

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук та технологій
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему **РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ТОВ «ТЕРРА»**

Виконав: студент 4 курсу, групи КНТ-520
спеціальності _____

123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація) _____

«Комп'ютерна інженерія»

УДОТ Д.А.

(ПРИЗВИЩЕ та ініціали)

Керівник РИБІН В.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ**Завідувач кафедри КУДЕРМЕТОВ Р.К.**

“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

УДОТУ Денису Андрійовичу

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка комп'ютерної системи ТОВ «Терра»керівник проєкту (роботи) старший викладач, РИБІН Вадим Олегович

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “01” квітня 2024 року № 105

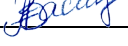
2. Строк подання студентом проєкту (роботи) 10 червня 2024 року3. Вихідні дані до проєкту (роботи) Існуючі підходи та техніки створення вебсайтів. Програмне заезпечення для розробки баз даних. Візуальні інструменти для створення діаграм і креслень..

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Теоретичні основи процесу бронювання в ресторанному бізнесі;
2) Проєктування комп'ютерної системи;
3) Розробка комп'ютерної системи;
4) Тестування комп'ютерної системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Діаграма використання;Структурна бази даних;Технологічний стек;Інтерфейс вебзастосунку.

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	РИБІН В.О., ст. викл.		
нормоконтроль	ПОЛЬСЬКА О.В., ст. викл.		

7. Дата видачі завдання 01.04.2024 р.

КАЛЕНДАРНИЙ ПЛАН

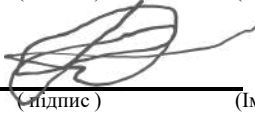
№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз наявних розробок для бронювання столиків	05.03.2024 р.	
	Аналіз недоліків та переваг		
2	Розробка специфікації вимог	05.04.2024 р.	
3	Вибір архітектури комп'ютерної системи	20.04.2024 р.	
4	Розробка бази даних	01.05.2024 р.	
5	Розробка комп'ютерної системи	10.05.2024 р.	
6	Оформлення отриманих результатів у ПЗ	15.05.2024 р.	
7	Оформлення графічного матеріалу	20.05.2024 р.	
8	Оформлення допоміжного матеріалу	25.05.2024 р.	

Студент (ка)


(підпис)

Денис УДОТ
(Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)


(підпис)

Вадим РИБІН
(Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

ПЗ: 64 с., 13 рис., 11 табл., 12 джерел, 1 додаток.

ВЕБЗАСТОСУНОК, БАЗА ДАНИХ, БРОНЮВАННЯ, CSS, HTML, JAVA SCRIPT, MYSQL

Метою даного дипломного проєкту є розробка комп'ютерної системи для бронювання місць у кав'ярнях для ТОВ «Терра», яке б надавало користувачам можливість з легкістю знаходити кав'ярні за їхніми уподобаннями, переглядати наявність місць та робити бронювання в реальному часі.

Предметом дослідження є процеси бронювання місць у кав'ярнях та оптимізація взаємодії між клієнтом та сервісом. Об'єктом дослідження є комп'ютерна система, що включає вебзастосунок для бронювання місць у кав'ярнях.

Для розробки системи було обрано найбільш підходящі програмні засоби. Як СУБД обрано MySQL. Мови програмування: для верстки JavaScript, CSS, React, для серверної розробки Node.

ЗМІСТ

Вступ.....	6
1 Теоретичні основи процесу бронювання в галузі ресторанного сервісу	8
1.1 Огляд сучасного стану ринку послуг кав'ярень.....	8
1.2 Аналіз існуючих систем бронювання та їх вплив на обслуговування клієнтів ..	9
1.3 Вимоги до розробки комп'ютерної системи.....	14
1.4 Висновки за розділом 1	16
2 Проектування комп'ютерної системи	17
2.1 Розробка специфікації вимог SRS.....	17
2.2 Архітектура комп'ютерної системи	19
2.3 База даних	21
2.4 Висновки за розділом 2	25
3 Розробка комп'ютерної системи	26
3.1 Технологічний стек	26
3.2 Апаратна частина комп'ютерної системи.....	31
3.3 Програмна частина комп'ютерної системи	32
3.4 Висновки за розділом 3	39
4 Тестування комп'ютерної системи	40
4.1 Використання локального сервера	40
4.2 Тестування вебзастосунку	42
4.3 Висновки за розділом 4	46
Висновки.....	48
Перелік джерел посилання	49
Додаток А Текст програми	50
Плакат 1 – Діаграма використання	
Плакат 2 – Структурна бази даних	
Плакат 3 – Технологічний стек	
Плакат 4 – Інтерфейс вебзастосунку	

ВСТУП

У сучасному світі, де темп життя постійно зростає, технології відіграють ключову роль у спрощенні та оптимізації повсякденних завдань. Цифровізація послуг стає невід'ємною частиною бізнес-моделей у багатьох галузях, включаючи гостинність та ресторанний бізнес. В цьому контексті розробка комп'ютерної системи для бронювання місць у кав'ярнях компанії ТОВ «Терра» є актуальною та перспективною задачею, яка вимагає глибокого аналізу потреб ринку та використання сучасних ІТ-рішень [1].

З розвитком Інтернету речей (IoT), штучного інтелекту (AI), машинного навчання та великих даних (Big Data) сфера послуг, зокрема індустрія кав'ярень, зазнає значних змін. Цифрові платформи для бронювання та управління замовленнями дозволяють підприємствам підвищити ефективність обслуговування, а клієнтам - зручніше планувати свій візит. Автоматизація процесів, зокрема бронювання столиків, стає нормою, що вимагає впровадження комплексних веб- та мобільних рішень [2].

Використання хмарних технологій, мобільних додатків та інтеграція з соціальними мережами та системами онлайн-оплати стають стандартом у розробці програм для бронювання. Інтеграція з картографічними сервісами, системами відгуків та рейтингів дозволяє користувачам здійснювати обґрунтований вибір, а аналітика великих даних допомагає підприємствам прогнозувати попит та оптимізувати роботу [1].

Метою даного дипломного проекту є розробка комп'ютерної системи для бронювання місць у кав'ярнях для ТОВ «Терра», яке б надавало користувачам можливість з легкістю знаходити кав'ярні за їхніми уподобаннями, переглядати наявність місць та робити бронювання в реальному часі.

Предметом дослідження є процеси бронювання місць у кав'ярнях та оптимізація взаємодії між клієнтом та сервісом. Об'єктом дослідження є

комп'ютерна система, що включає вебзастосунок для бронювання місць у кав'ярнях.

Для досягнення поставленої мети необхідно виконати наступні задачі:

- аналіз потреб користувачів та вимог до функціоналу системи бронювання;
- дослідження сучасних ІТ-рішень в галузі сервісів бронювання та визначення їх переваг та недоліків;
- проектування архітектури комп'ютерної системи, що забезпечує високу доступність, масштабованість та безпеку даних;
- розробка бази даних;
- тестування та впровадження вебзастосунку.

Завдяки виконанню цих задач, буде розроблено програмне забезпечення, яке не тільки задовольняє потреби сучасного користувача у зручності та доступності послуг, але й сприяє оптимізації бізнес-процесів кав'ярень.

1 ТЕОРЕТИЧНІ ОСНОВИ ПРОЦЕСУ БРОНЮВАННЯ В ГАЛУЗІ РЕСТОРАННОГО СЕРВІСУ

1.1 Огляд сучасного стану ринку послуг кав'ярень

Сучасний ринок послуг кав'ярень в Україні характеризується високим рівнем конкуренції та динамічним розвитком. Зростання урбанізації, підвищення життєвого рівня населення та зміна споживчих звичок сприяли розширенню сегменту кав'ярень. Культура споживання кави в Україні набуває нових форм, кав'ярні перетворюються не лише на місця для швидкого кавового перекусу, але й на зони відпочинку, роботи та соціального спілкування [2].

Тенденції та особливості ринку:

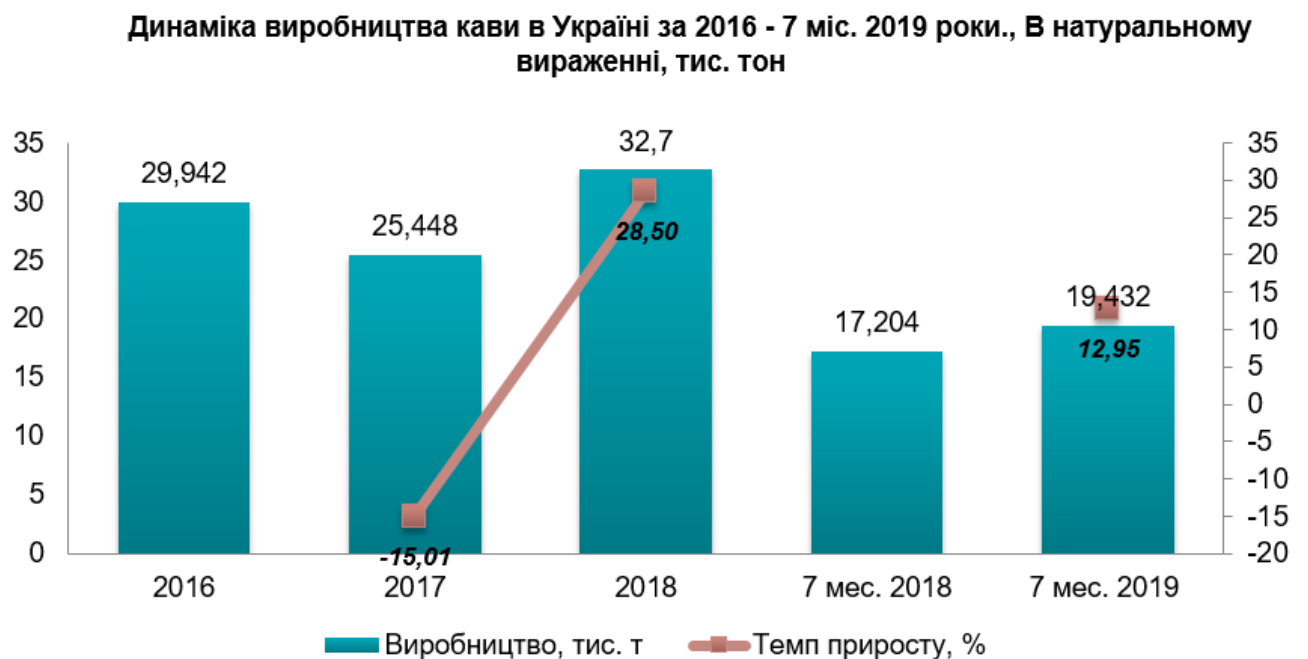
- розмаїття форматів. Від маленьких кавових кіосків і вуличних пунктів до великих тематичних кав'ярень та кавових мереж, які пропонують унікальні атмосферу та асортимент;

- фокус на якості. Споживачі стають більш обізнаними щодо якості кави, що стимулює зростання популярності крафтових кав'ярень та кав'ярень з власною обсмажкою;

- інновації у сервісі. Впровадження цифрових технологій для покращення обслуговування, таких як мобільні додатки для замовлення та оплати, бронювання столиків онлайн, а також програми лояльності;

- соціальна відповідальність. Зростаючий інтерес до екологічних ініціатив, таких як використання біорозкладних матеріалів, зниження відходів та підтримка місцевих виробників.

Однією з головних проблем ринку є висока конкуренція, яка змушує кав'ярні шукати нові способи приваблення та утримання клієнтів. Також, пандемія COVID-19 внесла свої корективи, прискоривши цифровізацію бізнесу та змінивши звички споживачів. Підвищення гнучкості, адаптація до нових умов роботи та інтеграція інноваційних ІТ-рішень стали ключовими факторами виживання на ринку (рис. 1.1).



Джерело: за даними Державної служби статистики України, оцінка Pro-Consulting

Рисунок 1.1 – Динаміка виробництва кави [1]

Перспективи розвитку ринку кав'ярень в Україні залишаються оптимістичними завдяки зростаючому інтересу населення до культури споживання кави, бажанню експериментувати з новими смаками та форматами. Важливу роль у цьому процесі відіграє використання сучасних технологій та цифрових інструментів, які допомагають кав'ярням створювати унікальний досвід для своїх відвідувачів, покращуючи сервіс та оптимізуючи внутрішні процеси.

1.2 Аналіз існуючих систем бронювання та їх вплив на обслуговування клієнтів

При аналізі сучасних українських вебсайтів та додатків для бронювання місць у кав'ярнях, важливо звернути увагу на такі фактори, як зручність користування, дизайн, функціональні можливості та відповідність сучасним

технологіям. На основі аналізу доступних даних можна виділити декілька ключових аспектів, що впливають на успішність таких вебсайтів (табл. 1.1).

Таблиця 1.1 – Ключові аспекти вебсайтів кав'ярень

Аспект	Опис
Різноманіття послуг	Хороший вебсайт або мобільний додаток для бронювання місць у кав'ярнях має пропонувати широкий спектр послуг, включаючи не лише бронювання столиків, але й можливість замовлення їжі та напоїв онлайн, а також перегляд меню.
Якість обслуговування	Відгуки користувачів часто згадують важливість швидкої та якісної обслуговування, як онлайн, так і в самій кав'ярні.
Інновації та технології	Сучасні технології, такі як обсмажування кави, ручне приготування кави та використання високоякісного обладнання для приготування напоїв, є важливими для приваблення клієнтів. Також значення має наявність альтернативних способів заварювання кави, що демонструє високий рівень майстерності бариста.

На основі цих спостережень, при розробці нового вебсайту або мобільного додатку для бронювання місць у кав'ярнях в Україні, рекомендується впровадити наступні інновації та підходи:

- інтеграція з соціальними мережами для легкого доступу до відгуків та оцінок користувачів;
- мультимедійний контент, такий як фотографії меню та інтер'єру, щоб клієнти могли краще уявити атмосферу закладу;
- персоналізація досвіду користувача через рекомендації на основі попередніх відвідувань або переваг;
- забезпечення зручного користування на різних пристроях;
- онлайн підтримка клієнтів для швидкого вирішення будь-яких запитань або проблем.

