

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 126 «Інформаційні системи та технології»
(освітня програма «Системи бізнес-аналітики»)
на тему:

**«Автоматизована система управління закладом
харчування»**

Виконав студент групи ІСТ-20-1
МІЩУК Вероніка Сергіївна

Керівник роботи:
МАРЧУК Дмитро Костянтинович

Рецензент:
ОРИНЧАК Андрій Іванович

Житомир – 2024

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

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

Марина ГРАФ

«25» січня 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 126 «Інформаційні системи та технології»
(освітня програма «Системи бізнес-аналітики»)

Здобувач вищої освіти: **МІЩУК Вероніка Сергіївна**

Керівник роботи: **МАРЧУК Дмитро Костянтинович**

Тема роботи: «**Автоматизована система управління закладом харчування**»,
затверджена Наказом закладу вищої освіти від «25» січня 2024 р., №24/с

Термін здачі закінченої роботи: «05» червня 2024 р.

Вихідні дані роботи (зазначається предмет і об'єкт дослідження): об'єктом дослідження є процес управління закладом харчування. Предметом дослідження є використання інформаційних технологій автоматизованої системи управління закладом харчування.

Консультанти випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв
1	МАРЧУК Д.К.	05.02.2024	07.02.2024
2	МАРЧУК Д.К.	12.03.2024	15.03.2024
3	МАРЧУК Д.К.	08.04.2024	30.04.2024

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	5 лютого 2024 – 12 березня 2024	Виконано
2	Проектування структури системи	12 березня 2024 – 23 березня 2024	Виконано
3	Написання програмного коду	25 березня 2024 – 22 квітня 2024	Виконано
4	Тестування системи	27 квітня 2024 – 10 травня 2024	Виконано
5	Оформлення пояснювальної записки	15 травня 2024 – 1 червня 2024	Виконано
6	Попередній захист	5 червня 2024	Виконано

Здобувач вищої освіти **Вероніка МІЩУК**
Керівник **Дмитро МАРЧУК**

РЕФЕРАТ

Основний зміст роботи викладено на 68 сторінках тексту, містить 17 таблиць, 73 ілюстрації, 17 сторінок додатків, налічує 20 найменувань літературних джерел.

Метою кваліфікаційної роботи є створення автоматизованої системи управління закладом харчування із використанням сучасних технологій веброзробки, що включає управління замовленнями, обліком та управління персоналом.

У роботі запропоновано розробити інформаційну систему, яка забезпечує ефективне управління замовленнями, обліком та персоналом закладу харчування, що дозволяє оптимізувати внутрішні процеси та покращити обслуговування клієнтів.

КЛЮЧОВІ СЛОВА: АВТОМАТИЗАЦІЯ, УПРАВЛІННЯ ЗАКЛАДОМ ХАРЧУВАННЯ, ІНФОРМАЦІЙНА СИСТЕМА, WEB.

					ІСТ.КР.Б-126-24-ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата	Автоматизована система управління закладом харчування Пояснювальна записка	Літ.	Арк.	Аркуші
Розроб.		В.С.МІЩУК					2	87
Керівник		Д.К.МАРЧУК				Житомирська політехніка, група ІСТ-20-1		
Рецензент		А.І.ОРИНЧАК						
Н. контр.								
Зав. каф.		М.С.ГРАФ						

ABSTRACT

The main content of the work is presented on 68 pages of text, including 17 tables, 73 illustrations, 17 pages of appendices, and comprises 20 references.

The aim of the qualification work is to develop an automated management system for a catering establishment using modern web development technologies, which includes order management, accounting, and personnel management.

The work proposes to develop an information system that ensures efficient management of orders, accounting, and personnel of the catering establishment, allowing to optimize internal processes and improve customer service.

KEYWORDS: AUTOMATION, CATERING ESTABLISHMENT MANAGEMENT, INFORMATION SYSTEM, WEB.

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						3
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	5
ВСТУП	6
РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ЗАКЛАДОМ ХАРЧУВАННЯ.....	8
1.1 АНАЛІЗ ПРОЄКТУ ТА ПОСТАНОВКА ЗАДАЧІ	8
1.2 АНАЛІЗ АНАЛОГІВ ПРОГРАМНОГО ПРОДУКТУ	9
1.3 ВИБІР АРХІТЕКТУРИ АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ЗАКЛАДОМ ХАРЧУВАННЯ.....	13
1.4 ОБҐРУНТУВАННЯ ВИБОРУ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ТА ВИМОГИ ДО АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ.....	16
ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ	19
РОЗДІЛ 2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	20
2.1 ВИЗНАЧЕННЯ ВАРІАНТІВ ВИКОРИСТАННЯ ТА СТРУКТУРИ СИСТЕМИ.....	20
2.2 РОЗРОБКА БАЗИ ДАНИХ СИСТЕМИ.....	25
2.3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ АЛГОРИТМІВ РОБОТИ СИСТЕМИ.....	32
2.4 РЕАЛІЗАЦІЯ ФУНКЦІОНАЛУ СИСТЕМИ	34
ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ	39
РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З АВТОМАТИЗОВАНОЮ СИСТЕМОЮ УПРАВЛІННЯ ЗАКЛАДОМ ХАРЧУВАННЯ	40
3.1 ПОРЯДОК ВСТАНОВЛЕННЯ ТА НАЛАШТУВАННЯ ПАРАМЕТРІВ СИСТЕМИ....	40
3.2 СТРУКТУРА ІНТЕРФЕЙСУ СИСТЕМИ	41
3.3 ТЕСТУВАННЯ СИСТЕМИ	61
ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ	63
ВИСНОВКИ.....	64
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	66
ДОДАТКИ.....	68

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

ІТ – інформаційні технології

HTML – мова розмітки гіпертексту (HyperText Markup Language)

CSS – каскадні таблиці стилів (Cascading Style Sheets)

СУБД – система управління базами даних

CRUD – створення, читання, оновлення, видалення (create, read, update, delete)

EF – Entity Framework

SQL – мова структурованих запитів (structured query language)

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. У світі, де інформаційні технології стають необхідним елементом у всіх сферах життя, важливо розвивати та впроваджувати нові інноваційні підходи в управлінні та роботі підприємств. Зокрема, компанії, які займаються сферою ресторанного бізнесу, потребують ефективних інформаційних систем для оптимізації процесів управління та підвищення якості обслуговування клієнтів. Тому розробка автоматизованих систем управління для закладів харчування є актуальною та перспективною задачею.

Метою кваліфікаційної роботи є розробка та аналіз інформаційної системи для управління закладом харчування. Основним завданням є створення програмного продукту, який забезпечить ефективне управління закладом, зокрема, контроль над складом і продажами, оптимізація роботи персоналу та забезпечення зручного і швидкого обслуговування клієнтів.

Постановлена мета дослідження обумовлює необхідність вирішення таких завдань: вивчення потреб та вимог клієнтів до автоматизованої системи управління закладом харчування; аналіз існуючих програмних продуктів для визначення їх переваг та недоліків; вибір оптимальної архітектури системи, що забезпечить ефективне управління та зручне обслуговування клієнтів.

Об'єктом дослідження є процес управління закладом харчування.

Предметом дослідження є використання інформаційних технологій для створення автоматизованої системи управління цим закладом.

