редагування, та “Створити” на сторінці додавання нового закладу.

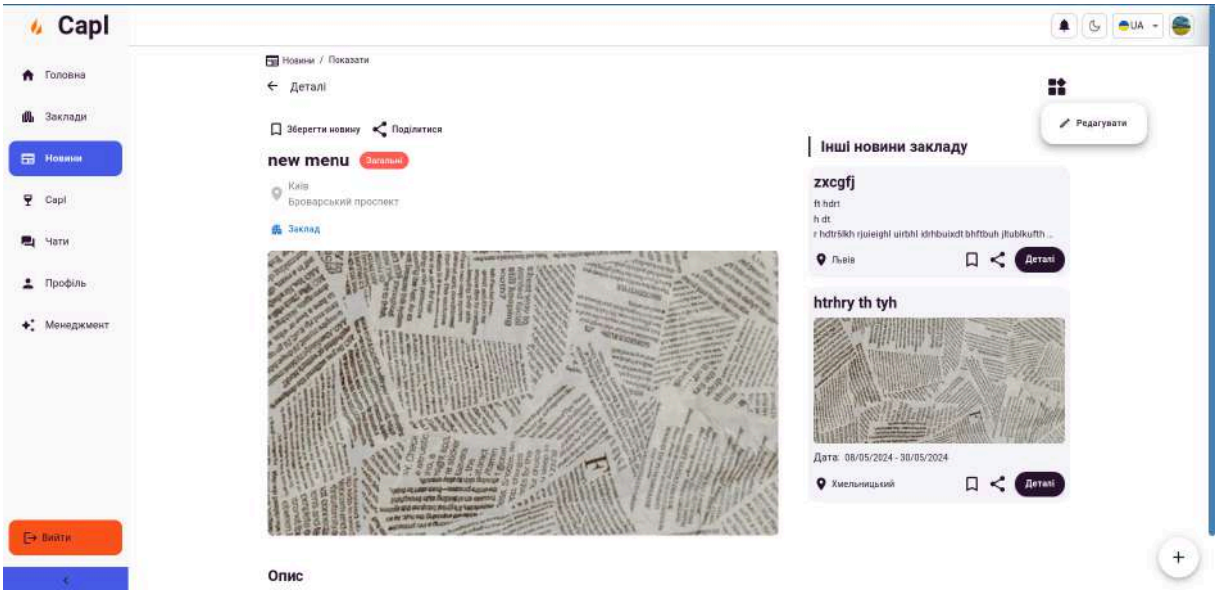


Рис. 22. Меню для редагування інформації про новину

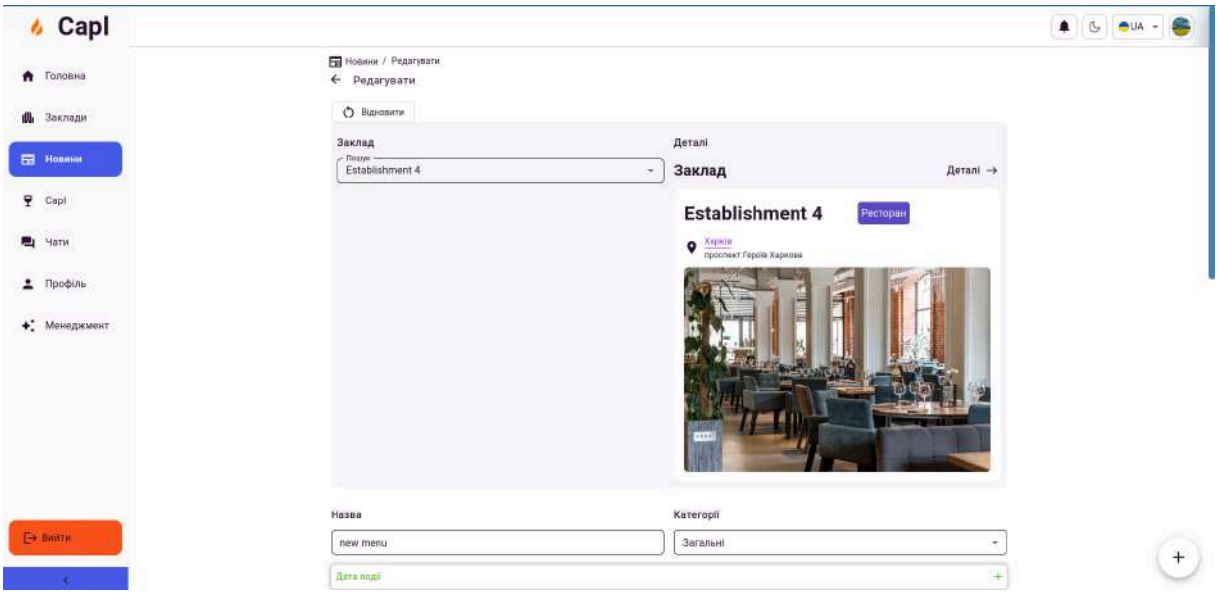


Рис. 23. Сторінка редагування новини

11. Сторінка перегляду резервувань

Сторінка перегляду закладів (рис. 24) складається з двох основних

блоків:

1. Фільтри, для пошуку резервувань за різними категоріями.
2. Список резервувань вибраних за фільтрами.