Check and Place є додатком для обслуговування гостей, який дозволяє управляти резервуванням місць, реєстрацією відвідувачів та розподілом столиків у закладах харчування через планшет. У закладах, де доступне бронювання, Check and Place забезпечує цифрове оформлення резервацій та уникнення помилок, що можуть виникати через людський фактор. Цей інструмент сприяє підвищенню рівня задоволення клієнтів, оптимізуючи весь процес від моменту бронювання до моменту оплати.

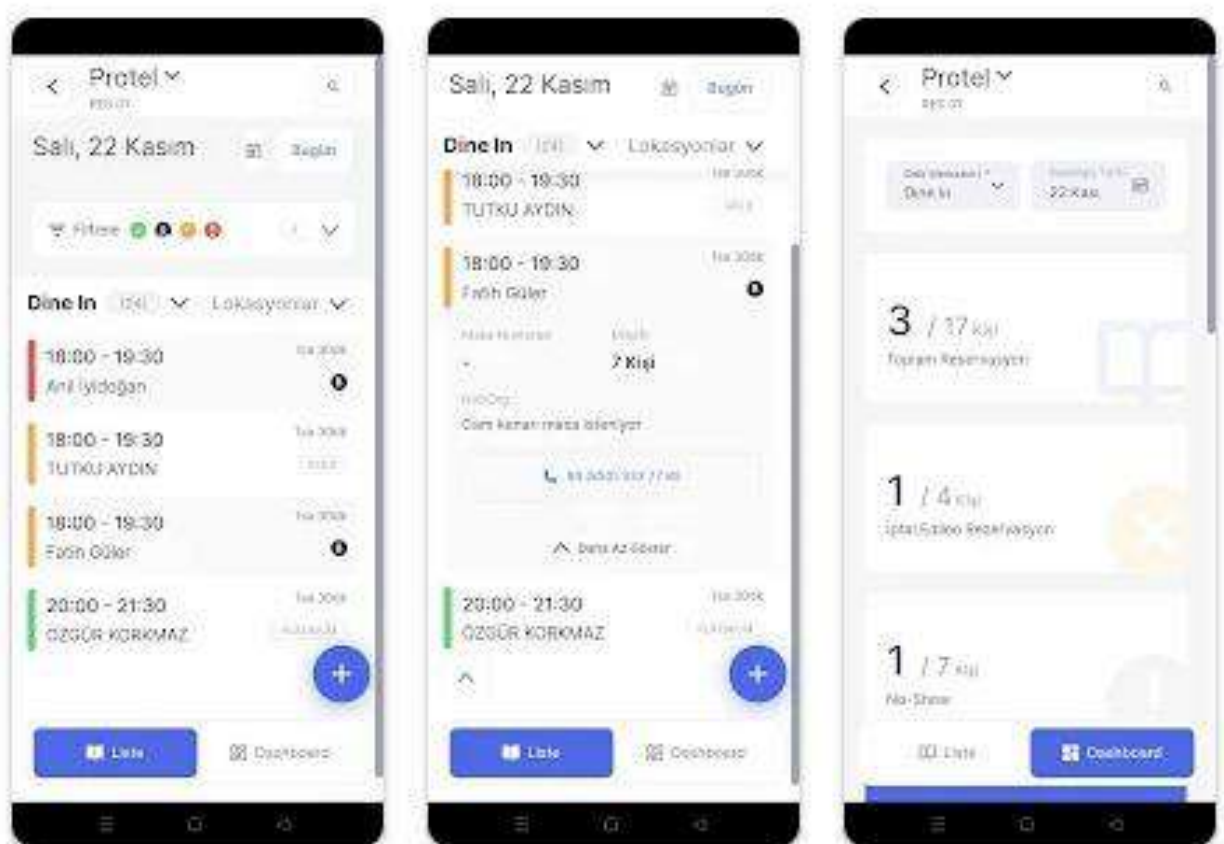


Рисунок 1.2 – Додаток Check and Place

Thestol – це новий безплатний мобільний додаток для резервування столиків, що став доступним у столиці України. Він дозволяє вибрати ресторан з переліку та забронювати місце на бажаний час онлайн.

Розробники ставили за ціль полегшити процедуру походу до ресторану завдяки додатку. Thestol пропонує функцію пошуку закладів за

місцезнаходженням чи за додатковими критеріями, такими як ціновий діапазон, тип кухні, наявність певних зручностей, як-от жива музика або ігрова зона для дітей.

Можливості застосунку:

- онлайн-бронювання столів;
- використання різноманітних фільтрів для пошуку ідеального місця (місцерозташування, тип кухні, розважальні можливості тощо);
- огляд інформації про заклад, включаючи фотографії, меню та вартість страв;
- можливість залишати відгуки та рейтинги для ресторанів;
- отримання особистих знижок та сповіщень про акційні пропозиції;
- отримання останніх новин про дегустації, оновлення меню, сезонні пропозиції та події.

Наразі додаток доступний лише для користувачів iOS, але розробники планують скоро випустити версію для Android.



Рисунок 1.3 – Додаток Thestol

На основі аналізу наведених додатків і загальних тенденцій у сфері кав'ярень в Україні, можна зробити висновок, що при розробці нового сайту для бронювання місць у кав'ярнях, важливо інтегрувати кілька ключових аспектів (табл. 1.2).

Таблиця 1.2 – Ключові аспекти вебзастосування

№	Вимоги до програмного забезпечення
1	Сайт повинен мати привабливий та зручний для користувача інтерфейс, який надає швидкий доступ до основних функцій, таких як перегляд меню, бронювання столиків, та інформації про кав'ярню.
2	Сайт має використовувати сучасні технології для покращення досвіду користувача, такі як віртуальні тури кав'ярень, онлайн замовлення, та інтеграція з соціальними мережами для відгуків та рейтингів.
3	Сайт повинен надавати детальну інформацію про каву, що подається, включаючи походження кавових зерен, способи приготування та рекомендації бариста. Це підкреслить високу якість продукції та професійність персоналу.
4	Враховуючи звіт Poster про уподобання українських споживачів кави, важливо пропонувати широкий асортимент кавових напоїв, включаючи альтернативні способи заварювання, такі як Hario V60 та фільтр-каву.

Враховуючи ці аспекти, новий вебзастосунок для бронювання місць у кав'ярнях може стати не лише зручним інструментом для клієнтів, але й ефективним засобом просування кав'ярні, підкреслюючи її унікальність, якість обслуговування та професіоналізм.

1.3 Вимоги до розробки комп'ютерної системи

Таблиця 1.3 надає огляд ключових завдань, які необхідно виконати при розробці комп'ютерної системи для бронювання місць в кав'ярні ТОВ «Терра», починаючи від аналізу потреб користувачів та закінчуючи підтримкою та розвитком системи.

Таблиця 1.3 - Огляд ключових завдань

Етап	Опис завдань
Визначення потреб користувачів	Аналіз потреб кінцевих користувачів та адміністрації кав'ярень для визначення функціональних вимог до системи.
Аналіз конкурентних рішень	Дослідження існуючих систем бронювання для визначення їх сильних та слабких сторін.
Формулювання вимог до системи	Формулювання технічних та функціональних вимог до нової системи на основі зібраної інформації.
Вибір технологічного стеку	Вибір мов програмування, фреймворків, баз даних та інших технологій для реалізації системи.
Проектування архітектури системи	Створення архітектурної схеми системи, яка описує взаємодію компонентів, модульну структуру та ін.
Розробка інтерфейсу користувача	Дизайн інтерфейсу користувача, який має бути інтуїтивно зрозумілим та зручним.
Розробка основних модулів	Реалізація ключових модулів системи, включаючи модуль бронювання, управління закладом та облік клієнтів.
Тестування	Перевірка системи на наявність помилок, відповідність вимогам і надійність роботи.

Продовження таблиці 1.3 - Огляд ключових завдань

Етап	Опис завдань
Впровадження та налагодження	Запуск системи в експлуатацію з подальшим моніторингом і налагодженням її роботи.
Підтримка та розвиток	Забезпечення оперативної технічної підтримки, оновлення та розширення функціоналу системи.

Таблиця 1.3 детально висвітлює ключові завдання, які виконуються на кожному етапі розробки комп'ютерної системи для бронювання місць в кав'ярні ТОВ «Терра». Ця таблиця охоплює весь процес розробки — від початкового аналізу потреб користувачів до підтримки та розвитку готової системи.

На початковому етапі «Визначення потреб користувачів» відбувається аналіз потреб як кінцевих користувачів, так і адміністрації кав'ярень. Цей аналіз важливий для визначення функціональних вимог до системи, зокрема, які можливості має включати система, щоб задовольнити потреби всіх сторін.

Другий етап "Аналіз конкурентних рішень" передбачає дослідження існуючих на ринку систем бронювання для визначення їхніх сильних та слабких сторін. Це дозволяє зібрати ідеї для інновацій та вдосконалень у розроблюваній системі, а також уникнути поширених помилок.

На етапі "Формулювання вимог до системи" формулюються технічні та функціональні вимоги на основі зібраної раніше інформації. Це створює основу для всіх наступних стадій розробки.

"Вибір технологічного стеку" включає визначення мов програмування, фреймворків, баз даних, і інших технологій, які будуть використані для реалізації системи. Правильний вибір технологій є критичним для успішної реалізації проекту.

"Проектування архітектури системи" включає створення архітектурної схеми, яка описує взаємодію компонентів, модульну структуру та інші аспекти системи. Це допомагає команді розробників мати чітке уявлення про будову системи.

"Розробка інтерфейсу користувача" фокусується на дизайні інтерфейсу, що має бути не тільки привабливим, але й інтуїтивно зрозумілим та зручним для користувача.

"Розробка основних модулів" включає реалізацію ключових компонентів системи, таких як модуль бронювання, управління закладом, і облік клієнтів.

"Тестування" є важливою фазою, де система перевіряється на наявність помилок, відповідність функціональним вимогам і надійність роботи.

На завершальних етапах, "Впровадження та налагодження" та "Підтримка та розвиток", система запускається в експлуатацію, а потім відбувається її моніторинг, налагодження, оновлення і розширення функціоналу для забезпечення її ефективності та відповідності сучасним потребам.

1.4 Висновки за розділом 1

Ринок послуг кав'ярень в Україні є високо конкурентним і динамічним, з зростаючим акцентом на якості, інноваціях у сервісі та соціальній відповідальності. Перспективи розвитку сприятливі завдяки зміні споживчих уподобань. Існуючі системи бронювання місць у кав'ярнях мають певні недоліки, такі як недостатня зручність, обмежений функціонал та застарілий дизайн. Для підвищення конкурентоспроможності необхідно впроваджувати інновації з урахуванням сучасних тенденцій.

При розробці нового програмного забезпечення для бронювання столиків у кав'ярнях важливо інтегрувати такі аспекти: привабливий та зручний інтерфейс, сучасні технології (віртуальні тури, онлайн-замовлення), детальну інформацію про каву та різноманітний асортимент напоїв.

Розробка якісної комп'ютерної системи вимагає ретельного планування та виконання ключових етапів: аналізу потреб користувачів, формулювання вимог,

вибору технологій, проектування архітектури, розробки, тестування, впровадження та подальшої підтримки.

Новий вебсайт чи мобільний додаток для бронювання місць у кав'ярнях може стати не лише зручним інструментом для клієнтів, але й ефективним засобом просування закладу, підкреслюючи його унікальність та високий рівень сервісу.

2 ПРОЄКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

2.1 Розробка специфікації вимог SRS

Вимоги до програмного забезпечення комп'ютерної системи для бронювання столиків обумовлюють набір специфікацій та критеріїв, що визначають бажані характеристики, функціональні можливості та якість розроблюваної програми. Ці вимоги формуються під час аналізу потреб і документуються у вигляді специфікацій вимог, діаграм важливості та інших документів аналізу та розробки.

Для системи бронювання столиків вимоги до користувачів включають:

- можливість легкого вибору дати та часу;
- інтуїтивно зрозумілий інтерфейс для зручності користувачів;
- ефективна система навігації для швидкого доступу до резервацій;
- змогу перегляду історії бронювань;
- процес реєстрації та авторизації;
- інформацію про ресторан та доступні столики.

Адміністратори системи мають такі вимоги:

- інструменти для управління інформацією про бронювання;
- функції для оновлення даних про столики та доступність;
- зручний інтерфейс для адміністрування бази даних;
- можливості редагування, видалення або додавання деталей бронювання;

- перегляд і управління користувацькими даними;
- доступ до статистики бронювань для аналізу попиту.

Крім того, система для бронювання столиків має забезпечувати такі можливості, як:

- категоризація ресторанів за розташуванням;
- інтеграція з платіжними системами для оплати онлайн;
- отримання сповіщень про статус бронювання.

Специфікація вимог до комп'ютерної системи включає детальний опис бажаної поведінки програми, з врахуванням всіх можливих сценаріїв взаємодії між користувачами та системою, а також описує нефункціональні вимоги, такі як продуктивність та стандарти якості.

Таким чином, система бронювання столиків повинна відповідати сучасним стандартам зручності та ефективності, забезпечуючи надійний та зручний доступ до послуг резервації для користувачів та адміністраторів.

Рисунок 2.1 ілюструє загальну схему використання системи.