Методи дослідження. Для досягнення поставлених цілей використовувалися різні методи. Зокрема, було проведено аналіз вимог користувачів і бізнес-процесів, порівняльний аналіз існуючих рішень, проектування архітектури системи, включаючи створення діаграм використання та класів, а також розробка бази даних. Крім того,

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		6

здійснювалося тестування функціоналу системи для забезпечення її надійності та зручності використання.

Практичне значення полягає в тому, що результати роботи можуть бути використані:

- для підвищення ефективності управління закладом харчування;
- для зменшення витрат на управління та підвищення прибутковості закладу;
- для забезпечення швидкого та якісного обслуговування клієнтів.

За темою кваліфікаційної роботи було опубліковано 3 тези:

1. Проєктування та використання бази даних у сфері обслуговування [1].

2. Огляд сучасних інструментів для розробки веб-орієнтованих систем управління закладами харчування на базі PHP з використанням фреймворку SYMFONY [2].

3. Оптимізація розробки програмного забезпечення у сфері управління закладом харчування за допомогою систем контролю версій [3].

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		7

РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ЗАКЛАДОМ ХАРЧУВАННЯ

1.1 Аналіз проєкту та постановка задачі

Постановка задачі полягає у створенні автоматизованої системи управління закладом харчування для підвищення ефективності роботи, якості обслуговування клієнтів і точності фінансового обліку. Заклади харчування часто стикаються з проблемами, такими як складність управління запасами, затримки та помилки при прийомі замовлень, а також складності у веденні фінансового обліку. Це призводить до зниження ефективності та задоволеності клієнтів.

Мета проєкту полягає в автоматизації ключових операційних процесів, включаючи прийом та обробку замовлень, управління запасами, фінансовий облік. Система забезпечить точне відстеження залишків продуктів, планування закупівель, обробку замовлень та управління фінансами.

На етапі аналізу вимог необхідно дослідити потреби користувачів, вивчити поточні процеси та визначити зони для автоматизації. Проєктування системи включає розробку гнучкої та масштабованої архітектури, моделювання бази даних, розробку користувацького інтерфейсу.

Розробка основного функціоналу охоплює модулі для управління замовленнями, запасами та обслуговуванням клієнтів[4]. Впровадження автоматизованої системи дозволить підвищити ефективність операційних процесів, покращити якість обслуговування та забезпечити прозорість фінансового обліку, що сприятиме збільшенню задоволеності клієнтів і покращенню фінансових показників закладу.

Для досягнення мети проєкту, необхідно вирішити наступні задачі:

1. Збір і аналіз вимог[5,6]:

- визначення потреб та очікувань користувачів системи (адміністраторів, офіціантів, клієнтів).

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		8

- вивчення та уточнення функціональних та нефункціональних вимог до системи.

2. Проектування системи:

- розроблення концепції архітектури системи, включаючи вибір технологічного стеку та створення структури.

- визначення логічної та фізичної моделі бази даних, включаючи таблиці для замовлень, інвентаризації, меню та персоналу.

3. Реалізація функціоналу:

- розроблення базових модулів для управління замовленнями, столами, меню та резерваціями.

- впровадження функцій звітності та аналітики для покращення прийняття управлінських рішень.

4. Тестування та впровадження:

- проведення тестування системи на різних етапах розробки.

- підготовка до впровадження: налаштування середовища.

1.2 Аналіз аналогів програмного продукту

Автоматизовані системи управління закладом харчування є важливою частиною галузі ресторанного бізнесу. Для розробки високоефективної та конкурентоздатної системи важливо вивчити та проаналізувати аналоги, що існують на ринку. У даному розділі буде розглянуто три провідних продукти – Toast, Square for Restaurants та Revel Systems. Кожен з них має свої особливості та можливості, які слід врахувати при створенні нової автоматизованої системи.

Toast - це платформа для автоматизації ресторанного бізнесу, яка включає в себе замовлення столиків, облік запасів, облік робочого часу персоналу та інші функції. Програмне забезпечення Toast інтегрується з POS-терміналами та мобільними пристроями для ефективного управління різними аспектами закладу харчування.

Переваги:

- Широкий функціонал, який охоплює всі аспекти ресторанного бізнесу.
- Зручний інтерфейс та можливість налаштування під конкретні потреби закладу.
- Інтеграція з різними платіжними системами та банківськими терміналами.

Недоліки:

- Вартість підписки може бути високою для деяких закладів.
- Залежність від інтернет-з'єднання для деяких функцій.

На рисунку 1.1 показується інтерфейс Toast.

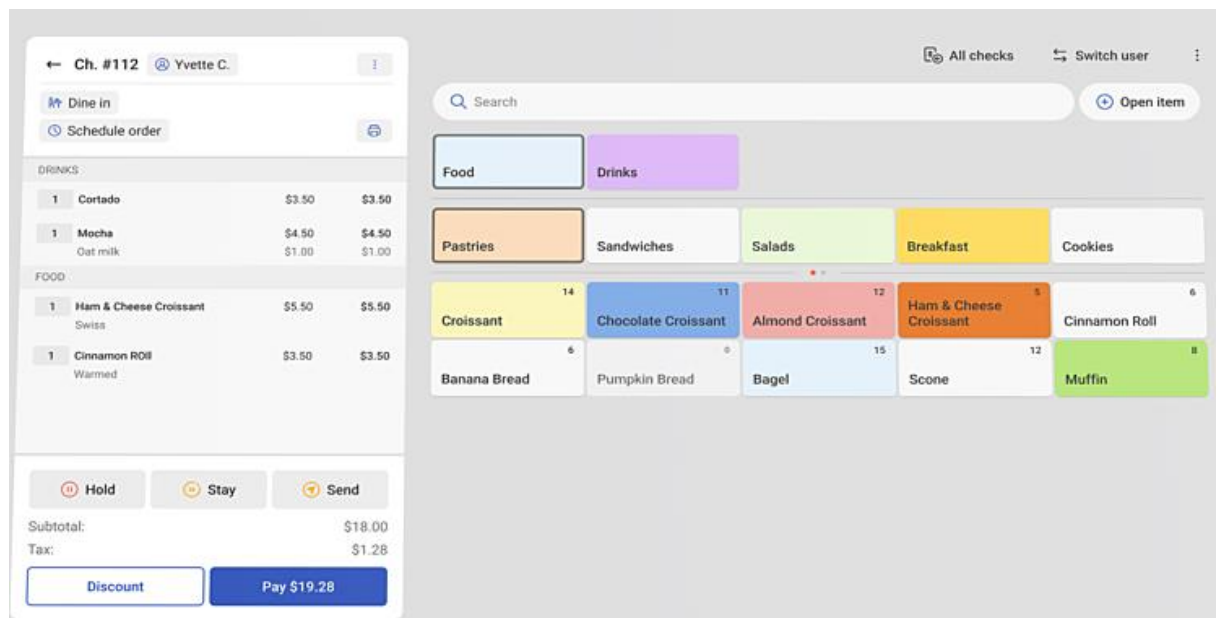


Рисунок 1.1 - Інтерфейс Toast

Square for Restaurants - це програмне забезпечення для управління рестораном, яке пропонує рішення для прийому замовлень, обліку запасів, ведення клієнтської бази та інших аспектів ресторанного бізнесу. Square надає послуги із платіжних терміналів, що може бути інтегровано з програмним забезпеченням.

Переваги:

- Простий та інтуїтивний інтерфейс, що полегшує використання для персоналу.
- Широкі можливості інтеграції з іншими продуктами Square, такими як платіжні термінали.