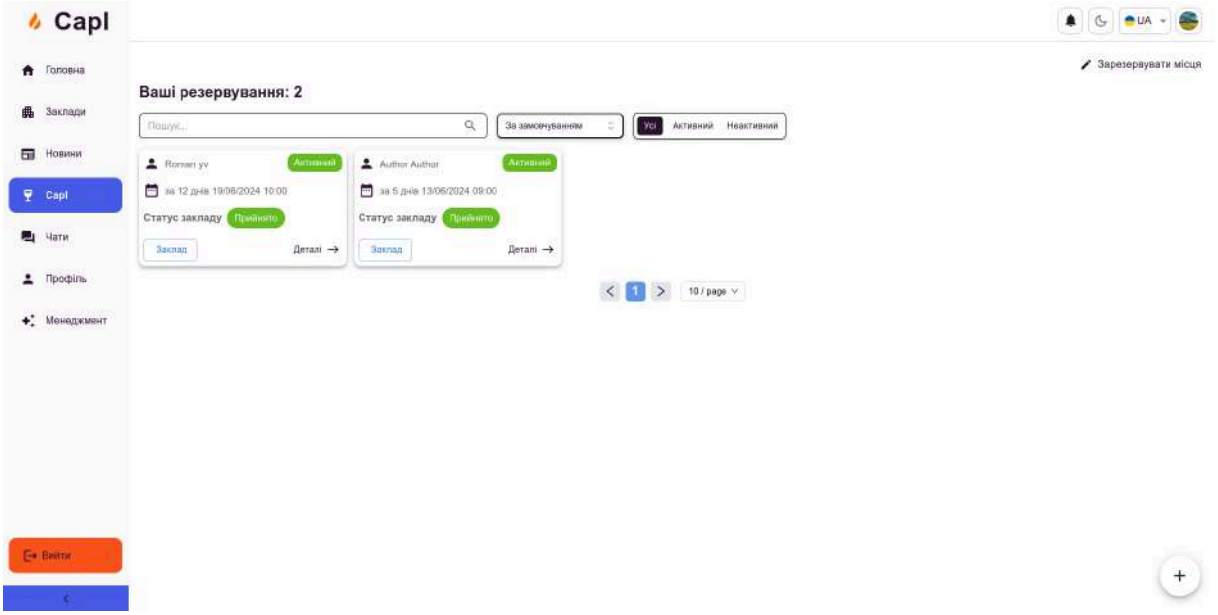


Рис. 24. Сторінка перегляду резервувань

Основними фільтрами на сторінці є:

1. Поле пошуку, в якому можна ввести текст для пошуку резервувань за такими полями: ім’я, хто буде платити, назва події, коментар.
2. Поле вибору метода сортування резервувань.
3. Вибір актуальних резервувань, чи є бронювання активним, тобто ще не минула дата на яку зарезервовано місця, або заклад не підтвердив резервування місць.