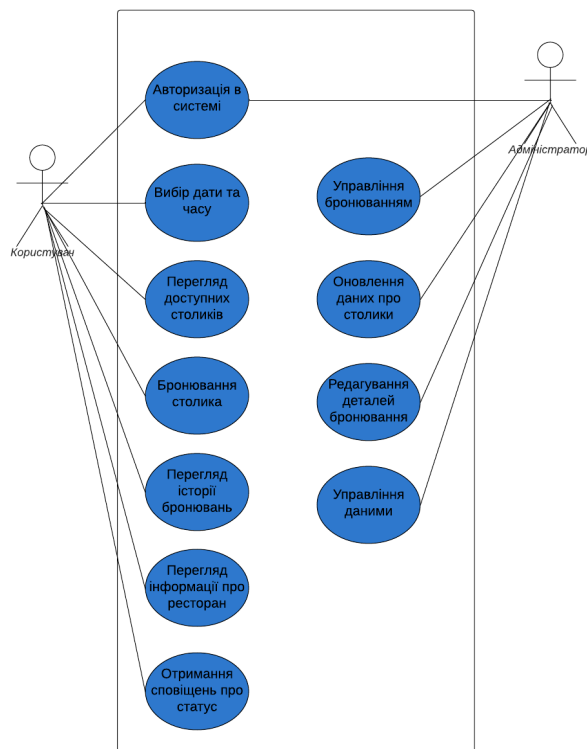


Рисунок 2.1 – Діаграма варіантів використання системи

2.2 Архітектура комп'ютерної системи

Архітектура комп'ютерної системи для бронювання столиків має вирішальне значення для забезпечення ефективності, масштабованості та надійності сервісу. Існує кілька можливих підходів до архітектури таких систем, зокрема монолітна, мікросервісна, та клієнт-серверна. Кожна має свої особливості та призначена для різних сценаріїв використання.

Монолітна архітектура інтегрує всі компоненти системи в один великий блок, що спрощує розробку та розгортання у короткостроковій перспективі, але може створити проблеми з масштабуванням і оновленням у майбутньому. Мікросервісна архітектура, навпаки, розділяє систему на багато незалежних сервісів, кожен з яких відповідає за певний аспект функціональності, що сприяє гнучкості та легкості масштабування, але може збільшити складність управління і координації між сервісами (рис. 2.2).

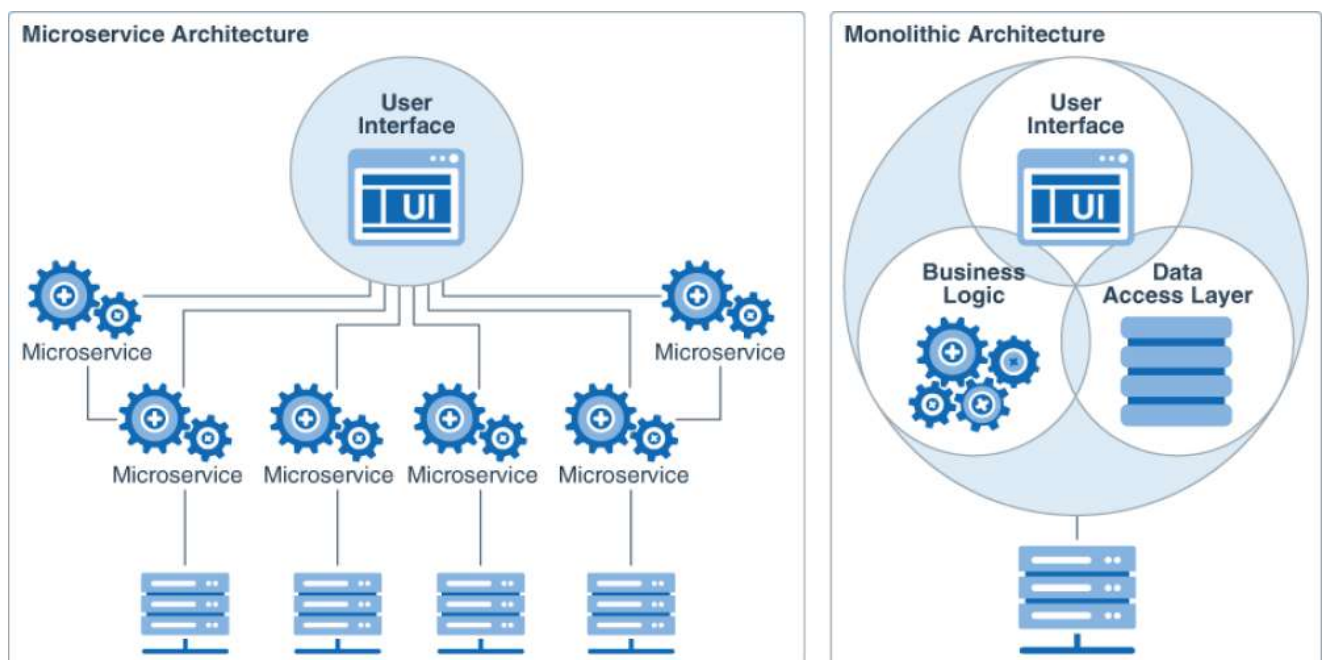


Рисунок 2.2 – Монолітна архітектура

Архітектура клієнт-сервер стала популярним вибором для веб-застосунків, включно з системами бронювання. Вона поділяє систему на дві основні частини: клієнт, що взаємодіє з користувачем, і сервер, що обробляє дані, логіку та управління ресурсами. Цей підхід відомий своєю здатністю ефективно розподіляти навантаження між клієнтом та сервером, що забезпечує вищу продуктивність та кращу шкалабельність (рис. 2.3).

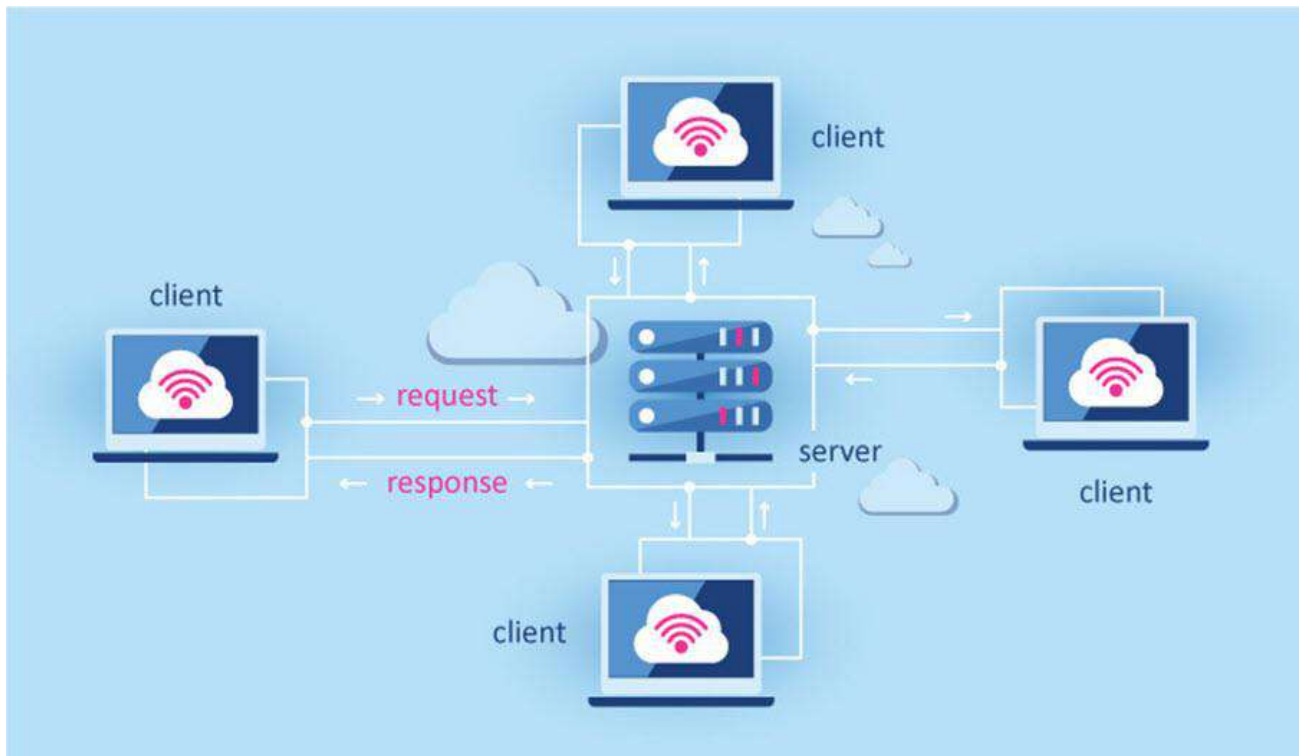


Рисунок 2.3 – Клієнт-серверна архітектура

Переваги клієнт-серверної архітектури для системи бронювання столиків включають зручність розподіленого доступу, де користувачі можуть взаємодіяти з системою з будь-якої точки світу через Інтернет. Сервер може ефективно управляти великими об'ємами даних, такими як інформація про резервації, відгуки користувачів, деталі столиків, та інші. Централізоване управління даними також сприяє кращій безпеці, оскільки усі дані обробляються та зберігаються на сервері, що дозволяє впроваджувати уніфіковані заходи безпеки та резервного копіювання.

Також, клієнт-серверна архітектура дозволяє легко масштабувати систему, збільшуючи кількість серверів або їх ресурси відповідно до зростання навантаження від користувачів, що особливо важливо для популярних сервісів з бронювання. Це забезпечує стабільну та ефективну роботу системи навіть під час піків навантаження, що може виникнути під час важливих свят або заходів.

Враховуючи всі ці переваги, клієнт-серверна архітектура є оптимальним вибором для розробки комп'ютерної системи для бронювання столиків, забезпечуючи надійність, гнучкість та масштабованість, необхідні для сучасного сервісу резервацій.

2.3 База даних

Таблиця 2.1 призначена для зберігання основної інформації про користувачів системи бронювання столиків. Вона забезпечує ідентифікацію та аутентифікацію користувачів, а також зберігає контактні дані для зв'язку.

Таблиця 2.1 - Users

Колонка	Тип даних	Опис
UserID	INT	Унікальний ідентифікатор користувача
Username	VARCHAR(50)	Логін користувача для входу в систему
Password	VARCHAR(255)	Хеш пароля для безпеки користувача
Email	VARCHAR(100)	Електронна пошта користувача
CreatedAt	DATETIME	Дата та час створення запису користувача

Таблиця 2.2 містить всю інформацію про бронювання столиків. Це включає дані про час бронювання, кількість гостей та статус.

Таблиця 2.2 - Reservations

Колонка	Тип даних	Опис
ReservationID	INT	Унікальний ідентифікатор бронювання
UserID	INT	Ідентифікатор користувача, який зробив бронювання
TableID	INT	Ідентифікатор столика, що було заброньовано
ReservationTime	DATETIME	Дата та час бронювання
GuestsCount	INT	Кількість гостей, на яких зроблено бронювання
Status	VARCHAR(10)	Статус бронювання (активна, завершена, скасована)

Таблиця 2.3 забезпечує інформацію про всі столики, доступні для бронювання у системі. Вона включає дані про місткість та розташування кожного столика.

Таблиця 2.3 - Tables

Колонка	Тип даних	Опис
TableID	INT	Унікальний ідентифікатор столика
Capacity	INT	Максимальна кількість людей, які можуть розміститися за столиком
Location	VARCHAR(100)	Опис розташування столика у закладі (наприклад, біля вікна, в куточку)

Таблиця 2.4 містить інформацію про заклади, в яких знаходяться столики для бронювання. Вона важлива для ідентифікації та опису різних локацій.

Таблиця 2.5 для зберігання даних про адміністраторів системи, які мають розширені права для управління резерваціями та іншими параметрами системи.

Таблиця 2.4 - Establishments

Колонка	Тип даних	Опис
EstablishmentID	INT	Унікальний ідентифікатор закладу
Name	VARCHAR(100)	Назва закладу
Location	VARCHAR(255)	Адреса закладу
ContactInfo	VARCHAR(100)	Контактна інформація закладу (телефон або електронна адреса)

Таблиця 2.5 - Admins

Колонка	Тип даних	Опис
AdminID	INT	Унікальний ідентифікатор адміністратора
Username	VARCHAR(50)	Логін адміністратора для входу в систему
Password	VARCHAR(255)	Хеш пароля адміністратора
Email	VARCHAR(100)	Електронна пошта адміністратора
CreatedAt	DATETIME	Дата та час створення запису адміністратора

На рисунку 2.4 показана схема бази даних.

На представленій схемі бази даних виділено п'ять таблиць: Users, Reservations, Tables, Establishments і Admins, кожна з яких має свої унікальні атрибути та взаємозв'язки.

Кожна таблиця обладнана унікальним ідентифікатором (Primary Key, PK), який дозволяє унікально ідентифікувати кожен запис. Наприклад, в таблиці Users цей ідентифікатор називається UserID, у таблиці Reservations — ReservationID, і так далі для інших таблиць. Використання зовнішніх ключів (Foreign Keys, FK) забезпечує створення зв'язків між таблицями, підтримуючи цілісність даних. Наприклад, таблиця Reservations використовує UserID та TableID для зв'язку з таблицями Users і Tables відповідно, що дозволяє відстежувати, який користувач зробив бронювання та на який столик.