Недоліки:

- Обмежена функціональність порівняно з деякими конкурентами.
- Вартість деяких додаткових сервісів та обладнання.

На рисунку 1.2 показується інтерфейс Square for Restaurants.

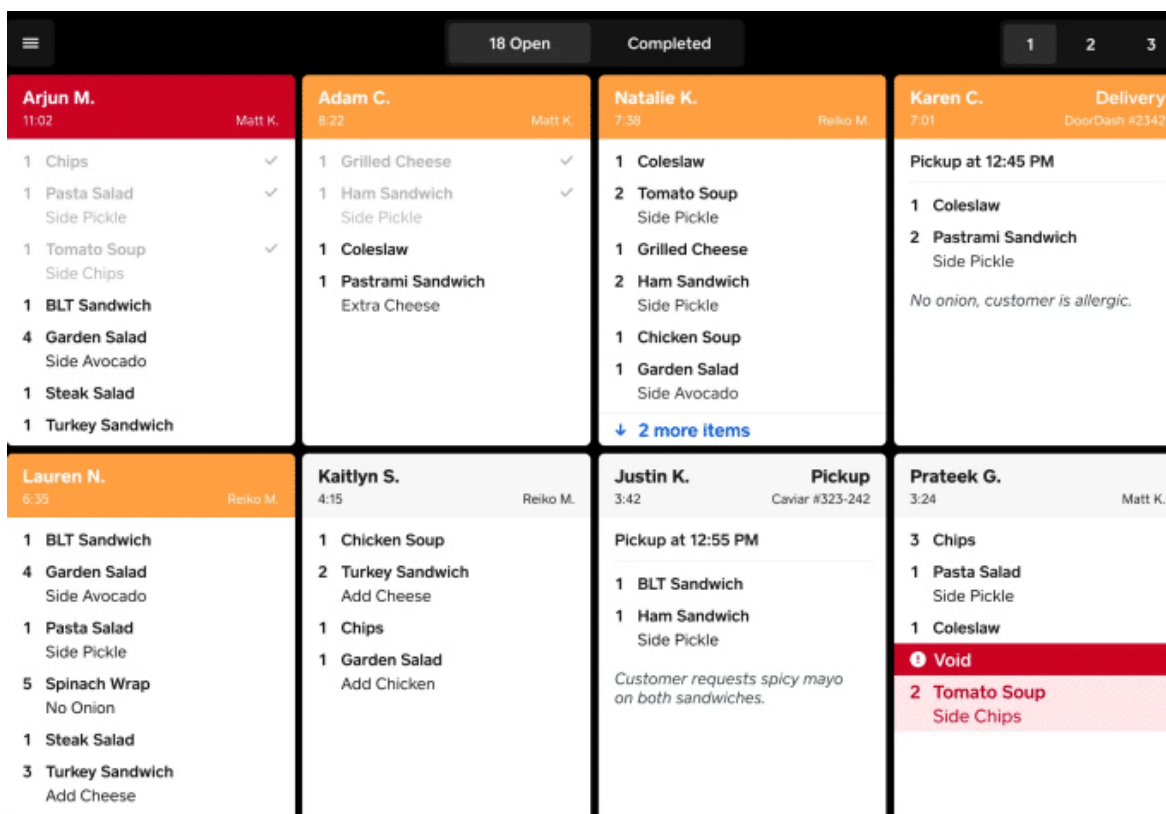


Рисунок 1.2 - Інтерфейс Square for Restaurants

Revel Systems - це комплексна система управління рестораном, яка включає в себе POS-термінали, замовлення онлайн, облік запасів та аналітику для прийняття управлінських рішень. Ця система спрямована на великі та середні заклади харчування.

Переваги:

- Розширений функціонал для великих ресторанів та ланцюгів закладів.

- Висока аналітика та звітність для прийняття стратегічних рішень.

Недоліки:

- Вартість рішення може бути високою для невеликих бізнесів.
- Складніше встановлення та налаштування порівняно з іншими продуктами.

На рисунку 1.3 показується інтерфейс Revel Systems.

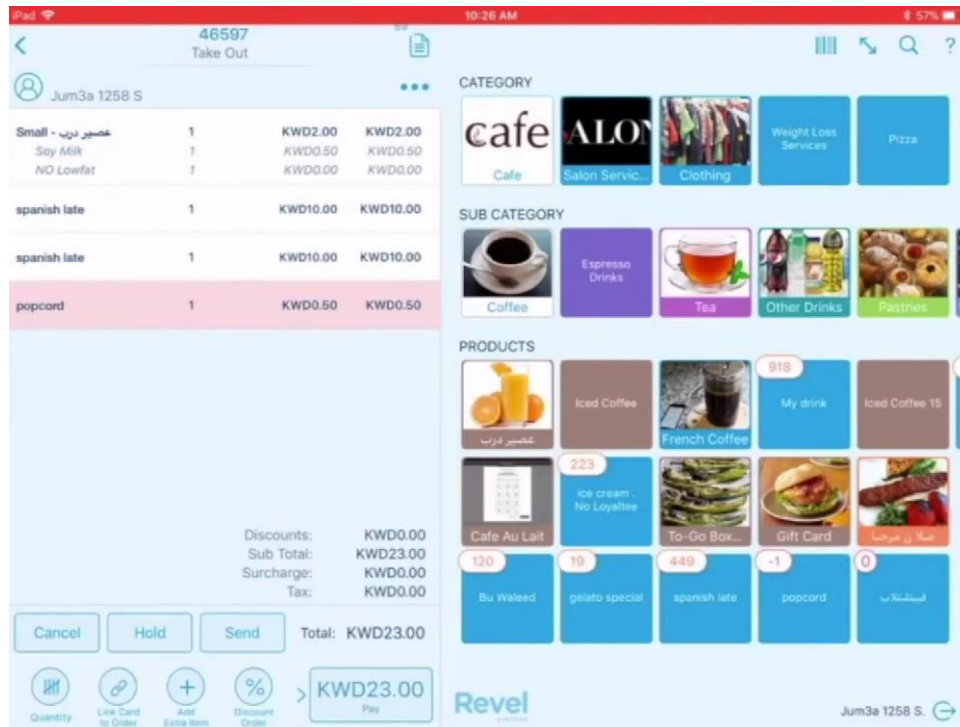


Рисунок 1.3 - Інтерфейс Revel Systems

В таблиці 1.1 представлено порівняння характеристик різних систем, зокрема Toast, Square for Restaurants та Revel Systems.

Таблиця 1.1

Порівняння аналогів

Характеристика	Toast	Square for Restaurants	Revel Systems
Зручність інтерфейсу	Зручний та настроюваний	Простий і інтуїтивний	Розширений та функціональний
Сумісність	Підтримка різних пристроїв та ОС	Широка, особливо з іншими продуктами Square	Призначений для великих та середніх закладів
Інтеграція з іншими сервісами	Забезпечена	Висока	Висока

Продовження таблиці 1.1

Облік робочого часу персоналу	Так	Так	Так
Ціна	Висока	Різні пакети вартістю від \$60 на місяць	Залежить від потреб клієнта, висока для великих ресторанів
Інтеграція з POS-терміналами	Так	Так	Так
Облік запасів	Так	Так	Так

Під час розгляду аналогів автоматизованих систем управління закладом харчування важливо враховувати конкретні потреби та масштаб закладу. Toast відзначається розгорнутим функціоналом та можливістю індивідуального налаштування. Square for Restaurants приваблює простотою використання та широкою інтеграцією з іншими сервісами Square. Revel Systems надає розширений функціонал для великих ресторанів та ланцюгів закладів.