Список резервувань містить картки, на яких відображається основна інформація, така як: дата резервування, ім’я, статус закладу, посилання на сторінку закладу та статус резервування. Щоб переглянути всю інформацію про резервування, необхідно на картці натиснути “Деталі”, після чого користувача буде перенаправлено на сторінку з детальною інформацією про резервування.

12. Персональна сторінка резервування

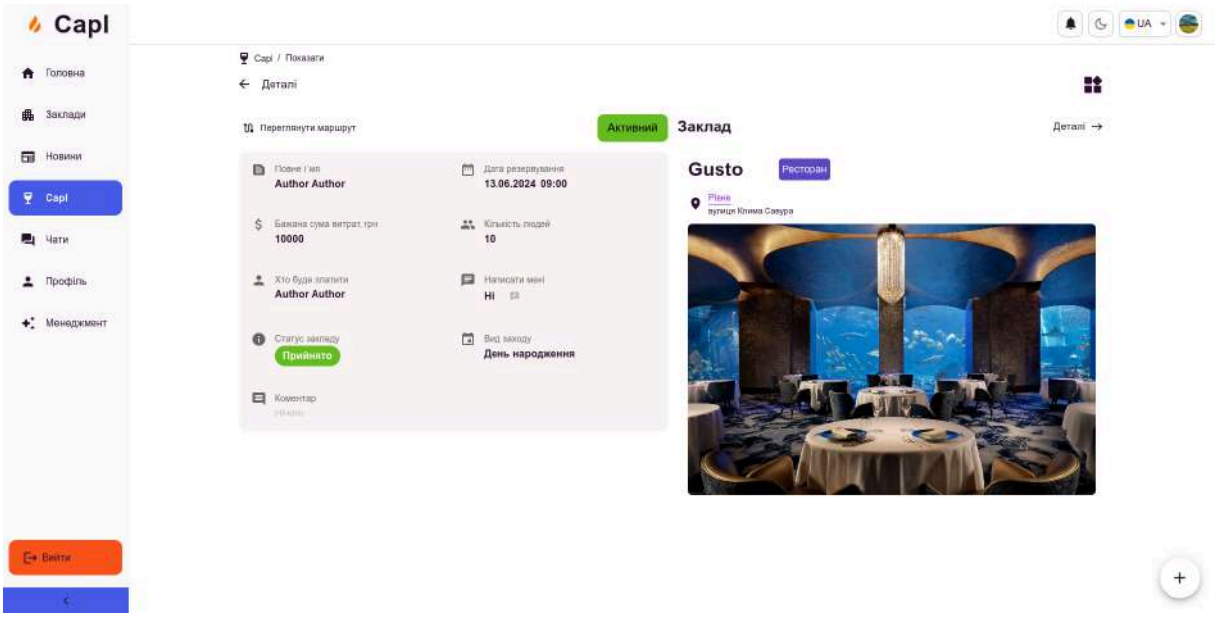


Рис. 25. Персональна сторінка резервування

На персональній сторінці резервування (рис. 25) можна побачити детальну інформацію, а саме:

1. Повне ім’я.
2. Хто буде платити, після події на яку зарезервовано місця.
3. Бажана сума витрат.
4. Кількість людей.
5. Дата резервування.
6. Статус закладу.
7. Вид заходу.
8. Коментар.
9. Позначка чи написати користувачеві для уточнення деталей.

Також поряд з інформацією про резервування знаходиться певна інформація про заклад, яка містить посилання на персональну сторінку закладу.