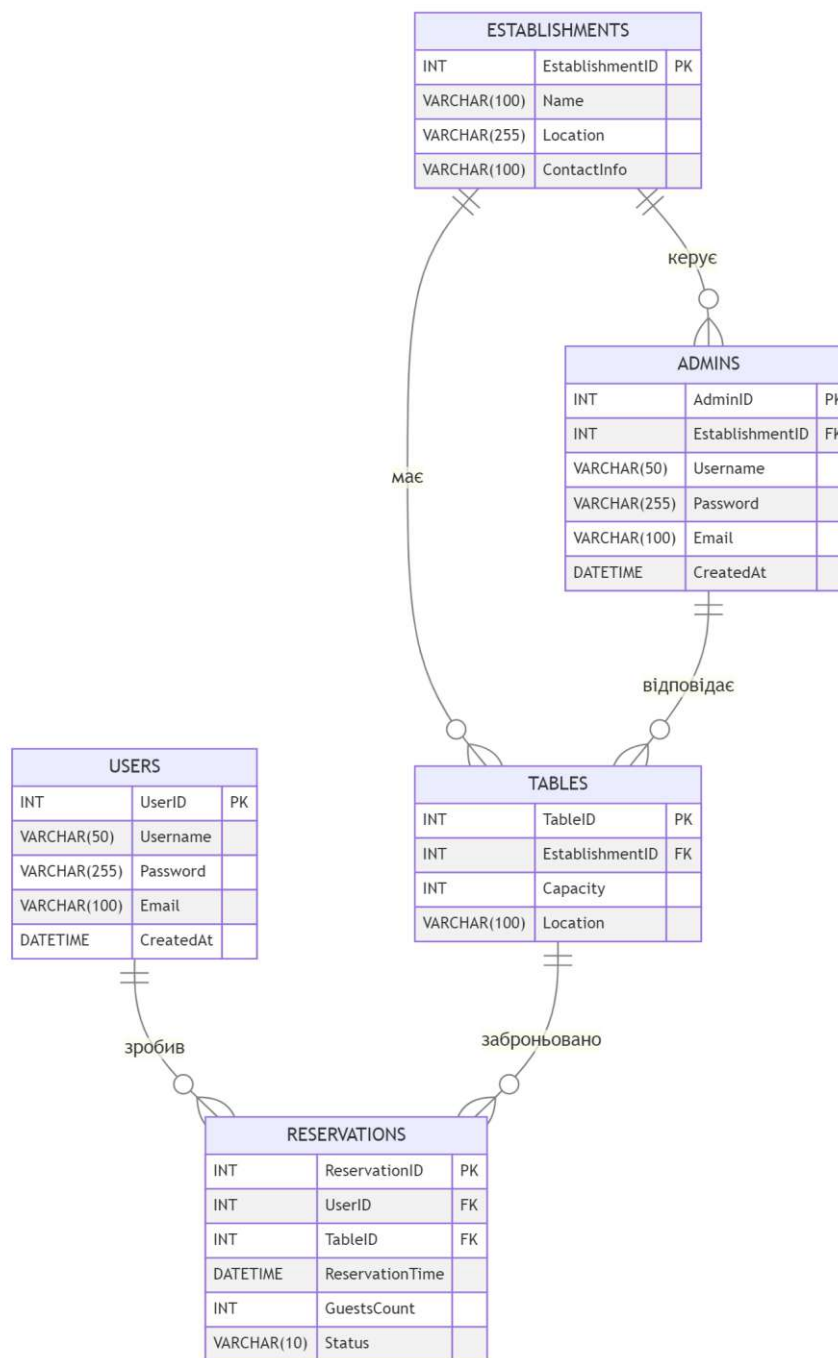


Рисунок 2.4 – Схема бази даних

Таблиця Reservations містить деталі про бронювання столиків, включаючи ID бронювання, час, кількість гостей та статус бронювання. Вона зв'язана не тільки з таблицею Users, але й з таблицею Tables через поле TableID. Цей зв'язок дозволяє відслідковувати, який столик було заброньовано. Зв'язок з таблицею Users вказує на особу, яка здійснила бронювання.

Таблиця Tables зберігає інформацію про столики, що доступні для бронювання. Кожен столик має свій унікальний ID, інформацію про місткість і розташування в закладі. Таблиця Tables має зв'язок з таблицею Reservations, який дозволяє визначати доступність столиків і адміністрування замовлень на конкретні дати.

Таблиця Establishments відображає інформацію про заклади, в яких розташовані столики. Вона зберігає назви, адреси і контактну інформацію закладів. Також ця таблиця зв'язана з таблицею Admins через поле EstablishmentID, що вказує на адміністраторів, які відповідають за управління конкретними закладами.

Таблиця Admins містить дані про адміністраторів системи, їхні ідентифікатори, імена, паролі та електронні адреси. Завдяки зв'язку з таблицею Establishments, можна визначити, які адміністратори відповідають за окремі заклади, надаючи їм можливість управляти бронюваннями і забезпечувати оперативне обслуговування відповідно до потреб закладу.

Таким чином, всі таблиці інтегровані таким чином, що забезпечується ефективне управління користувачами, резерваціями, столиками, закладами і персоналом через централізовану систему управління базою даних. Ця інтеграція дозволяє ефективно координувати діяльність всієї системи бронювання.

2.4 Висновки за розділом 2

У другому розділі було розроблено детальну специфікацію вимог (SRS) для комп'ютерної системи бронювання столиків, яка охоплює функціональні вимоги як для користувачів, так і для адміністраторів системи. Ця специфікація забезпечує чітке розуміння очікуваної функціональності та характеристик системи.

Архітектура системи була ретельно проаналізована, і було обрано клієнт-серверну модель як найбільш підходящу для даного проєкту. Цей вибір обґрунтований перевагами в плані масштабованості, безпеки та ефективності розподілу навантаження.

Розроблено структуру бази даних, яка включає п'ять ключових таблиць: Users, Reservations, Tables, Establishments та Admins. Ця структура забезпечує ефективне зберігання та управління всіма необхідними даними для функціонування системи бронювання.

Створено діаграму варіантів використання (Use Case Diagram), яка наочно демонструє взаємодію різних типів користувачів з системою та основні функціональні можливості, доступні кожному типу користувачів.

Проектування системи було виконано з урахуванням сучасних стандартів розробки програмного забезпечення, що забезпечує надійність, зручність використання та можливість подальшого розширення функціоналу.

Таким чином, другий розділ надає всебічний огляд проектування комп'ютерної системи для бронювання столиків, створюючи міцну основу для подальшої розробки та впровадження системи. Детальна специфікація вимог, обґрунтований вибір архітектури та ретельно спроектована структура бази даних забезпечують високу якість та ефективність майбутньої системи.

3 РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Технологічний стек

Вибір відповідного технологічного стеку має значний вплив на час розробки, витрати, якість програмного продукту та його можливості для масштабування в майбутньому. Засновано на архітектурі, описаній у попередньому розділі, були вибрані певні технології для кожного рівня.

Схематично це відображено на рисунку 3.1, яка демонструє розподіл технологій за рівнями.

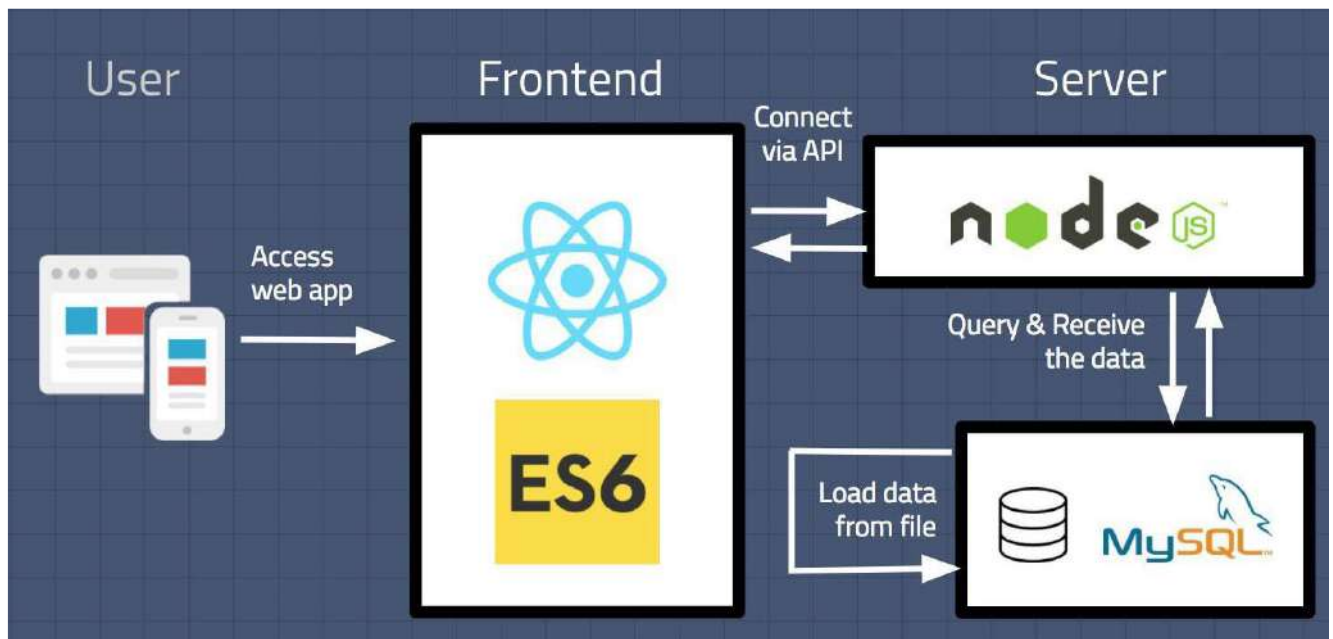


Рисунок 3.1 – Технологічний стек

При вході на вебсайт користувач зустрічається з графічним інтерфейсом, створеним за допомогою React. React є високоефективною бібліотекою JavaScript, розробленою Facebook для створення інтерактивних користувацьких інтерфейсів. Одна з головних переваг React полягає у використанні компонентного підходу до розробки, де інтерфейс розбивається на незалежні, повторно використовувані компоненти, що управляють своїм станом. Це робить код більш організованим і спрощує управління складними проектами, оскільки зміни в одному компоненті не впливають на інші.

React використовує концепцію віртуального DOM, що є однією з ключових інноваційних характеристик цієї бібліотеки. Віртуальний DOM дозволяє React зберігати легку копію структури DOM у пам'яті, що сприяє оптимізації оновлень інтерфейсу. Коли дані змінюються, React спочатку вносить зміни в віртуальний DOM, а потім, за допомогою алгоритму порівняння, визначає найбільш ефективний спосіб внесення цих змін у реальний DOM. Це значно зменшує кількість дорогих в плані продуктивності маніпуляцій з DOM, що робить

оновлення інтерфейсу швидшими і менш витратними з точки зору продуктивності.

Крім того, React підтримує парадигму реактивного програмування та взаємодію з іншими бібліотеками та фреймворками через свій гнучкий API, що робить його відмінним вибором для інтеграції у великі проекти. React має велику спільноту розробників і багатий екосистему інструментів та бібліотек, таких як Redux для управління станом аплікацій і React Router для навігації. Ці інструменти сприяють створенню масштабованих і надійних веб-додатків, які можуть ефективно обробляти складні взаємодії користувачів і великі обсяги даних.

Коли користувач робить запит на отримання даних, інтерфейс підключається до API кінцевих точок, які були реалізовані за допомогою NodeJS та запитує ці дані. NodeJS звертається до бази даних MySQL для отримання даних та відправляє результати назад до веб-інтерфейсу.

Node.js є потужним середовищем виконання, яке дозволяє виконувати JavaScript на серверній стороні. Відрізняється своєю асинхронною, подієво-орієнтованою архітектурою, Node.js дозволяє розробникам побудову швидких та масштабованих мережевих додатків. Використання JavaScript на сервері, а також на клієнті, сприяє більшій уніфікації розробки, знижує криву навчання та підвищує продуктивність розробки завдяки перевикористанню коду та знань.

Однією з ключових переваг Node.js є його неблокуючий ввід/вивід, що дозволяє йому обробляти велику кількість одночасних з'єднань з високою ефективністю. Ця особливість робить його ідеальним для застосунків, які потребують інтенсивної обробки даних у реальному часі, таких як чати, швидкі ігри або системи бронювання, де важлива здатність швидко обробляти велику кількість запитів та відповідей. Використання моделі подій дозволяє легко розширювати додатки без значного збільшення витрат на ресурси сервера.

Крім того, Node.js підтримує широкий спектр пакетів та модулів через систему управління пакетами npm, найбільшу екосистему відкритого програмного забезпечення на світі. Це дає розробникам величезну кількість вже готових

рішень, які можуть бути легко інтегровані в їх проекти, знижуючи час на розробку та підвищуючи загальну надійність продукту. Використання цих інструментів та пакетів може значно прискорити процес розробки та дозволяє розробникам зосередитися на унікальних аспектах своїх проектів замість вирішення загальних проблем.

Аргументи на користь JavaScript як основної мови для веб-розробки очевидні, зважаючи на те, що ця мова використовується в понад 94% вебсайтів. JavaScript служить для клієнтської частини розробки, допомагаючи розробникам створювати динамічні, інтерактивні веб-сторінки і дозволяючи використовувати такі крос-платформні середовища як Node.js для серверної частини. Використання JavaScript сприяє створенню веб-сторінок, що ідеально працюють в різноманітних браузерах, платформах та пристроях, інтегруючи JavaScript з HTML5 і CSS3. Ринок пропонує численні JavaScript платформи, такі як AngularJS, ReactJS, NodeJS, які спрощують розробку і дозволяють створювати масштабовані веб-додатки.

Для створення графічного інтерфейсу користувача було обрано JavaScript-фреймворк React. Сучасні фреймворки для розробки інтерфейсу включають у себе управління HTML-шаблонами, стилями і інтерактивністю. ReactJS не є фреймворком; це бібліотека користувацького інтерфейсу, розроблена Facebook, підтримувана великою спільнотою і орієнтована на складні додатки із повторно використовуваними компонентами. ReactJS має розгалужену екосистему, що включає сам ReactJS, бібліотеку React-DOM для управління DOM об'єктами, React-Router для маршрутизації, і JSX для створення шаблонів у ReactJS. Завдяки цим можливостям, ReactJS є відмінним вибором для розробки мобільних додатків через React Native, який має багато спільного з основним ReactJS.

Середній вибір MySQL для системи управління базами даних було зумовлено її масштабованістю, гнучкістю, високою продуктивністю, надійністю, забезпеченням потужних транзакцій, а також відмінними можливостями для веб-сайтів та сховищ даних. MySQL відома своїм швидким впровадженням,

простотою управління та великою підтримкою спільноти, що робить її ідеальною для швидкої та ефективної розробки веб-проектів.

MySQL є однією з найпопулярніших систем управління базами даних (СУБД), яка використовується в інтернет-проектах по всьому світу. Це відкрите програмне забезпечення, що працює на багатьох платформах, включаючи Linux, Windows і macOS. MySQL відома своєю надійністю, швидкістю обробки даних і простотою використання, що робить її ідеальним вибором для веб-додатків, які вимагають ефективного зберігання та швидкого доступу до великих обсягів даних.

Одним з головних переваг MySQL є її висока продуктивність при роботі з великими базами даних. Це досягається завдяки оптимізованій архітектурі та можливостям індексації, що дозволяє швидко обробляти запити на пошук та вибірку даних. MySQL також підтримує різноманітні типи таблиць, що дають можливість вибирати оптимальний тип зберігання залежно від специфіки проекту. Наприклад, InnoDB підтримує транзакції, що є критично важливим для додатків, де необхідна висока надійність обробки даних.