Розробка нової автоматизованої системи повинна враховувати ринкові потреби, конкурентоспроможність вартості та забезпечення високоякісного обслуговування.

1.3 Вибір архітектури автоматизованої системи управління закладом харчування

Однією з найважливіших стратегічних вирішень у розробці автоматизованої системи управління закладом харчування є вибір оптимальної архітектури. Коректно побудована архітектура може значно вплинути на ефективність, гнучкість та масштабованість системи. У рамках аналізу напрямків використання інформаційних технологій для реалізації автоматизованої системи управління закладом харчування було ретельно розглянуто кілька архітектурних підходів, серед яких монолітна, мікросервісна та сервер-клієнтська архітектури визначені як найбільш перспективні.

1. Монолітна архітектура:

- Опис: монолітна архітектура передбачає розміщення всіх компонентів та функціональності системи в одному кодовому блоку.
- Переваги: легкість розробки та підтримки, одна кодова база, простота масштабування.
- Недоліки: складність масштабування окремих компонентів, обмежена гнучкість та оновлення.

Монолітна архітектура може бути ефективною у випадках, коли система не має великої кількості функціональних взаємозв'язків та потребує швидкого впровадження. Однак у випадку розробки системи управління закладом харчування, яка повинна взаємодіяти з різними модулями, такими як управління меню, замовленнями, робота з персоналом та складськими запасами, монолітна архітектура може виявитися обмеженою у забезпеченні гнучкості та масштабованості.

2. Мікросервісна архітектура:

- Опис: мікросервісна архітектура розбиває систему на невеликі, автономні та незалежні сервіси, що виконують конкретні функції.
- Переваги: легкість масштабування окремих компонентів, незалежність сервісів, зручність у розробці та підтримці.
- Недоліки: складність управління та координації сервісів, можливі проблеми з консистентністю даних.

Мікросервісна архітектура видається найбільш привабливою для системи управління закладом харчування, оскільки вона дозволяє розгортати та масштабувати окремі сервіси незалежно один від одного. Кожен сервіс може відповідати за конкретний аспект системи, такий як обробка замовлень, управління меню чи робота зі складськими запасами. Це спрощує розробку та підтримку, а також дозволяє ефективно використовувати ресурси системи.

3. Сервер-клієнтська архітектура:

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

- Опис: сервер-клієнтська архітектура передбачає розділення системи на серверну та клієнтську частини, що взаємодіють через API.
- Переваги: ефективність у розробці та масштабуванні, спрощена підтримка різних платформ.
- Недоліки: залежність від мережі для взаємодії між клієнтом та сервером, можливі проблеми з безпекою.

Сервер-клієнтська архітектура дозволяє ефективно розділити логіку обробки даних та відображення інтерфейсу між сервером та клієнтом. Це особливо корисно для системи управління закладом харчування, оскільки дозволяє забезпечити оптимальну ефективність та зручність для користувачів на різних пристроях.

Порівняльний аналіз наведено у Таблиці 1.2.

Таблиця 1.2

Порівняльний аналіз архітектур

Характеристика	Мікросервіси	Моноліт	Сервер-клієнтська
Гнучкість та масштабованість	Висока	Середня	Середня
Незалежність та розподіленість	Середня	Низька	Висока
Технологічна різноманітність	Висока	Низька	Висока
Підтримка різних платформ	Висока	Середня	Висока
Швидкість розгортання та вдосконалення	Висока	Середня	Середня

Після ретельного аналізу вищезазначених архітектур, було визначено, що сервер-клієнтська архітектура є оптимальним вибором для автоматизованої системи управління закладом харчування з численними перевагами. Вона забезпечує чітке розділення між серверною і клієнтською частинами, що сприяє ефективній організації бізнес-логіки та взаємодії з користувачем. Такий підхід дозволяє системі бути масштабованою, забезпечуючи легкість у підтримці і розширенні функціоналу без значних витрат на переробку коду або зміну архітектури. Крім того, чітке розділення

між клієнтською та серверною частинами сприяє підвищенню надійності системи, зменшуючи вплив відмов однієї частини на інші.

Такий підхід забезпечує ефективне використання ресурсів сервера для обробки та зберігання даних, тоді як клієнтська частина може оптимально взаємодіяти з користувачем, забезпечуючи зручний інтерфейс та швидкий доступ до інформації. Така архітектура сприяє також гнучкості у розробці, оскільки окремі компоненти можуть бути розроблені та оновлюватися незалежно, що робить процес розробки більш швидким та ефективним.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

При розробці автоматизованої системи управління закладом харчування, обрана сервер-клієнтська архітектура та інструменти грають важливу роль у забезпеченні ефективності, надійності та зручності користування. Нижче наведено обґрунтування вибору інструментальних засобів для серверної та клієнтської частин системи та визначено вимоги до апаратного забезпечення.

У розробці серверної частини системи використання сучасних веб-технологій та моделі REST API є стратегічно важливим. Для забезпечення оптимальної взаємодії з іншими компонентами системи обрано веб-сервер Nginx та систему управління базами даних MySQL.

Nginx обраний через його відмінність у швидкодії, ефективності та надійності. Використання цього веб-сервера дозволяє забезпечити стабільну роботу системи при великому навантаженні. Також, Nginx володіє високим рівнем захисту та можливістю обробки багатьох одночасних з'єднань, що є важливим для оптимальної роботи системи управління закладом харчування.

MySQL визначений як система управління базами даних через свою надійність, швидкодію та широкі можливості оптимізації. Використання цієї

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

бази даних дозволяє забезпечити ефективне зберігання та обробку інформації про меню, запаси і замовлення в системі управління закладом харчування.

Для забезпечення коректної роботи всієї системи та полегшення розробки, рекомендується використовувати платформу Docker. Docker дозволяє поєднати всі компоненти системи та забезпечити їхню незалежність від конкретного середовища виконання. Це забезпечує високий рівень стабільності та зручності у розгортанні.

Клієнтська частина системи також відіграє ключову роль у забезпеченні зручності користування та взаємодії з користувачем. Обрано такі технології як React, HTML, CSS і JavaScript для розробки клієнтського інтерфейсу. Обґрунтування цього вибору базується на наступних перевагах:

React є одним з найпопулярніших фреймворків для розробки користувацьких інтерфейсів веб-додатків[7]. Він пропонує модульну архітектуру, яка дозволяє розбити інтерфейс на невеликі компоненти, що спрощує розробку та підтримку коду. Крім того, React використовує віртуальний DOM, що підвищує продуктивність додатка шляхом ефективного оновлення інтерфейсу при зміні даних.

HTML, CSS і JavaScript: HTML, CSS і JavaScript є основними мовами для розробки веб-інтерфейсів. Вони забезпечують можливість створювати структуру, стиль та поведінку веб-сторінок відповідно. Використання цих мов дозволяє створювати красивий та функціональний інтерфейс, який ефективно взаємодіє з користувачем.