13. Сторінка керування резервуванням

Щоб потрапити на сторінку керуванням інформації про резервування, необхідно спочатку перейти на його персональну сторінку, у правому верхньому куті буде відображатися кнопка меню. При натисканні відкривається меню, де відображається кнопка “Редагувати”, при кліку користувача перенаправляє на сторінку редагування (рис. 26-27). Важливо, що звичайний користувач може редагувати інформацію лише доки заклад не прийняв резервування. Менеджер може редагувати інформацію у будь-який момент.

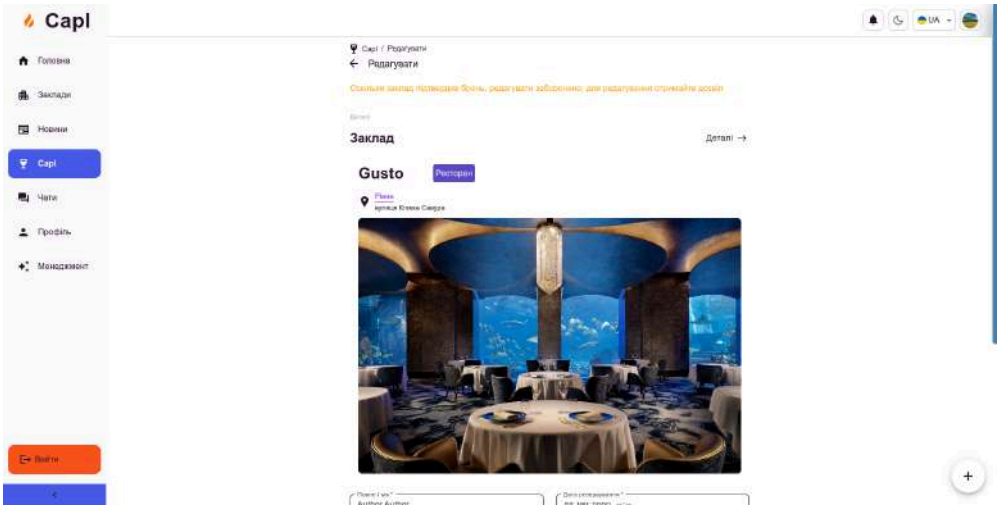


Рис. 26. Сторінка редагування резервування. Перша частина

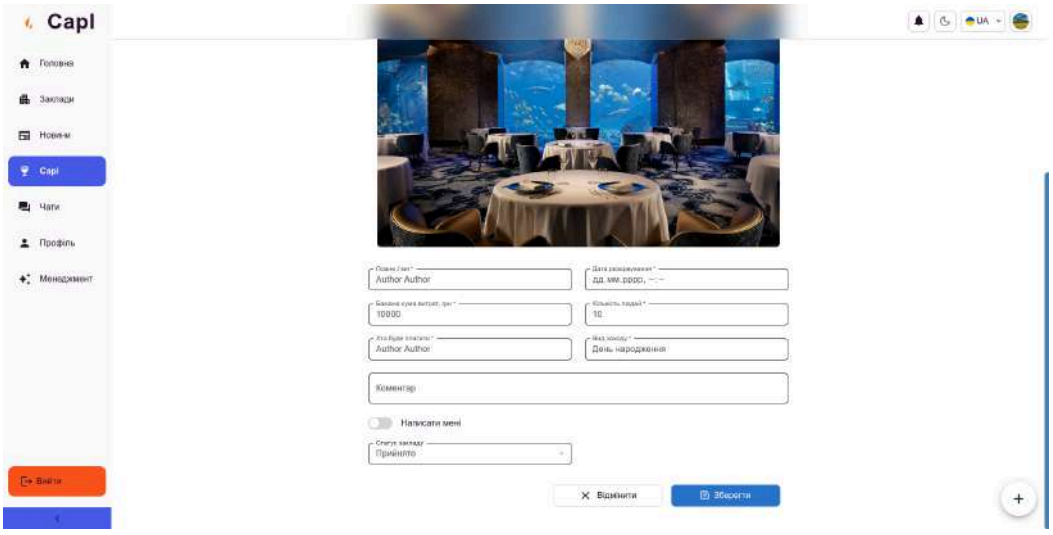


Рис. 27. Сторінка редагування резервування. Друга частина

14. Сторінка створення резервування

Для того, щоб зарезервувати місце на платформі є три методи:

1. На персональній сторінці закладу є кнопка “Зарезервувати місця”.
2. На сторінці перегляду резервувань є кнопка “Зарезервувати місця”.
3. На кожній сторінці є кнопка “+”, при кліку на яку відкривається меню, яке містить кнопку “Зарезервувати місця”.

Після натиснення “Зарезервувати місця” користувача перенаправить на сторінку створення резервування (рис. 28-30).

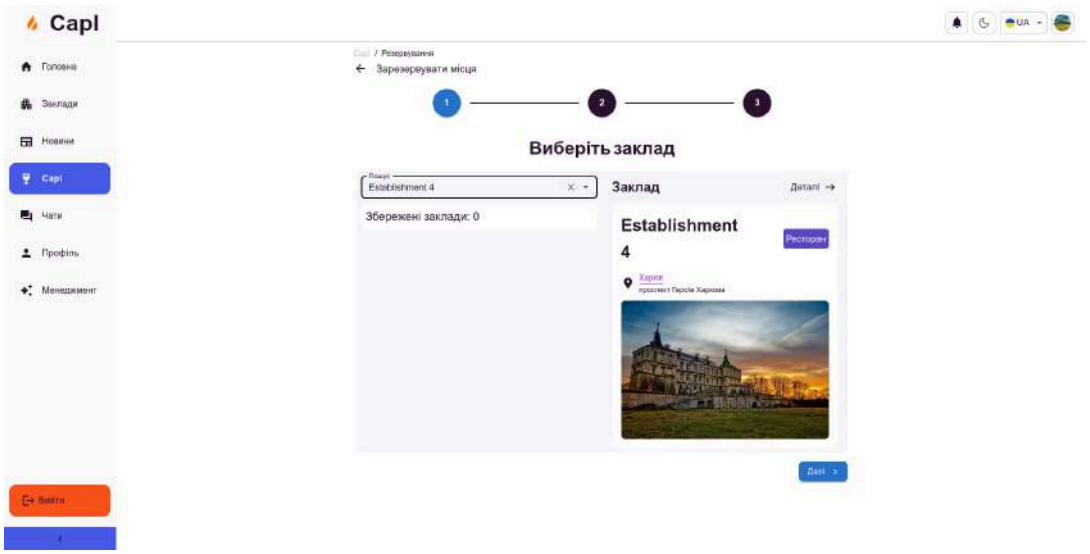


Рис. 28. Сторінка створення резервування. Вибір закладу

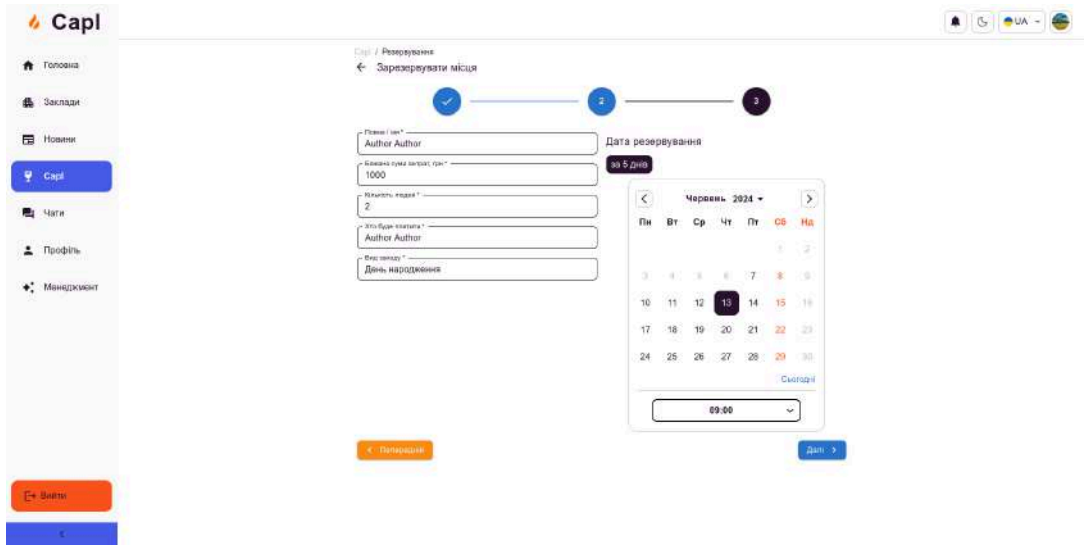