MySQL також пропонує гнучкість у масштабуванні, як вертикальному, так і горизонтальному. Вертикальне масштабування може бути досягнуте шляхом збільшення потужності сервера, а горизонтальне масштабування — через додавання більшої кількості серверів та використання шардінгу або реплікації. Ці можливості роблять MySQL дуже привабливим вибором для розвитку великих проектів, що вимагають розподіленого зберігання даних та високої доступності.

Окрім того, широка підтримка з боку спільноти і постійні оновлення від Oracle, яка є поточним власником MySQL, гарантують, що ця СУБД постійно вдосконалюється та адаптується до нових технологічних вимог та стандартів. Це робить MySQL надійним рішенням для довготривалого використання в проектах різної складності.

3.2 Апаратна частина комп'ютерної системи

Для розгортання сервера бази даних та веб-сервера на локальному комп'ютері, важливо вибрати апаратну конфігурацію, яка забезпечить оптимальну продуктивність і надійність. На основі цих вимог, розглянемо конкретні компоненти, які будуть найбільш підходящими для такої системи.

Для процесора важливо вибрати багатоядерний варіант з високою частотою роботи. Intel Xeon або AMD EPYC серії процесори будуть ідеальними завдяки їх здатності обробляти великі обсяги даних та велику кількість одночасних запитів, що є критично важливим для серверів.

Оперативна пам'ять також має ключове значення. Мінімум 32 ГБ оперативної пам'яті типу DDR4 дозволить з легкістю управляти великими обсягами даних та користувачами, забезпечуючи швидку реакцію на запити до бази даних та серверних ресурсів.

Щодо дискового простору, SSD є вибором номер один через їх швидкість і надійність. Рекомендується використовувати SSD з мінімальним обсягом 1 ТБ, що дозволить не тільки зберігати великі обсяги даних, але й забезпечити швидкий доступ до них. Для забезпечення додаткової надійності та швидкості можна налаштувати SSD у RAID масиві.

Мережеве обладнання також має важливе значення для підтримки швидкого зв'язку між сервером та клієнтами. Гігабітний мережевий інтерфейс (10 GbE) забезпечить достатньо ширини каналу для високого трафіку, який є типовим для веб-серверів та баз даних.

Нарешті, наявність системи безперебійного живлення (UPS) гарантуватиме, що у випадку відключення електроенергії, сервери продовжать працювати, давши можливість безпечно завершити важливі операції та уникнути втрати даних.

Такий підхід до вибору апаратних компонентів забезпечить створення міцної та ефективної локальної серверної інфраструктури, що відповідає високим вимогам сучасних баз даних та веб-додатків.

3.3 Програмна частина комп'ютерної системи

Програмна частина комп'ютерної системи ТОВ "Терра" розроблена так, щоб забезпечувати ефективну і гнучку взаємодію користувачів з системою через веб-браузери як на комп'ютерах, так і на мобільних телефонах. Ця система включає декілька ключових функцій, які спрощують управління бізнес-процесами та покращують зручність використання для кінцевих користувачів.

Перша важлива функція — реєстрація та авторизація користувачів. Система надає можливість користувачам створювати власні облікові записи, використовуючи свою електронну адресу та пароль, а також виконувати вхід у систему для доступу до персоналізованих функцій. Завдяки використанню адаптивного дизайну, реєстрація та авторизація зручно працюють як на великих, так і на маленьких екранах мобільних телефонів.

Друга функція — мобільний доступ. Система оптимізована для використання в мобільних браузерах, що дозволяє користувачам здійснювати доступ до функціоналу системи з будь-якого місця і в будь-який час. Адаптивний дизайн забезпечує, що інтерфейс виглядає коректно та зручно для використання незалежно від розміру екрану пристрою.

Початковий екран розробленого вебзастосунку призначений для відображення списку закладів, де можливе бронювання столиків (рис. 3.2). На екрані присутній пошуковий бар на верхній частині, що дозволяє користувачам шукати заклади за назвою або іншими ключовими словами. Також є можливість фільтрування результатів за допомогою розкритого меню, яке дозволяє вибрати конкретні параметри, наприклад, тип кухні або ціновий діапазон.

Кожен заклад представлений у формі картки, яка містить фотографію, назву та короткий опис. Додатково вказані спеціалізації закладу, наприклад, тип кухні та середній чек. Під кожним назвою закладу розташований рейтинг, що відображає загальну оцінку відвідувачів. Цей рейтинг супроводжується іконкою серця, яка дозволяє користувачам додавати заклади до свого списку улюблених.

Цей екран ефективно залучає користувачів, надаючи їм інтуїтивно зрозумілий інтерфейс для перегляду та вибору закладів, де вони можуть забронювати столик. Він забезпечує основну інформацію, необхідну для вибору місця, зручно організовану та легко доступну для користувачів.



Рисунок 3.2 – Початковий екран

На рисунку 3.3 представлено інтерфейс для бронювання столика в ресторані через мобільний пристрій. Цей екран дозволяє користувачам обрати дату та час для бронювання. У верхній частині екрана є розгортаючий список з доступними датами та часами. Користувач може вибрати зручний для себе час шляхом натискання на один з варіантів.

Далі на екрані розташований інтерактивний елемент для вибору кількості гостей, що прибудуть разом з користувачем. Вибір кількості гостей виконується за допомогою кнопки зі стрілкою, що веде до додаткового меню або слайдера для налаштування потрібної кількості осіб.

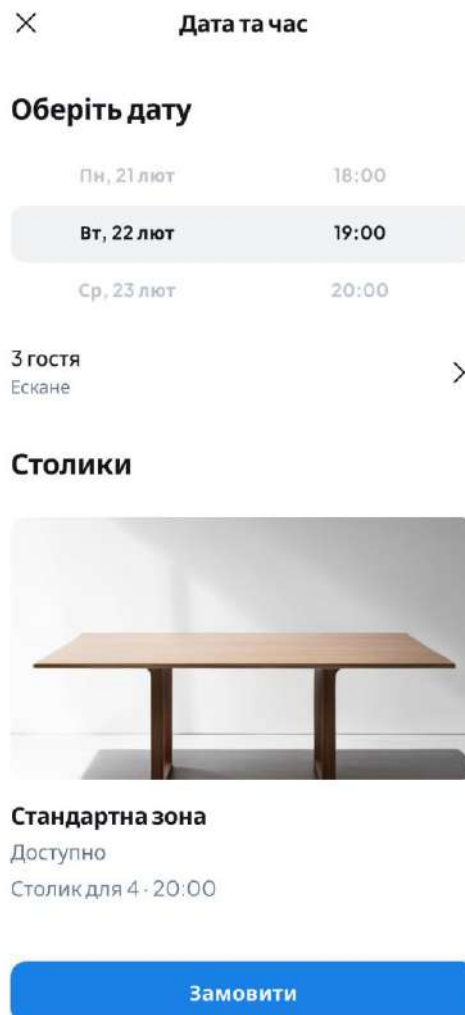


Рисунок 3.3 – Бронювання столика

Нижче розміщено секцію з інформацією про доступні столики. На фотографії показано конкретний столик, його розташування в ресторані (наприклад, у стандартній зоні), а також інформацію про доступність. У даному випадку, столик для чотирьох осіб доступний о 20:00.

Внизу екрана розташована кнопка "Замовити", яка дозволяє підтвердити бронювання після усіх виборів. Ця кнопка активує процес бронювання в системі і є останнім кроком у процедурі замовлення столика.

Рисунок 3.4 показує інтерфейс для перегляду оновлень бронювань. Цей інтерфейс складається з двох основних сегментів: оновлення за сьогодні та оновлення за тиждень.

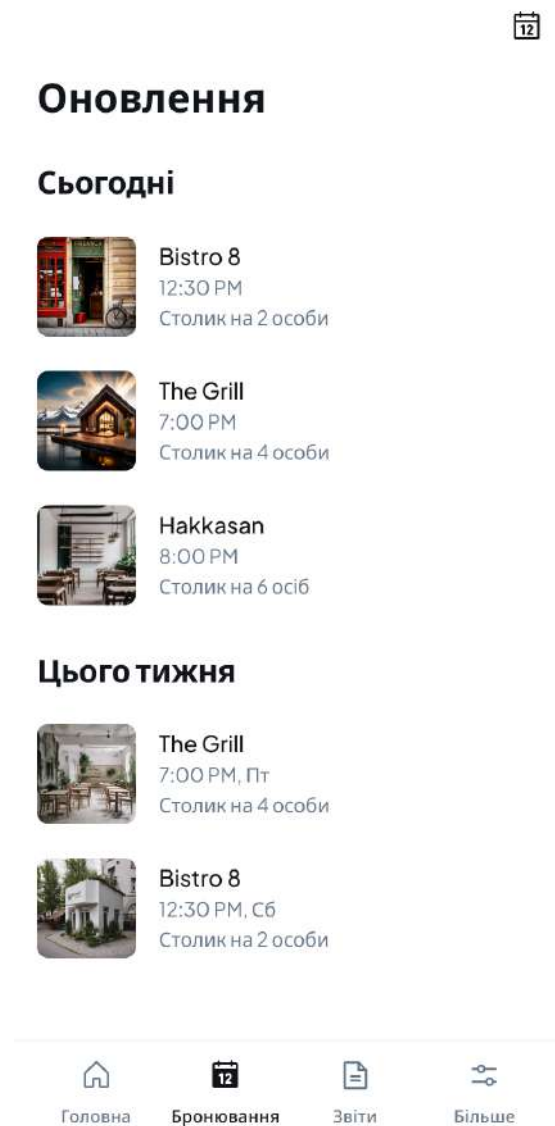


Рисунок 3.4 – Історія бронювання

У верхній частині екрана відображаються оновлення за сьогоднішній день, де користувачі можуть бачити недавні та майбутні бронювання. Кожне бронювання представлено карткою, яка включає зображення закладу, назву закладу, час бронювання та кількість осіб. Наприклад, є бронювання в "Bistro 8" на 12:30 PM на двоє осіб, в "The Grill" на 7:00 PM на чотирьох осіб та в "Hakkasan" на 8:00 PM на шість осіб.

У нижній частині екрана знаходиться секція з оновленнями за тиждень, де відображені заплановані бронювання на майбутнє. Таке розташування дозволяє користувачам швидко перевірити свої плани на тиждень, що сприяє кращому плануванню та організації.

На зображенні 3.5 відображено екран фільтрації для мобільного застосунку, що дозволяє користувачам персоналізувати пошук за різними критеріями. Це допомагає користувачам звужити вибір закладів, що відповідають їхнім уподобанням.

Вибір категорій, таких як "Винний бар", "Кафе", "Паб", дозволяє користувачам швидко відсортувати заклади за типом, спрощуючи пошук спеціалізованих місць. Кнопки категорій мають вигляд тегів, що сприяє легкому вибору та візуальному розпізнаванню.

Слайдер для встановлення діапазону цін дозволяє користувачам встановити мінімальну та максимальну ціну, яку вони готові витратити, що робить пошук більш цільовим та ефективним. Цей інструмент корисний для планування витрат та пошуку закладів, що відповідають бюджету користувача.

Секція для вибору рейтингу містить можливість вибрати мінімальний рейтинг закладу, який користувач вважає прийнятним. Це включає кілька значень, таких як 4.5, 4.0, 3.5, що забезпечує гнучкість при відборі високооцінених закладів.

У нижній частині екрана розміщена кнопка "Відкрито зараз", яка дозволяє швидко знайти заклади, які відкриті у поточний час. Ця функція особливо корисна для користувачів, які шукають місце для відвідування вже наразі.



Рисунок 3.5 – Фільтр для пошуку закладів

На рисунку 3.6 представлено інтерфейс панелі адміністратора для системи бронювання столиків у кафе. Цей інтерфейс дозволяє адміністратору відслідковувати та управляти доступністю столиків, а також переглядати актуальні бронювання.

У верхній частині екрана розміщено заголовок "Система бронювання столиків для кафе", який чітко вказує на призначення цієї сторінки. Нижче розташовані дані про конкретні столики. Кожен столик має індивідуальний номер (наприклад, Стол 2, Стол 3 тощо), інформацію про кількість місць (4 особи, 6 осіб), статус (доступний, зайнятий) і кнопку "Змінити", яка дозволяє адміністратору оновити статус столика або кількість місць.

В нижній частині екрана відображаються актуальні бронювання з інформацією про клієнта, часом бронювання, кількістю осіб і також кнопкою "Змінити" для кожного бронювання. Це дозволяє адміністратору швидко реагувати на зміни в бронюваннях, наприклад, коригувати час або кількість осіб.



Рисунок 3.6 – Панель адміністратора

Внизу екрана розташована кнопка "Нове бронювання", що дозволяє адміністратору додавати нові бронювання вручну. Ця кнопка надає додаткову гнучкість для управління резерваціями, зокрема у випадках, коли клієнти бажають зробити бронювання безпосередньо через адміністратора або по телефону.

На основі наведеного опису можна зробити висновок, що розроблений вебзастосунок для бронювання столиків у закладах харчування має інтуїтивно зрозумілий та функціональний інтерфейс. Система пропонує користувачам зручний початковий екран зі списком закладів, можливістю пошуку та фільтрації, а також детальну інформацію про кожен заклад. Процес бронювання реалізовано через простий та зрозумілий інтерфейс, де користувач може вибрати дату, час та кількість гостей. Додатково, застосунок надає функціонал для перегляду історії бронювань та майбутніх резервацій, що допомагає користувачам ефективно планувати свої візити. Важливим елементом є наявність розширених можливостей фільтрації, які дозволяють користувачам точно налаштувати пошук відповідно до їхніх уподобань. Для адміністраторів системи передбачено окрему панель управління, яка забезпечує ефективний контроль над доступністю столиків та управління бронюваннями. Загалом, описаний вебзастосунок демонструє комплексний підхід до вирішення задачі онлайн-бронювання столиків, забезпечуючи зручність як для користувачів, так і для адміністраторів закладів.

3.4 Висновки за розділом 3

У цьому розділі описано розробку комп'ютерної системи для бронювання столиків у закладах харчування ТОВ «Терра».