Інтеграція та ефективність також є важливими аспектами розробки системи. Використання Docker допомагає у забезпеченні незалежності компонентів системи та легкості у розгортанні. Обґрунтування вибору Docker базується на наступних перевагах:

Docker є відомим контейнеризаційним інструментом, який дозволяє упаковувати програмне забезпечення та його залежності в стандартизовані контейнери. Це спрощує розгортання та управління додатками, забезпечуючи

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

їхню консистентність у будь-якому середовищі. Docker також дозволяє ефективно використовувати ресурси сервера, запускаючи кожен компонент системи в окремому контейнері.

Загалом, вибір технічних інструментів та визначення вимог до апаратного забезпечення базується на необхідності забезпечити ефективну та стабільну роботу системи управління закладом харчування. Кожен вибраний інструмент має свої унікальні переваги, які сприяють досягненню цієї мети.

Для оптимальної роботи серверу та пристрою користувача встановлені наступні мінімальні вимоги:

Системні вимоги до серверу:

- Процесор: мінімум 4-ядерний процесор з тактовою частотою не менше 2.0 ГГц.
- Оперативна пам'ять: мінімум 8 ГБ оперативної пам'яті для забезпечення швидкої обробки запитів та динамічних операцій з базою даних.
- Зберігання: жорсткий диск або SSD об'ємом не менше 100 ГБ для зберігання бази даних, конфігурацій та інших необхідних файлів.

Системні вимоги до пристрою користувача:

- Процесор: мінімум 2-ядерний процесор з тактовою частотою не менше 1.5 ГГц.
- Оперативна пам'ять: мінімум 4 ГБ оперативної пам'яті для ефективного використання клієнтської частини системи та забезпечення зручності користування.
- Відображення: монітор з роздільною здатністю не менше 1280x800 пікселів для оптимального відображення інтерфейсу.

Ці мінімальні системні вимоги спрямовані на забезпечення ефективності та стабільності роботи системи управління закладом харчування при нормальному навантаженні. В разі великого обсягу даних та

великої кількості одночасних користувачів може знадобитися збільшення обсягу ресурсів.

ВИСНОВКИ ДО ПЕРШОГО РОЗДІЛУ

Розділ наголошує на важливості інтеграції сучасних технологій у сферу ресторанного бізнесу. Розглянуті напрямки показують, що автоматизація процесів може значно підвищити ефективність управління, якість обслуговування клієнтів та точність фінансового обліку.

Перш за все, аналіз проєкту та постановка задачі визначають основні вимоги до майбутньої системи, серед яких збір і аналіз вимог, розробка гнучкої архітектури та реалізація функціоналу. Ці етапи важливі для впровадження системи, яка забезпечить точне відстеження залишків продуктів, планування закупівель та управління замовленнями.

Далі, аналіз аналогів програмних продуктів, таких як Toast, Square for Restaurants та Revel Systems, підкреслює важливість вибору оптимального рішення для конкретного закладу. Кожен з цих продуктів має свої переваги та обмеження, що варто враховувати при розробці нової системи.

Вибір оптимальної архітектури для системи управління закладом харчування, а саме сервер-клієнтської архітектури, обґрунтовується її здатністю до ефективного розподілу завдань між клієнтською та серверною частинами, що сприяє збереженню надійності та швидкості роботи системи.

Обґрунтування вибору інструментальних засобів і вимог до апаратного забезпечення вказують на необхідність використання сучасних технологій та відповідності їх функціональних характеристик потребам системи управління закладом харчування.

Отже, розділ надає повний обзор основних аспектів, які необхідно врахувати при розробці та впровадженні автоматизованої системи управління закладом харчування, зокрема вимоги до функціоналу, вибір архітектури та вибір інструментів.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		19

РОЗДІЛ 2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Визначення варіантів використання та структури системи

Для забезпечення неперервного виконання необхідних функцій і відповідності встановленим стандартам та специфікаціям, наведено високорівневі вимоги, які повинна задовольняти автоматизована система управління закладом харчування. Ці вимоги допомагають забезпечити безперебійну роботу системи та виконання всіх необхідних стандартів і специфікацій[8].

Бізнес вимоги

- Основні цілі: проект створюється з метою спростити взаємодію між власниками закладів харчування та автоматизації певних процесів[9].
- Представлення проекту: проект буде реалізовано у вигляді веб-сайту, що містить структуровану інформацію по даній тематичній області.

Вимоги користувачів

- Можливість створення та редагування особистих даних як власника, так і відвідувачів закладу харчування[10];
- Можливість зручного внесення інформації;
- Можливість створення меню та планів розважальних заходів;
- Можливість зміни інформації або її видалення;
- Можливість переглядати список як закладів харчування, меню, так і розважальних заходів;
- Можливість робити замовлення страв;

Вимоги персоналу веб-додатку:

- Можливість перегляду списку, створення, редагування та видалення усіх даних, які стосуються лише закладів харчування.
- Можливість зв'язку з користувачами, коли вони замовляють страви, та останні вже будуть відправлятися у чат-бот як замовлення.

Визначення варіантів використання на основі юзер-сторі

Адміністратор

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		20

Управління меню:

- Адміністратор може додавати, редагувати та видаляти страви в меню.
- Додавання нових страв включає введення інгредієнтів через пов'язану таблицю menu_item. На сторінці меню має бути окрема колонка "Інгредієнти", що відображає список продуктів.

Управління персоналом:

- Адміністратор може додавати, редагувати та видаляти інформацію про персонал.
- На сторінці додавання/редагування персоналу повинна бути можливість призначення ролей.

Управління столами:

- Адміністратор може додавати, редагувати та видаляти інформацію про столи.

Управління резерваціями:

- Адміністратор може переглядати, редагувати та видаляти резервації столів.

Управління змінами:

- Адміністратор може переглядати і затверджувати робочі зміни співробітників.

2. Офіціант

Прийом замовлень:

- Офіціант має змогу бачити, редагувати, додавати замовлення.
- При створенні замовлення є поле для додавання кількох продуктів.
- Офіціант може перевіряти статус замовлень в реальному часі та редагувати його.

Перегляд змін:

- Адміністратор може переглядати свої робочі зміни.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		21

3. Клієнт

Перегляд меню:

- Клієнт може переглядати меню з описом страв і цінами.

Відстеження статусу замовлення:

- Клієнт може відстежувати статус свого замовлення в реальному часі.

Резервування столу:

- Клієнт може резервувати стіл.

Аналіз вимог користувачів дав підстави для визначення варіантів використання додатку автоматизованої системи управління закладом харчування[10]. Діаграма взаємодії (USE CASE) допомагає візуалізувати взаємодію між різними ролями (Адміністратор, Офіціант, Клієнт) та системою управління (рис. 2.1). Основні функції системи включають управління меню, управління персоналом, прийом замовлень, резервування столів та управління змінами. Кожна роль пов'язана з конкретними завданнями/діями, які вони можуть виконувати відповідно до своїх прав та обов'язків.