Рис. 29. Сторінка створення резервування. Введення основних даних

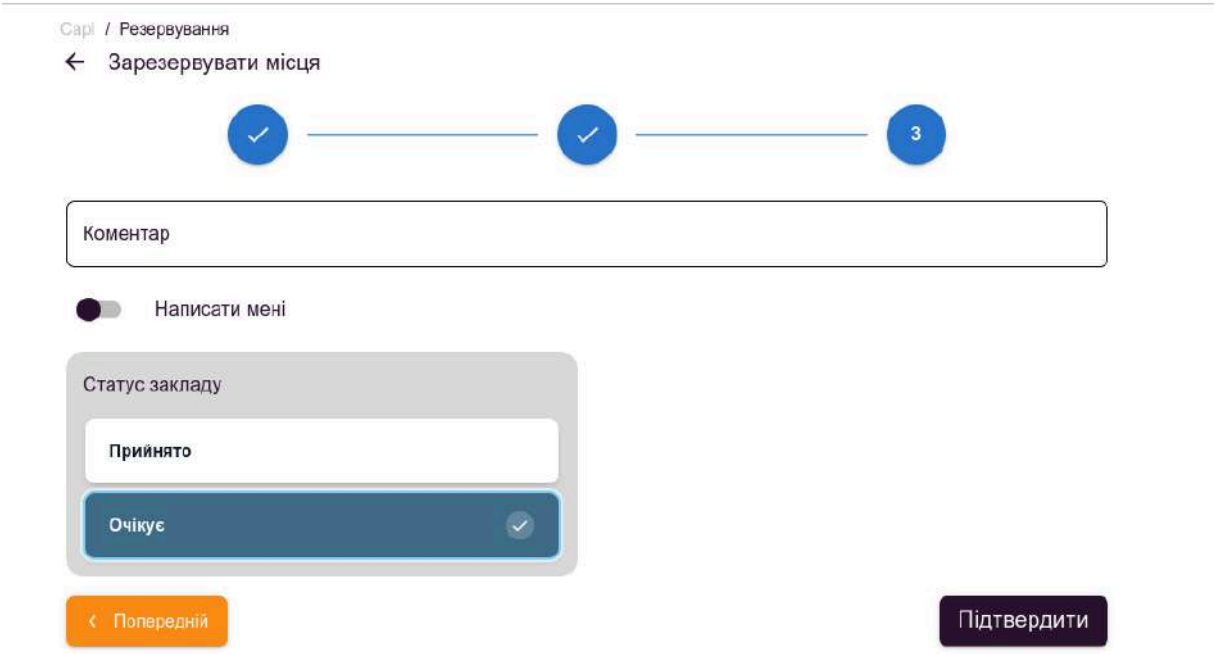


Рис. 30. Сторінка створення резервування. Введення додаткових даних

Щоб зарезервувати місце початку необхідно вибрати заклад. Для цього можна використати поле пошуку або вибрати заклад зі збережених, потім натиснути кнопку “Далі”. Після чого необхідно ввести основні дані, такі як:

1. Повне ім’я.
2. Хто буде платити, після події на яку зарезервовано місця.
3. Бажана сума витрат.
4. Кількість людей.
5. Дата резервування.
6. Вид заходу.