Технологічний стек. Система побудована на основі JavaScript з використанням React для фронтенду, Node.js для бекенду та MySQL для бази

даних. Цей вибір забезпечує ефективність, масштабованість та зручність розробки.

Апаратна частина. Рекомендовано використання потужного серверного обладнання, включаючи багатоядерний процесор, мінімум 32 ГБ оперативної пам'яті, SSD-накопичувачі та гігабітне мережеве з'єднання для забезпечення високої продуктивності та надійності системи.

Програмна частина. Детально описано інтерфейс користувача, який включає:

- початковий екран зі списком закладів та можливістю пошуку і фільтрації;
- екран бронювання столика з вибором дати, часу та кількості гостей;
- інтерфейс для перегляду історії та оновлень бронювань;
- розширені можливості фільтрації для точного пошуку закладів;
- панель адміністратора для управління столиками та бронюваннями

Розроблена система демонструє комплексний підхід до вирішення задачі онлайн-бронювання столиків, забезпечуючи зручність використання як для клієнтів, так і для адміністраторів закладів. Вона поєднує сучасні технології, ефективну апаратну конфігурацію та інтуїтивно зрозумілий інтерфейс користувача, що робить її потужним інструментом для управління бронюваннями в закладах харчування.

4 ТЕСТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

4.1 Використання локального сервера

Для розробки та використання системи бронювання столиків, можливість використання локального сервера є ключовою для гнучкого тестування та розгортання. Існує кілька популярних варіантів локальних серверів, які можуть бути ефективно використані в залежності від конкретних потреб системи (табл. 4.1).

Таблиця 4.1 - Порівняння локальних серверів

Характеристика	Apache HTTP Server	Nginx	Microsoft IIS
Головні переваги	Гнучкість, широке підтримування модулів	Висока продуктивність, ефективність	Тісна інтеграція з Windows
Особливості	Підтримка .htaccess, модульна структура	Швидка обробка статичного контенту	Підтримка .NET технологій
Продуктивність	Добре справляється з динамічним контентом	Оптимізований для великої кількості одночасних з'єднань	Оптимізований для продуктів Microsoft
Складність налаштування	Може вимагати складної настройки	Простіший у налаштуванні	Інтегроване управління з Windows
Масштабованість	Добре масштабується з належною конфігурацією	Відмінно масштабується	Добре масштабується в середовищах Microsoft
Безпека	Велика кількість налаштувань безпеки	Високий рівень безпеки за замовчуванням	Сильні налаштування безпеки для Windows середовищ
Підтримка спільноти	Велика спільнота та документація	Широка підтримка спільноти	Підтримка з боку Microsoft та спільноти

Серед відомих варіантів – Apache HTTP Server, який є дуже популярним у розробників веб-додатків через його гнучкість та могутність. Apache підтримує широкий спектр модулів, що розширюють його функціональність, включаючи безпеку, кешування та управління трафіком.

Інший варіант – Nginx, який знаний своєю високою продуктивністю та ефективністю у роботі зі статичними файлами та як засіб балансування навантаження. Nginx широко використовується для великих веб-додатків та служб через його здатність швидко обробляти велику кількість з'єднань.

Ще один популярний варіант – Microsoft Internet Information Services (IIS), який ідеально підходить для додатків, розроблених на .NET та інших технологіях Microsoft. IIS пропонує тісну інтеграцію з Windows, що може бути перевагою у певних розробницьких та продакшн середовищах.

Виберемо для використання Nginx через його переваги для системи бронювання столиків. Nginx відомий своєю високою продуктивністю та ефективністю, особливо при обробці статичного контенту та зменшенні часу завантаження веб-сторінок. Це важливо для системи бронювання, де швидкість відгуку є критичною для задоволення користувачів. Nginx також забезпечує високий рівень масштабованості, що дозволяє системі бронювання столиків ефективно справлятися з ростом кількості користувачів та запитів. Його можливості з балансування навантаження і управління сесіями дозволяють забезпечити стабільність та доступність сервісу, що є важливим для будь-якої комерційної системи.

4.2 Тестування вебзастосунку

Тестування веб-застосунку для бронювання столиків розпочиналося з функціонального тестування, яке мало на меті переконатися, що всі основні функції застосунку працюють згідно з технічними вимогами.

Детальна таблиця 4.2 результатів функціонального тестування для системи бронювання столиків. Таблиця охоплює перевірку основних функцій системи, які включають реєстрацію, авторизацію, бронювання столика, перегляд історії бронювань та управління бронюваннями адміністратором.

Ця таблиця допомагає візуально представити результати функціонального тестування, виділяючи ключові аспекти функціонування системи і ефективності внесених корективів. Загалом, більшість функцій працювали без помилок, а виявлені проблеми були швидко виправлені, забезпечивши високий рівень надійності та користувацької задоволеності системою.

Тестування продуктивності веб-застосунку для бронювання столиків було спрямоване на оцінку здатності системи витримувати велику кількість одночасних користувачів і обробляти їхні запити без істотних затримок. Для цього використовувалися інструменти навантажувального тестування, такі як JMeter, що дозволяють імітувати одночасний доступ багатьох користувачів до системи та вимірювати час відгуку та загальну стабільність (табл. 4.3).

Apache JMeter є міцним інструментом для проведення навантажувального тестування, дозволяючи імітувати велику кількість одночасних користувацьких сесій для веб-аплікацій, баз даних, і систем обміну повідомленнями. Використовуючи JMeter, можна створити сценарії, які відтворюють доступ до веб-сервісів через HTTP або HTTPS, обмін даними через JDBC, взаємодію з JMS, відправлення повідомлень через SMTP, та виконання запитів до LDAP-серверів.

Запис веб-тестів з JMeter починається з активації вбудованого проксі-сервера, який перехоплює запити веб-браузера та автоматично створює тестовий план. Це дозволяє розробникам просто навігувати по веб-сайту, в той час як JMeter реєструє кожну взаємодію, зберігаючи їх у форматі, готовому для повторення тестових сценаріїв.

Для додання реалістичності, JMeter дозволяє параметризувати запити, використовуючи змінні та функції, що забезпечують можливість індивідуалізації даних для кожного віртуального користувача.

Таблиця 4.2 – Функціональне тестування

Функція	Опис тесту	Результат тестування	Проблеми виявлені в процесі	Статус виправлення
Реєстрація	Перевірка можливості створення нового користувацького облікового запису	Пройшла успішно	Без проблем	Не потребує виправлення
Авторизація	Перевірка можливості входу в систему з використанням існуючих облікових даних	Пройшла успішно	Без проблем	Не потребує виправлення
Бронювання столика	Перевірка процесу вибору столика та підтвердження бронювання	Пройшла успішно	Без проблем	Не потребує виправлення
Перегляд історії бронювань	Перевірка можливості перегляду попередніх бронювань користувачем	Виявлено помилки	Некоректне відображення деяких бронювань	Виправлено
Управління бронюваннями адміністратором	Тестування можливостей адміністратора щодо модифікації та видалення бронювань	Пройшла успішно	Без проблем	Не потребує виправлення

Логічні контролери в JMeter управляють виконанням частин тесту на основі заданих умов, дозволяючи вибірково включати або виключати певні частини тестового плану.

Після виконання тестів, JMeter надає детальні звіти про продуктивність, включаючи час відгуку, швидкість передачі даних, і кількість помилок. Ці звіти можуть бути подальше аналізовані та візуалізовані для глибокого розуміння продуктивності системи.

Для масштабних тестувань, JMeter дозволяє виконувати сценарії на кількох машинах або в декількох потоках, що імітують велику кількість користувачів, працюючи одночасно. Ця функція особливо корисна для ідентифікації вузьких місць у великих системах, де потрібно тестувати реальну здатність системи витримувати навантаження.

Таблиця 4.3 - Результати тестування продуктивності

Кількість користувачів	Середній час відгуку (сек)	Максимальний час відгуку (сек)	Примітки
100	0.5	1	Без проблем
500	1	1.5	Легкі затримки при піку
1000	2	4	Прийнятні затримки
1500	5	10	Значні затримки
2000	10	20	Збої в обробці запитів

Ця таблиця (табл. 4.3) відображає, як система реагує на різні рівні навантаження. З результатів видно, що до 1000 користувачів система працює досить стабільно, але при подальшому збільшенні навантаження з'являються серйозні затримки. Це вказує на необхідність подальшої оптимізації системи або зміцнення серверної інфраструктури для підтримки більшої кількості одночасних користувачів.

Методи тестування:

- стрес-тестування. Визначення меж продуктивності системи шляхом імітації екстремальних умов використання, щоб з'ясувати, при якому навантаженні система починає давати збій або значно сповільнюватися;
- тестування збільшення навантаження. Поступове збільшення кількості користувачів або запитів до системи для визначення моменту, коли продуктивність системи починає погіршуватися;
- моніторинг ресурсів. Спостереження за використанням апаратних ресурсів, таких як CPU, пам'ять, дисковий простір та мережеве навантаження, для виявлення вузьких місць у продуктивності.

4.3 Висновки за розділом 4

У цьому розділі було розглянуто ключові аспекти розробки та тестування системи бронювання столиків.

Після порівняння різних варіантів (Apache HTTP Server, Nginx, Microsoft IIS) було обрано Nginx через його високу продуктивність, ефективність обробки статичного контенту та здатність до масштабування. Це рішення оптимальне для забезпечення швидкої роботи системи бронювання та її стабільності під навантаженням.

Проведено комплексне тестування основних функцій системи, включаючи реєстрацію, авторизацію, бронювання столиків та управління бронюваннями. Більшість функцій пройшли тестування успішно, а виявлені проблеми були оперативно виправлені.

Використовуючи інструмент Apache JMeter, було проведено навантажувальне тестування системи. Результати показали, що система стабільно працює при навантаженні до 1000 одночасних користувачів, але при більшому навантаженні виникають значні затримки та збої.

Застосовано різноманітні методи, включаючи стрес-тестування, тестування збільшення навантаження та моніторинг ресурсів, що дозволило всебічно оцінити продуктивність системи.

Загалом, розроблена система демонструє хорошу функціональність та прийнятну продуктивність для середнього рівня навантаження. Однак результати тестування вказують на необхідність подальшої оптимізації та посилення серверної інфраструктури для забезпечення стабільної роботи при високих навантаженнях. Це важливо для майбутнього масштабування системи та забезпечення якісного обслуговування зростаючої кількості користувачів.

ВИСНОВКИ

У рамках дипломного проєкту було розроблено комп'ютерну систему для бронювання місць у кав'ярнях ТОВ «Терра». Проєкт включав детальний аналіз ринку послуг кав'ярень в Україні, вивчення існуючих систем бронювання та їх впливу на обслуговування клієнтів.

Основні результати проєкту:

- проведено аналіз сучасного стану ринку послуг кав'ярень, виявлено тенденції та особливості, що впливають на розвиток галузі;
- розроблено специфікацію вимог до програмного забезпечення (SRS), яка враховує потреби як користувачів, так і адміністраторів системи;
- спроектовано архітектуру комп'ютерної системи на основі клієнт-серверної моделі, що забезпечує масштабованість, безпеку та ефективність;
- створено структуру бази даних, яка включає п'ять ключових таблиць для ефективного управління даними про користувачів, резервації, столики, заклади та адміністраторів;
- обрано технологічний стек, що включає JavaScript з React для фронтенду, Node.js для бекенду та MySQL для бази даних;
- розроблено програмну частину системи з інтуїтивно зрозумілим інтерфейсом користувача, що включає функціонал для пошуку закладів, бронювання столиків, перегляду історії бронювань та панель адміністратора;
- надано рекомендації щодо апаратної частини системи для забезпечення високої продуктивності та надійності.

Розроблена комп'ютерна система відповідає сучасним вимогам ринку, забезпечує зручність використання для клієнтів та ефективне управління для адміністраторів закладів. Вона має потенціал для подальшого розвитку та масштабування, що дозволить ТОВ «Терра» підвищити якість обслуговування клієнтів та оптимізувати бізнес-процеси у сфері кав'ярень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Saputra S. A., Setiawan E. Web-based cafe search app using bootstrap in the city of surabaya. Jurnal simantec. 2022. Vol. 10, no. 2. P. 57–66. URL: <https://doi.org/10.21107/simantec.v10i2.13726>
2. Baker M. Cross-Site Requests. Secure Web Application Development. Berkeley, CA, 2022. P. 229–270. URL: https://doi.org/10.1007/978-1-4842-8596-1_8
3. Updated on the Communications Society's Web Site. IEEE Communications Magazine. 2022. Vol. 60, no. 6. P. 16. URL: <https://doi.org/10.1109/mcom.2022.9800166>
4. Lengstorf J., Prettyman S., Hansen T. B. PHP 8 for Absolute Beginners: Basic Web Site and Web Application Development. Apress L. P., 2022.
5. Secure digital authentication system for shopping web site using qr code. International research journal of modernization in engineering technology and science. 2022. URL: <https://doi.org/10.56726/irjmets31364>
- 6 AB M. MySQL Administrator's Guide. Upper Saddle River : Pearson Education, 2005. 430 p.
- 7 Moutaouakkil A. MVC Frameworks Modernization Approach. International Journal of Advanced Computer Science and Applications. 2019. Vol. 10, no. 10.
- 8 Automatic Code Generation of MVC Web Applications. Computers. 2020. Vol. 9, no. 3. P. 56.
- 9 Carneiro N. Web 4. Independently published, 2019. 64 p.
- 10 Collins M. J. MVC Web Applications. Pro HTML5 with Visual Studio 2015. Berkeley, CA, 2015. P. 43–78.
- 11 Consortium W. W. W. The Extensible Hypertext Markup Language: A Reformulation of Html 4 in Xml 1.0. Iuniverse Inc, 2000. 108 p.
- 12 Research A., i4cp, Ney J. Web 3. 0. American Society for Training & Development, 2010.