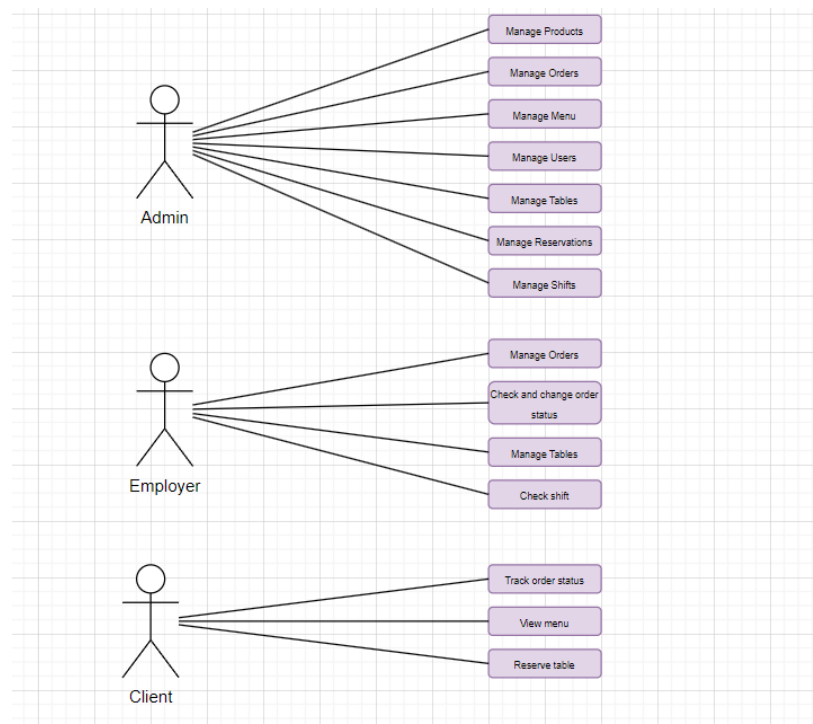


Рисунок 2.1 - Діаграма взаємодії

Об'єктно орієнтована модель системи зображується у вигляді діаграми класів. Діаграма класів показує структуру системи шляхом зображення класів, атрибутів, методів та зв'язків між класами [16].

Отримавши необхідні дані, було створено діаграму класів, яку можна побачити на рисунку 2.2.

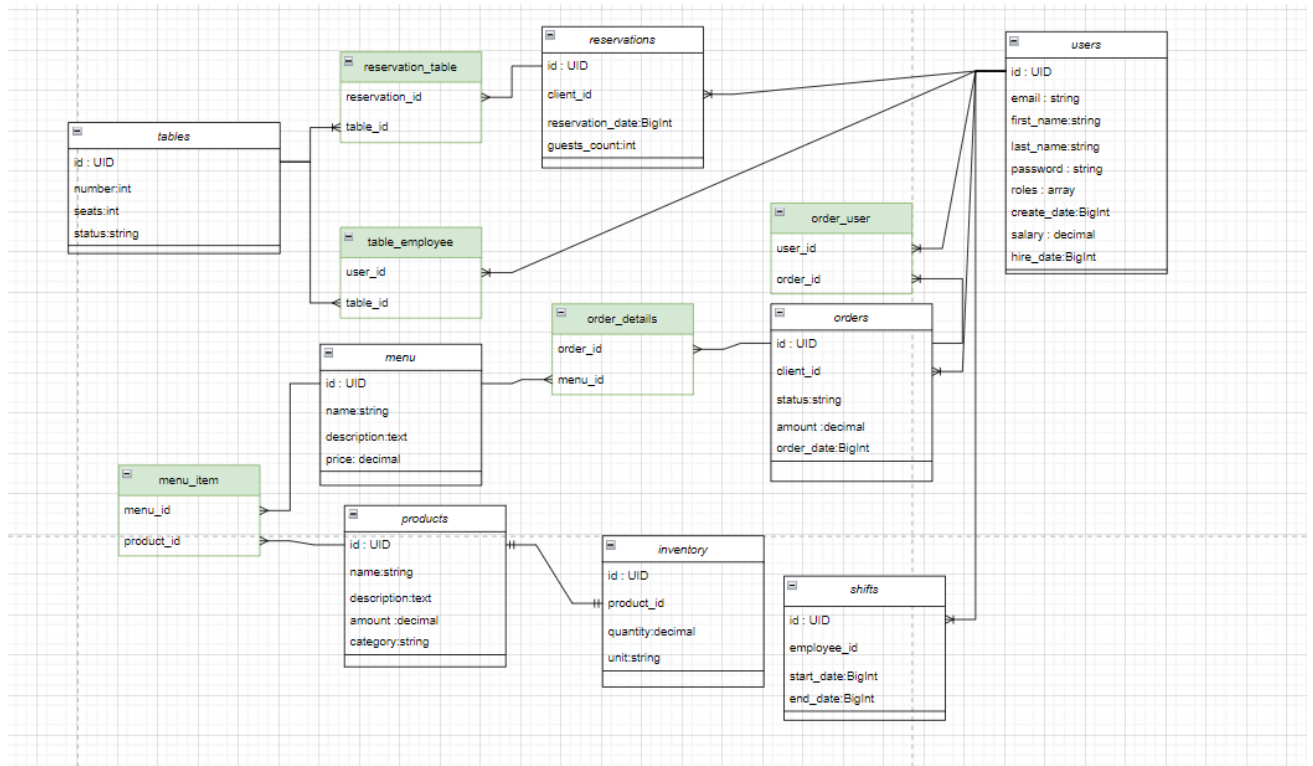


Рисунок 2.2 - Діаграма класів

Отже, діаграма класів складається з:

1. User (Користувач): базовий клас, що містить загальну інформацію для всіх ролей користувачів (адміністратор, клієнт, офіціант).
2. Menu (Меню): цей клас представляє перелік страв, яку можна замовити.
3. Product (Продукт): цей клас представляє продукт, який використовується для приготування страв.
4. Inventory (Інвентар): цей клас представляє інформацію про кількість продуктів, які входять в інвентар.
5. Menu_item (Елемент меню): цей клас представляє зв'язок між стравами та продуктами, що входять до їх складу.

6. Order (Замовлення): цей клас представляє замовлення, яке робить клієнт.
7. Order_details (Деталі замовлення): цей клас представляє деталі замовлення, а саме які страви входять до замовлення.
8. Order_user (Замовлення клієнта): цей клас представляє зв'язок між замовленням та клієнтом, що його замовив.
9. Table (Стіл): цей клас представляє інформацію про столи у закладі.
10. Reservation (Резервація): цей клас представляє бронювання.
11. Reservation_table (Резервація столу): цей клас представляє зв'язок між столами та резерваціями, що входять до резервації.
12. Shift (Робоча зміна): цей клас представляє робочі зміни працівників.

На етапі проектування вирішальним є передбачення проблем, з якими може стикнутися продукт. Однією з ключових є розширення чи модифікація логіки системи. Ці завдання можуть вимагати значних часових та трудових ресурсів для їх вирішення та подальшого управління подібними випадками. З метою ефективного вирішення таких та подібних викликів, створені специфічні патерни проектування – шаблони для типових проблем у програмуванні.

Під час розробки діаграм класів виявлено дві ключові проблеми:

1. Як врахувати можливість розширення параграфів залежно від різних властивостей?
2. Як ідентифікувати методи окремих ролей, особливо, якщо це всі користувачі?

Для подолання цих викликів обрано паттерн програмування - прототипування.

Прототипування

Для забезпечення можливості розширення параграфів та ідентифікації методів для різних ролей, використано патерн прототипування. Це дозволяє створювати нові об'єкти за допомогою копіювання вже існуючих, що робить систему гнучкою та легкою для розширення.

Переваги патерну:

- Легкість розширення: можна створювати нові об'єкти на основі існуючих.
- Гнучкість: легко змінювати структуру об'єкта без впливу на інші об'єкти.
- Спрощення коду: менша кількість дублювання коду.