На третьому етапі можна ввести додаткові дані, а саме:

1. Коментар.
2. Позначка чи написати користувачеві для уточнення деталей.

3. Якщо резервування створює менеджер заклад, то він має можливість вибрати статус резервування, чи воно є прийняте чи має статус очікування.

Для того створити резервування необхідно натиснути “Підтвердити”. Після чого буде відправлено запит на сервер для створення резервування.

15. Сторінка перегляду профілю користувача

Сторінка перегляду профілю користувача (рис. 31) складається з інформації про самого користувача, та його даних, таких як: збережені заклади та новини, коментарі, відгуки які залишав користувач, недоступні для ролі “Менеджер”, для ролі “Менеджер” є можливість перегляду власних закладів. Інформація про користувача включає наступне: роль користувача, номер телефону, електронна адреса та дата народження.

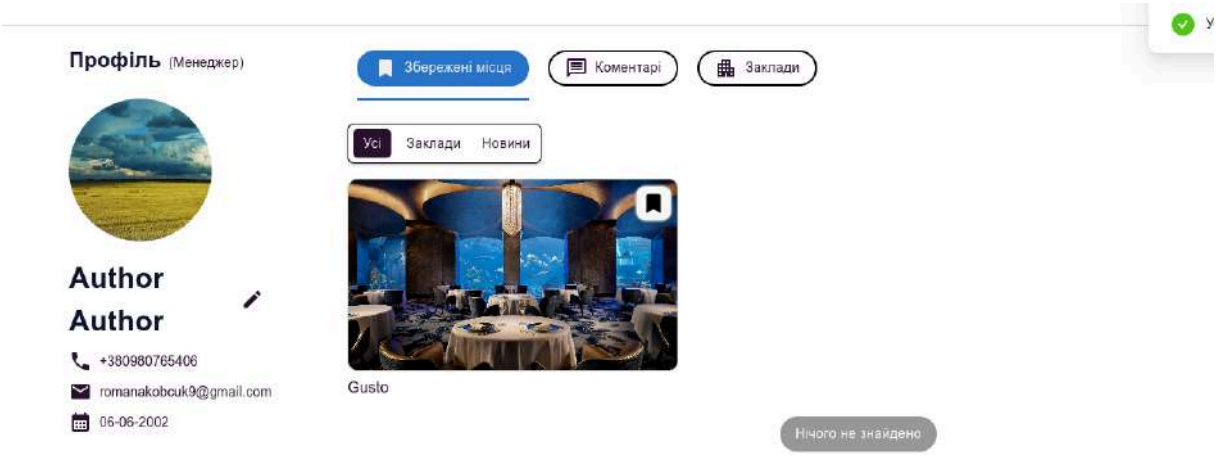


Рис. 31. Сторінка перегляду профілю користувача

16. Сторінка менеджменту

На сторінці менеджменту (рис. 32), яка доступна тільки менеджерам закладів, менеджер може вибрати, з якими залежностями взаємодіяти, із закладами, новинами чи резервуваннями. Для переходу на певну сторінку

(заклади, новини, резервування) для менеджменту, необхідно натиснути на відповідну картку.