ДОДАТОК А

Текст програми

```

import React from 'react';
import CafeList from './CafeList';
import './App.css';

function App() {
  return (
    <div className="app">
      <header className="app-header">
        <h1>Знайдіть стіл</h1>
        <input      type="text"      placeholder="Пошук      кафе"
className="search-input" />
        <div className="filters">
          <select>
            <option>Будь-яка дата</option>
          </select>
          <select>
            <option>Будь-який час</option>
          </select>
          <select>
            <option>Для 2</option>
          </select>
        </div>
      </header>
      <CafeList />
    </div>
  );
}

export default App;

import React from 'react';
import './CafeList.css';

```

```

const cafes = [
  {
    id: 1,
    name: 'Емілі',
    cuisine: 'Італійська',
    price: '$$',
    location: 'Вест-Віледж',
    image: 'path-to-image1.jpg',
  },
  {
    id: 2,
    name: 'Бокерія',
    cuisine: 'Іспанська',
    price: '$$',
    location: 'Флетайрон',
    image: 'path-to-image2.jpg',
  },
  {
    id: 3,
    name: 'Кафе Могадор',
    cuisine: 'Середземноморська',
    price: '$$',
    location: 'Іст-Віледж',
    image: 'path-to-image3.jpg',
  },
];

const CafeList = () => {
  return (
    <div className="cafe-list">
      <h2>Популярні</h2>
      {cafes.map(cafe => (
        <div className="cafe-card" key={cafe.id}>
          <img src={cafe.image} alt={cafe.name} />
          <div className="cafe-info">

```

```

        <h3>{cafe.name}</h3>
        <p>{cafe.cuisine} - {cafe.price}</p>
        <p>{cafe.location}</p>
      </div>
    </div>
  )})
  <h2>Усі кафе</h2>
  { /* Додати більше кафе з рейтингами тут */ }
</div>

);
};

export default CafeList;

.app {
  font-family: Arial, sans-serif;
  padding: 20px;
}

.app-header {
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 10px;
}

.search-input {
  padding: 10px;
  width: 80%;
  max-width: 400px;
  font-size: 16px;
}

.filters {
  display: flex;
  gap: 10px;

```

```
}

.filters select {
  padding: 10px;
  font-size: 16px;
}

.cafe-list {
  padding: 20px;
}

.cafe-card {
  display: flex;
  gap: 10px;
  margin-bottom: 20px;
}

.cafe-card img {
  width: 100px;
  height: 100px;
  object-fit: cover;
}

.cafe-info {
  display: flex;
  flex-direction: column;
  justify-content: center;
}

.cafe-info h3 {
  margin: 0;
  font-size: 20px;
}

.cafe-info p {
  margin: 5px 0;
```

```

}

import React, { useState } from 'react';
import './DateSelector.css';

const dates = [
  { id: 1, day: 'Пн, 21 лют', time: '18:00' },
  { id: 2, day: 'Вт, 22 лют', time: '19:00' },
  { id: 3, day: 'Ср, 23 лют', time: '20:00' },
];

const guests = 3;

const tables = [
  {
    id: 1,
    zone: 'Стандартна зона',
    availability: 'Доступно',
    description: 'Столик для 4 - 20:00',
    image: 'path-to-table-image.jpg',
  },
];

const DateSelector = () => {
  const [selectedDate, setSelectedDate] = useState(null);

  const handleDateSelect = (date) => {
    setSelectedDate(date);
  };

  return (
    <div className="date-selector">
      <h2>Дата та час</h2>
      <div className="date-list">
        {dates.map((date) => (
          <div

```

```

        key={date.id}
        className={`date-item ${selectedDate === date ?
'selected' : ''}`}
        onClick={() => handleDateSelect(date)}
      >
        <p>{date.day}</p>
        <p>{date.time}</p>
      </div>
    )})
  </div>
  <div className="guest-info">
    <p>{guests} гостя</p>
    <p>Ескане</p>
  </div>
  <h2>Столики</h2>
  {tables.map((table) => (
    <div className="table-card" key={table.id}>
      <img src={table.image} alt={table.zone} />
      <div className="table-info">
        <h3>{table.zone}</h3>
        <p>{table.availability}</p>
        <p>{table.description}</p>
      </div>
      <button className="book-button">Замовити</button>
    </div>
  ))}
</div>
);
};

export default DateSelector;

.date-selector {
  padding: 20px;
}

```

```
.date-list {
  display: flex;
  gap: 10px;
  margin-bottom: 20px;
}

.date-item {
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  cursor: pointer;
  text-align: center;
}

.date-item.selected {
  background-color: #d0e2ff;
  border-color: #5b9aff;
}

.guest-info {
  margin-bottom: 20px;
}

.table-card {
  display: flex;
  flex-direction: column;
  gap: 10px;
  margin-bottom: 20px;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.table-card img {
  width: 100%;
  height: auto;
}
```

```

    border-radius: 5px;
  }

.table-info {
  display: flex;
  flex-direction: column;
  gap: 5px;
}

.book-button {
  padding: 10px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  text-align: center;
}

import React from 'react';
import CafeList from './CafeList';
import DateSelector from './DateSelector';
import './App.css';

function App() {
  return (
    <div className="app">
      <header className="app-header">
        <h1>Знайдіть стіл</h1>
        <input type="text" placeholder="Пошук кафе"
className="search-input" />
        <div className="filters">
          <select>
            <option>Будь-яка дата</option>
          </select>
          <select>

```

```

        <option>Будь-який час</option>
      </select>
      <select>
        <option>Для 2</option>
      </select>
    </div>
  </header>
  <CafeList />
  <DateSelector />
</div>
);
}

```

```
export default App;
```

```
import React from 'react';
import './Reservations.css';
```

```
const todayReservations = [
  {
    id: 1,
    name: 'Bistro 8',
    time: '12:30 PM',
    details: 'Столик на 2 особи',
    image: 'path-to-image1.jpg',
  },
  {
    id: 2,
    name: 'The Grill',
    time: '7:00 PM',
    details: 'Столик на 4 особи',
    image: 'path-to-image2.jpg',
  },
  {
    id: 3,
    name: 'Hakkasan',

```

```

        time: '8:00 PM',
        details: 'Столик на 6 особи',
        image: 'path-to-image3.jpg',
      },
    ];

const weekReservations = [
  {
    id: 1,
    name: 'The Grill',
    time: '7:00 PM, Пт',
    details: 'Столик на 4 особи',
    image: 'path-to-image4.jpg',
  },
  {
    id: 2,
    name: 'Bistro 8',
    time: '12:30 PM, Сб',
    details: 'Столик на 2 особи',
    image: 'path-to-image5.jpg',
  },
];

const Reservations = () => {
  return (
    <div className="reservations">
      <header className="reservations-header">
        <h2>Оновлення</h2>
      </header>
      <section className="reservations-section">
        <h3>Сьогодні</h3>
        {todayReservations.map(reservation => (
          <div
            className="reservation-card"
            key={reservation.id}>
            <img
              src={reservation.image}
              alt={reservation.name} />

```

```

        <div className="reservation-info">
            <h4>{reservation.name}</h4>
            <p>{reservation.time}</p>
            <p>{reservation.details}</p>
        </div>
    </div>
    )))
</section>
<section className="reservations-section">
    <h3>Цього тижня</h3>
    {weekReservations.map(reservation => (
        <div
            className="reservation-card"
            key={reservation.id}>
                <img
                    src={reservation.image}
                    alt={reservation.name} />
                <div className="reservation-info">
                    <h4>{reservation.name}</h4>
                    <p>{reservation.time}</p>
                    <p>{reservation.details}</p>
                </div>
            </div>
        )))
    </section>
<footer className="reservations-footer">
    <nav>
        <ul>
            <li>Головна</li>
            <li className="active">Бронювання</li>
            <li>Звіти</li>
            <li>Більше</li>
        </ul>
    </nav>
</footer>
</div>
);
};

```

```
export default Reservations;

.reservations {
  padding: 20px;
  font-family: Arial, sans-serif;
}

.reservations-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-bottom: 20px;
}

.reservations-header h2 {
  margin: 0;
}

.reservations-section {
  margin-bottom: 20px;
}

.reservations-section h3 {
  margin-bottom: 10px;
}

.reservation-card {
  display: flex;
  align-items: center;
  margin-bottom: 10px;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
}
```

```
.reservation-card img {  
  width: 50px;  
  height: 50px;  
  object-fit: cover;  
  border-radius: 5px;  
  margin-right: 10px;  
}  
  
.reservation-info {  
  display: flex;  
  flex-direction: column;  
}  
  
.reservation-info h4 {  
  margin: 0;  
  font-size: 16px;  
}  
  
.reservation-info p {  
  margin: 5px 0;  
  font-size: 14px;  
}  
  
.reservations-footer {  
  position: fixed;  
  bottom: 0;  
  left: 0;  
  width: 100%;  
  background-color: white;  
  border-top: 1px solid #ccc;  
}  
  
.reservations-footer nav ul {  
  display: flex;  
  justify-content: space-around;  
  padding: 10px 0;
```

```

    margin: 0;
    list-style: none;
}

.reservations-footer nav ul li {
    font-size: 14px;
}

.reservations-footer nav ul li.active {
    font-weight: bold;
    color: blue;
}

import React from 'react';
import CafeList from './CafeList';
import DateSelector from './DateSelector';
import Reservations from './Reservations';
import './App.css';

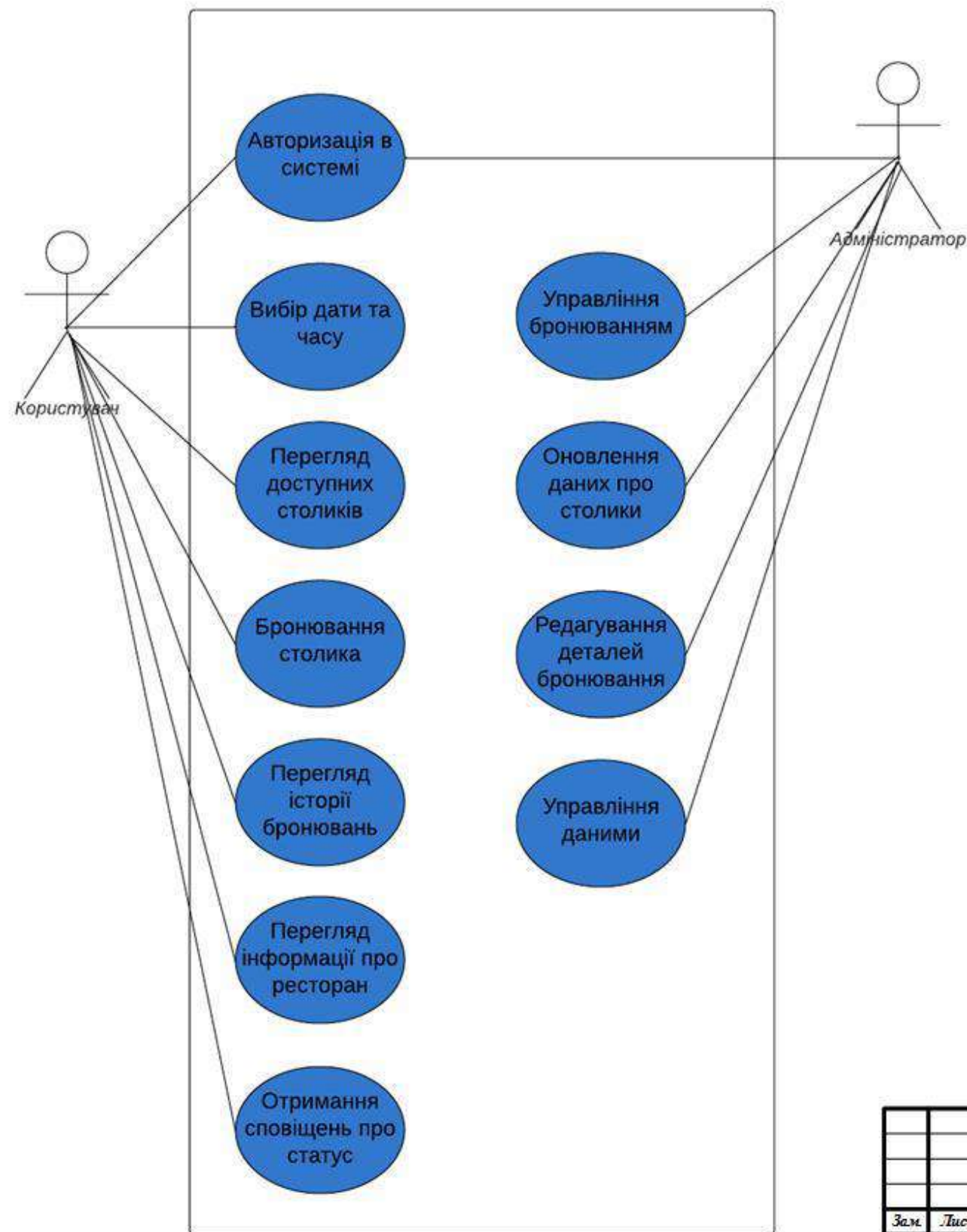
function App() {
    return (
        <div className="app">
            <header className="app-header">
                <h1>Знайдіть стил</h1>
                <input      type="text"      placeholder="Пошук      кафе"
className="search-input" />
            <div className="filters">
                <select>
                    <option>Будь-яка дата</option>
                </select>
                <select>
                    <option>Будь-який час</option>
                </select>
                <select>
                    <option>Для 2</option>
                </select>
            </div>
        </div>
    );
}