Діаграма класів відображає структуру та взаємодію між класами в системі управління закладом харчування. Використання патерну прототипування допомагає вирішувати проблеми розширення та модифікації логіки системи, забезпечуючи гнучкість та зручність в управлінні.

2.2 Розробка бази даних системи

Після проведеного аналізу було визначено, що найкращим вибором для зберігання даних є реляційна система управління базами даних MySQL. Вибір був обґрунтований кількома ключовими факторами. По-перше, MySQL має широку підтримку та активну спільноту користувачів, що забезпечує доступ до великої кількості документації та допомоги у вирішенні проблем. По-друге, ця СУБД відома високою продуктивністю, що є важливим для швидкої обробки великих обсягів даних. Крім того, MySQL забезпечує можливість масштабування як у вертикальному, так і в горизонтальному напрямках, що дозволяє збільшувати потужності системи у відповідності до зростання обсягу даних та кількості користувачів. Вбудовані функції безпеки, такі як управління доступом користувачів та шифрування даних, також є важливими для захисту чутливої інформації. Нарешті, MySQL

легко інтегрується з різними мовами програмування та фреймворками, забезпечуючи гнучкість у виборі технологій для реалізації системи.

На основі вимог до системи було розроблено концептуальну модель бази даних, яка включає кілька основних таблиць для зберігання даних про інвентар, меню, замовлення, продукти, бронювання та користувачів. Таблиця інвентарю зберігає інформацію про наявність продуктів на складі, а таблиця меню містить дані про доступні меню та їх опис. Зв'язок між меню та продуктами відображається в таблиці елементів меню. Замовлення та їх деталі зберігаються в окремих таблицях, що дозволяє відстежувати статус замовлень, загальну суму та дати замовлень. Таблиця продуктів містить інформацію про кожен продукт, включаючи назву, опис, кількість та категорію. Для бронювань створено окрему таблицю, яка зберігає дані про клієнтів, дати бронювань та кількість гостей. Уся ця інформація дозволяє ефективно керувати даними та забезпечувати швидкий доступ до них у рамках системи.

База даних системи для автоматизації управління закладом харчування складається із наступних таблиць, наведених на діаграмі на рисунку 2.3.

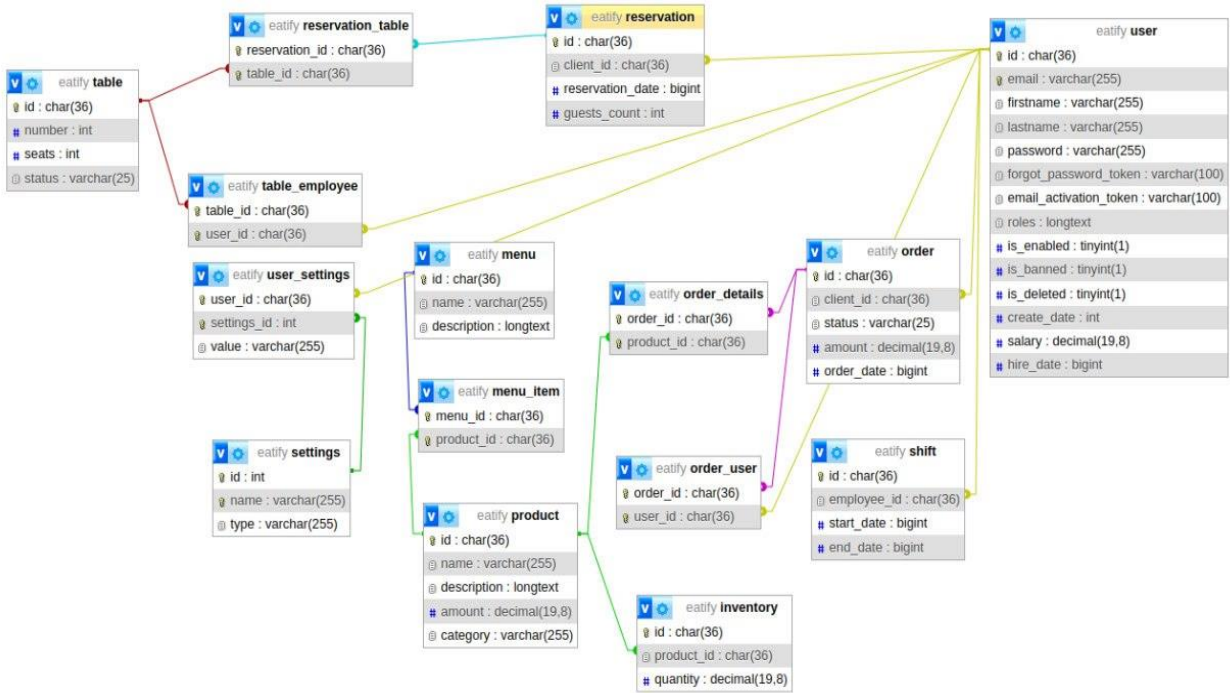


Рисунок 2.3 – Діаграма бази даних

Далі наведено детальний опис полів кожної таблиці бази даних на таблицях 2.1 – 2.15. Вони включають інформацію про інвентар, меню, замовлення, продукти, бронювання, користувачів та інші аспекти, необхідні для повного функціонування та управління рестораном чи кафе[11]. Кожна таблиця має свою структуру полів, що дозволяє чітко організувати дані і забезпечити їх ефективний доступ та обробку в межах системи.

Таблиця 2.1

Опис полів таблиці «menu»

Назва	Тип даних	PK	FK	Опис
id	char(36)	+	-	Унікальний ідентифікатор
name	varchar(255)	-	-	Назва страви
description	longtext	-	-	Опис страви

Таблиця 2.2

Опис полів таблиці «menu_item»

Назва	Тип даних	PK	FK	Опис
menu_id	char(36)	+	menu.id	Ідентифікатор страви
product_id	char(36)	+	product.id	Ідентифікатор продукту

Таблиця 2.3

Опис полів таблиці «order»

Назва	Тип даних	PK	FK	Опис
id	char(36)	+	-	Унікальний ідентифікатор
client_id	char(36)	-	user.id	Ідентифікатор клієнта
status	varchar(25)	-	-	Статус замовлення

Таблиця 2.4

Опис полів таблиці «order_details»

Назва	Тип даних	PK	FK	Опис
order_id	char(36)	+	order.id	Ідентифікатор замовлення
product_id	char(36)	+	product.id	Ідентифікатор продукту

Таблиця 2.5

Опис полів таблиці «order_user»

Назва	Тип даних	PK	FK	Опис
order_id	char(36)	+	order.id	Ідентифікатор замовлення
user_id	char(36)	+	user.id	Ідентифікатор клієнта

Таблиця 2.6

Опис полів таблиці «product»

Назва	Тип даних	PK	FK	Опис
id	char(36)	+	-	Унікальний ідентифікатор
name	varchar(255)	-	-	Назва продукту
description	longtext	-	-	Опис продукту
amount	decimal(19,8)	-	-	Кількість продукту
category	varchar(255)	-	-	Категорія продукту

Таблиця 2.7

Опис полів таблиці «reservation»

Назва	Тип даних	PK	FK	Опис
id	char(36)	+	-	Унікальний ідентифікатор
client_id	char(36)	-	user.id	Ідентифікатор клієнта
reservation_date	bigint	-	-	Дата бронювання
guests_count	int	-	-	Кількість гостей

Таблиця 2.8

Опис полів таблиці «reservation_table»