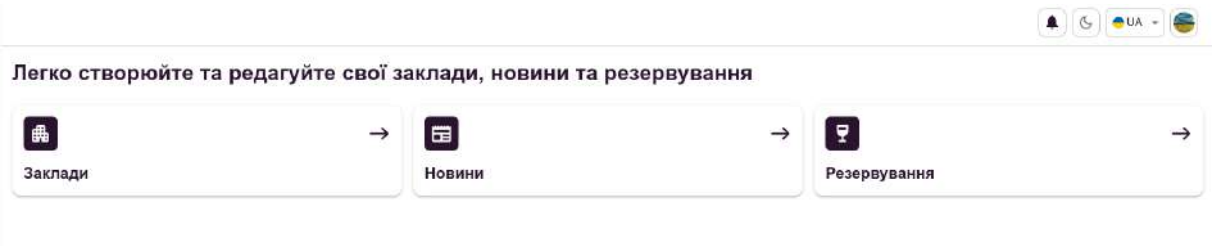


Рис. 32. Сторінка менеджменту

Кожна сторінка із залежностями (рис. 33) має схожу будову та містить такі елементи:

1. Кнопка додавання нової залежності.
2. Фільтри пошуку.
3. Поле пошуку.
4. Таблиця з відображеною інформацію про заклади, новини або резервування відповідно до обраної залежності.

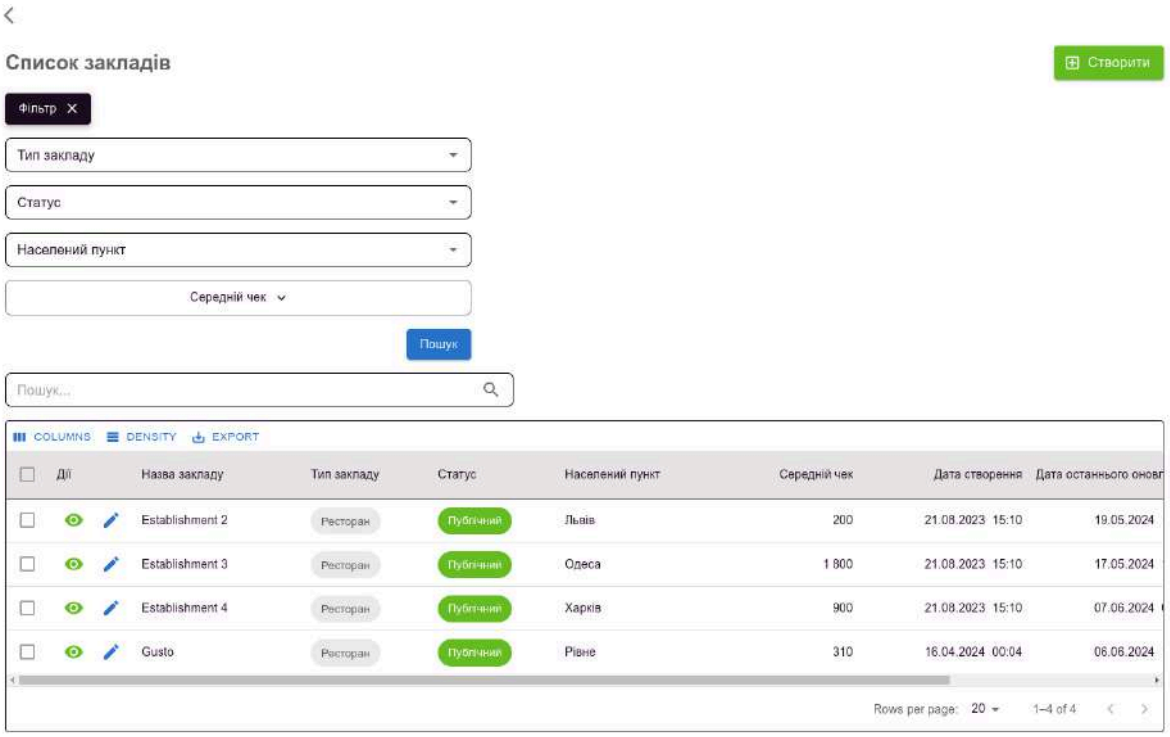


Рис. 33. Сторінка менеджменту закладів