```

```
        </div>
      </header>
      <CafeList />
      <DateSelector />
      <Reservations />
    </div>
  );
}

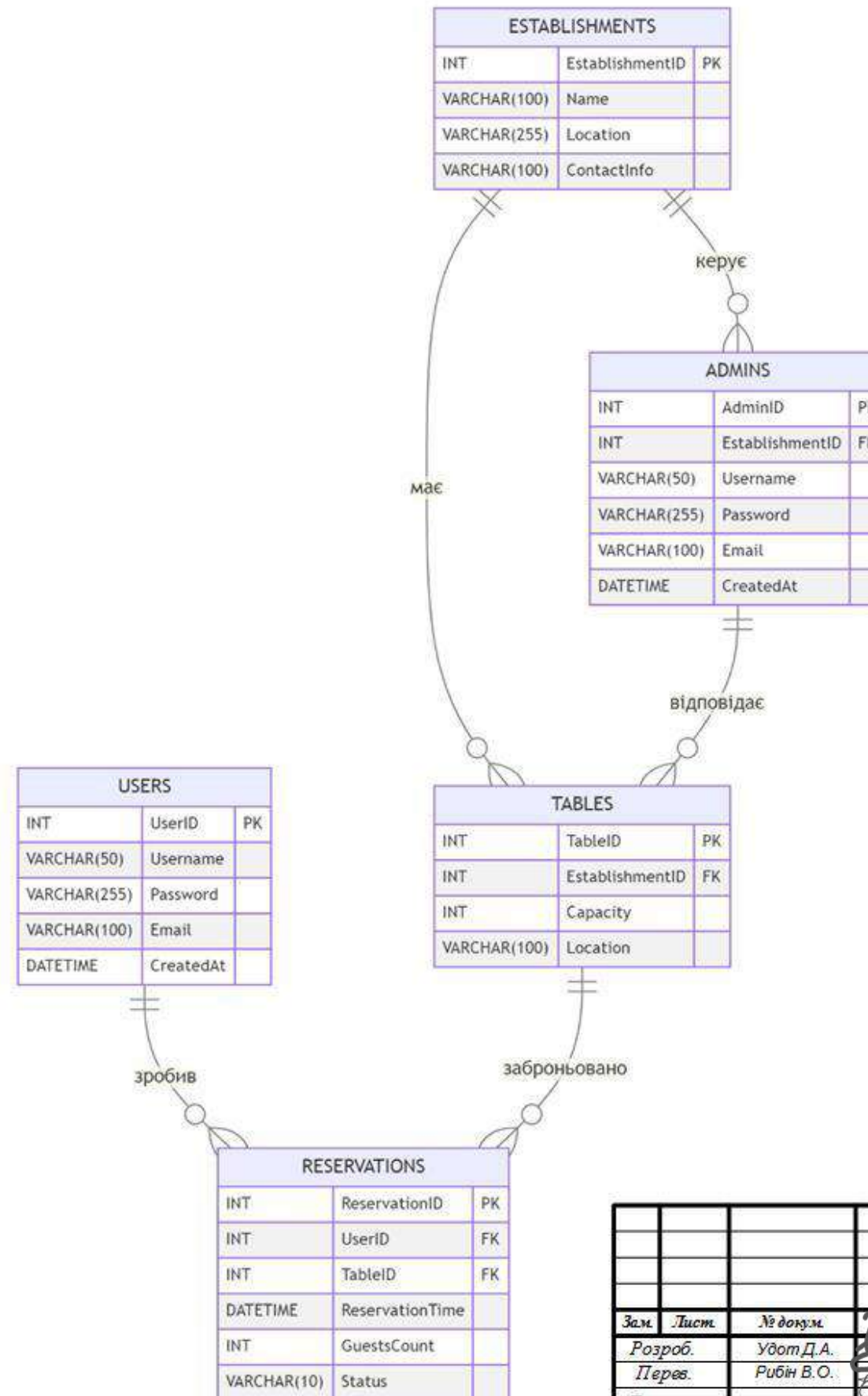
export default App;
```

Діаграма варіантів використання Комп'ютерної системи ТОВ «Терра»



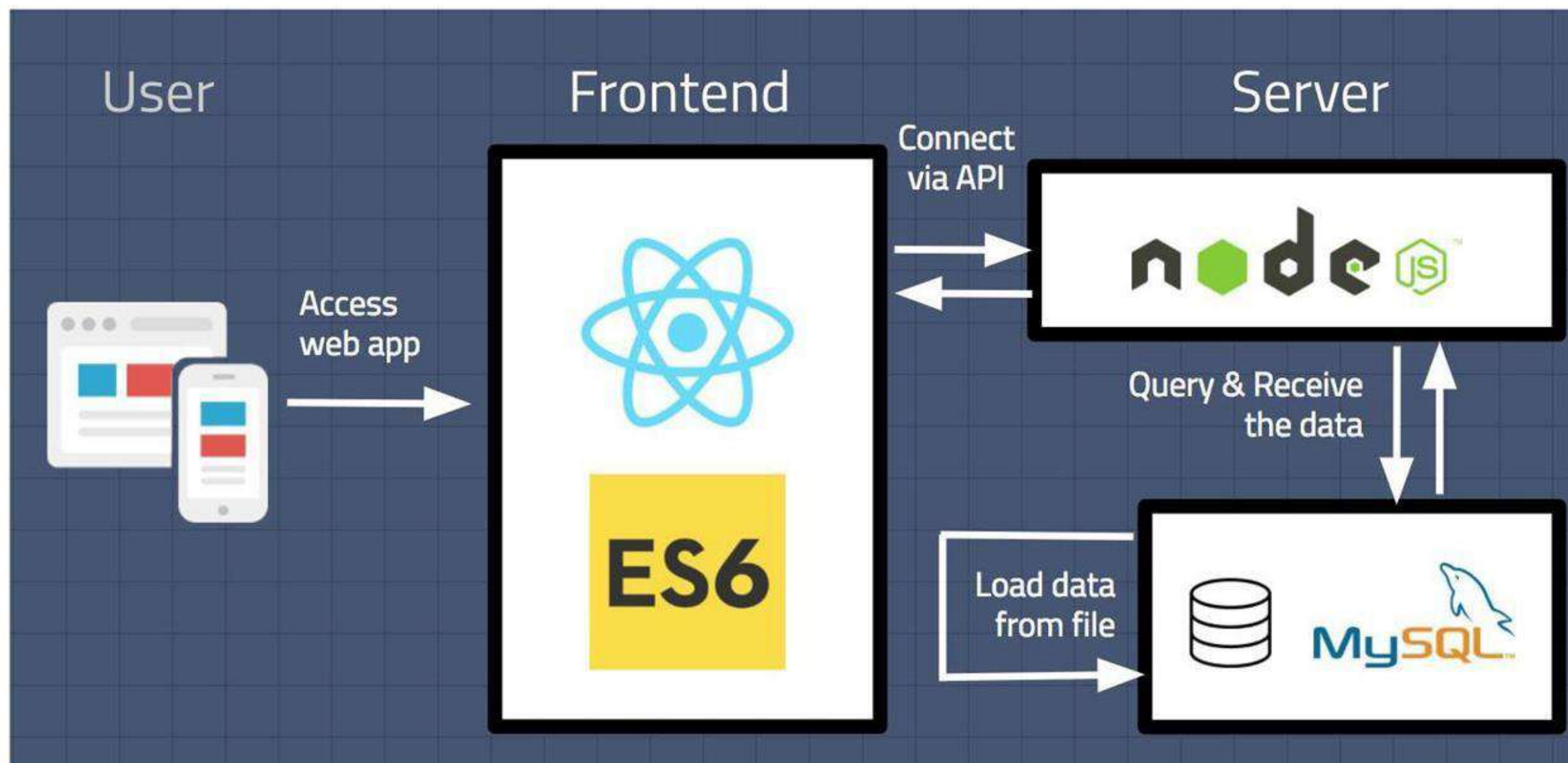
					13.02070849.00051 ПЛ1			
Зам.	Лист	№ докум.	Підп.	Дата	Розробка комп'ютерної системи ТОВ «Терра» Діаграма варіантів використання системи	Лист	Маса	Масшт.
Розроб.		Удов Д.А.						
Перев.		Рибін В.О.						
Т. контр.						Лист 1		Листів 1
Н. контр.		Пальська О.В.				НУ «Запорізька політехніка» КНТ-520		
Затв.		Кудерметов Р.К.						

Схема бази даних



					13.02070849.00051 ПЛІ2			
Зам.	Лист.	№ докум.	Підп.	Дата	Розробка комп'ютерної системи ТОВ «Терра» Схема бази даних	Лист.	Маса	Масштаб.
Розроб.	Удот Д.А.	Рибін В.О.						
Перев.								
Т.контр.								
Н.контр.	Пальська О.В.					Лист 1	Листів 1	
Затв.	Кудерметов Р.Р.					НУ «Запорізька політехніка» KHT-520		

Технологічний стек



					13.02070849.00051 ПЛЗ			
Зам.	Лист.	№ докум.	Підп.	Дата	Розробка комп'ютерної системи ТОВ «Терра» Технологічний стек	Лист.	Маса	Масшт.
Розроб.		Удовт Д.А.	<i>[Signature]</i>					
Перев.		Рибін В.О.	<i>[Signature]</i>					
Т.контр.						Лист 1		Листів 1
Н.контр.		Пальська О.В.	<i>[Signature]</i>			НУ «Запорізька політехніка»		
Затв.		Кудерметов Р.К.	<i>[Signature]</i>			КНТ-520		

Інтерфейс вебзастосунку



Рисунок 1 – Початковий екран

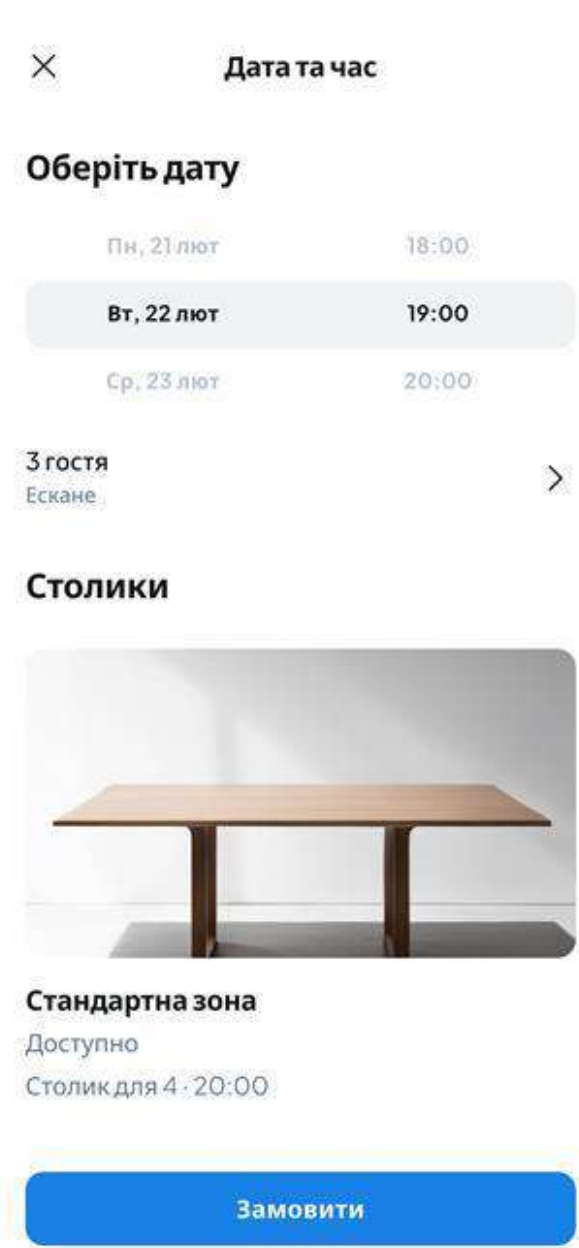


Рисунок 2 – Бронювання столика

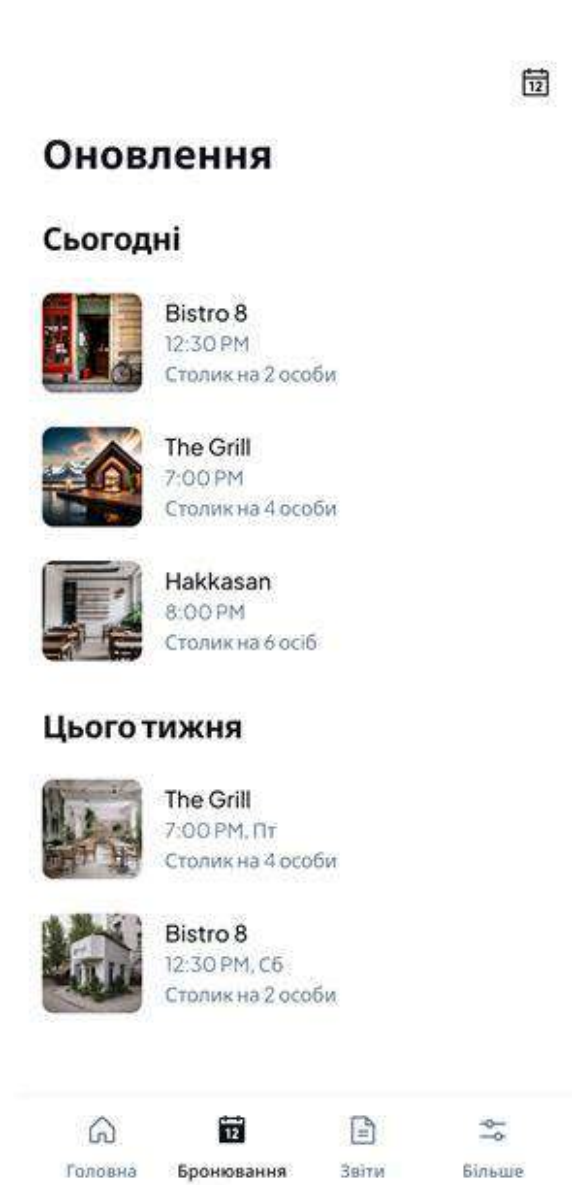


Рисунок 3 – Історія бронювань



Рисунок 4 – Панель адміністратора

					13.02070849.00051 ПЛ4			
					Розробка комп'ютерної системи ТОВ «Терра»			
					Інтерфейс вебзастосунку			
Зам.	Лист.	№ докум.	Підп.	Дата				
Розроб.	Лист.	Удот Д.А.	Підп.	Дата				
Перев.	Лист.	Рибін В.О.	Підп.	Дата				
Т.контр.	Лист.		Підп.	Дата				
Н.контр.	Лист.	Пальська О.В.	Підп.	Дата				
Затв.	Лист.	Кудряшова Р.К.	Підп.	Дата				
					Лист 1			
					Листів 1			
					НУ «Запорізька політехніка»			
					КНТ-520			