Назва	Тип даних	PK	FK	Опис
reservation_id	char(36)	+	reservation.id	Ідентифікатор резервації
table_id	char(36)	+	table.id	Ідентифікатор стола

Таблиця 2.9

Опис полів таблиці «shift»

Назва	Тип даних	PK	FK	Опис
id	char(36)	+	-	Унікальний ідентифікатор
employee_id	char(36)	-	user.id	Ідентифікатор працівника
start_date	bigint	-	-	Дата початку зміни
end_date	bigint	-	-	Дата завершення зміни

Таблиця 2.10

Опис полів таблиці «table»

Назва	Тип даних	PK	FK	Опис
id	char(36)	+	-	Унікальний ідентифікатор
number	int	-	-	Номер столу
seats	int	-	-	Кількість місць
status	varchar(25)	-	-	Статус

Таблиця 2.11

Опис полів таблиці «table_employee»

Назва	Тип даних	PK	FK	Опис
table_id	char(36)	+	table.id	Ідентифікатор стола
user_id	char(36)	+	user.id	Ідентифікатор працівника

Таблиця 2.12

Опис полів таблиці «user»

Назва	Тип даних	PK	FK	Опис
id	char(36)	+	-	Унікальний ідентифікатор
email	varchar	-	-	Електронна пошта
firstname	varchar	-	-	Ім'я
lastname	varchar(255)	-	-	Прізвище
password	varchar(255)	-	-	Пароль

Продовження таблиці 2.12

forgot_password_token	varchar(100)	-	-	Токен для відновлення пароля
email_activation_token	varchar(100)	-	-	Токен для активації електронної пошти
roles	longtext	-	-	Ролі користувача
is_enabled	tinyint(1)	-	-	Чи активований користувач (0 – ні, 1 – так)
is_banned	tinyint(1)	-	-	Чи заблокований користувач (0 – ні, 1 – так)
is_deleted	tinyint(1)	-	-	Чи видалений користувач (0 – ні, 1 – так)
create_date	int	-	-	Дата створення
salary	decimal(19,8)	-	-	Зарплата
hire_date	bigint	-	-	Дата найму

Таблиця 2.13

Опис полів таблиці «inventory»

Назва	Тип даних	РК	FK	Опис
-------	-----------	----	----	------

id	char(36)	+	-	Унікальний ідентифікатор
product_id	char(36)	-	product.id	Ідентифікатор продукту
quantity	decimal(19,8)	-	-	Кількість продукту на складі

Таблиця 2.14

Опис полів таблиці «user_settings»

Назва	Тип даних	PK	FK	Опис
user_id	char(36)	+	user.id	Ідентифікатор користувача
settings_id	char(36)	+	settings.id	Ідентифікатор налаштування

Таблиця 2.15

Опис полів таблиці «settings»

Назва	Тип даних	PK	FK	Опис
id	char(36)	+	-	Унікальний ідентифікатор
name	varchar	-	-	Назва налаштування
type	varchar	-	-	Тип налаштування

2.3 Проєктування та реалізація алгоритмів роботи системи

З метою зображення алгоритму роботи та певні стани програмної системи було створено діаграми активності.

Діаграма активності дозволяє моделювати послідовності бізнес-процесів або дій[15]. Вона демонструє дії, рішення, вузли, цикли та паралельні дії – логіку певного процесу.

Така діаграма будується з певної обмеженої кількості фігур, які з'єднані.

Основні з них:

1. Дії (скруглені прямокутники)
2. Стан прийняття рішення (ромб)
3. З'єднувальні лінії, які поєднують потік дій
4. Початок та кінець, які означають початок діяльності та кінець відповідно.

Було побудовано діаграми діяльності процесу оформлення замовлення[20]. Діаграма "Оформлення замовлення", яка зображена на рисунку 2.4, відображає послідовність дій офіціанта під час процесу оформлення замовлення в автоматизованій системі управління закладом харчування.

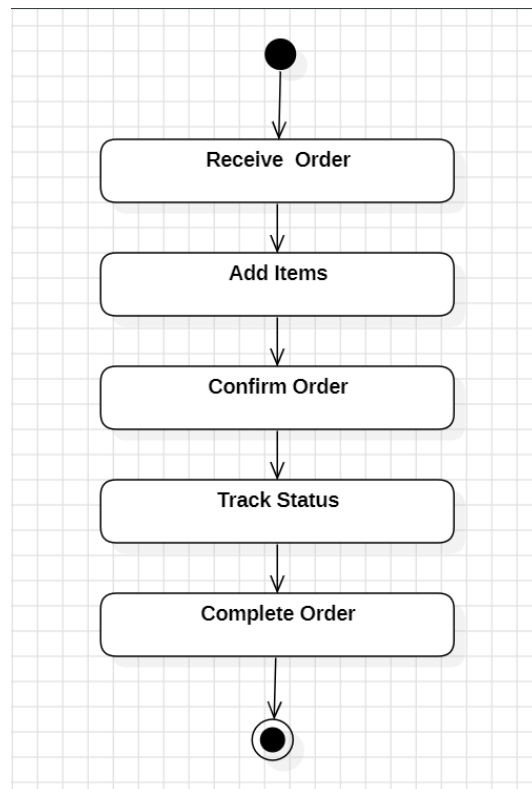


Рисунок 2.4 – Діаграма активності оформлення замовлення

Кроки процесу оформлення замовлення:

1. Прийом замовлення:
 - Офіціант отримує замовлення від клієнта або оформлює його через систему.
 - В системі відображається інформація про столик/клієнта, а також список обраних страв.
2. Додавання продуктів:
 - Офіціант додає до замовлення необхідні страви, вказуючи кількість порцій кожної страви.
3. Підтвердження замовлення:
 - Після додавання всіх страв офіціант перевіряє замовлення на вірність і повноту.
 - Він підтверджує замовлення в системі, щоб воно було передано на кухню для підготовки.
4. Відстеження статусу:
 - Офіціант може в реальному часі відстежувати статус підготовки замовлення на кухні.
 - Якщо необхідно, він може взаємодіяти з кухнею або кухарями для уточнень.
5. Завершення замовлення:
 - Після того як страви готові, офіціант отримує повідомлення від кухні про готовність замовлення.
 - Він забирає страви і доставляє їх клієнту або повідомляє про готовність столика для обслуговування.

2.4 Реалізація функціоналу системи

Підйом контейнерів Docker та структура проекту

Проект використовує Docker для контейнеризації своїх компонентів, що дозволяє забезпечити їх ізоляцію та зручність у розгортанні. Docker надає

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		34

можливість створити образи для кожного компоненту системи, а потім запустити їх як контейнери.

Основним інструментом для управління контейнерами є файл `docker-compose.yml`, який визначає конфігурацію для всіх необхідних сервісів. Цей файл визначає три основні служби:

- PHP: будується з контексту `./docker/php`, монтує код з папки `./api` і встановлює змінну середовища `APP_ENV`[13].
- DB: використовує образ MySQL, зберігає дані у об'ємі `db_data`, і встановлює необхідні змінні середовища для конфігурації бази даних.
- Nginx: будується з контексту `./docker/nginx`, монтує код з папки `./api`, відкриває порт 8080 для доступу і залежить від служби `php`[19].

Файл `recreate.sh` автоматизує процес підняття контейнерів Docker, очищення старих контейнерів та конфігурацію JWT:

```
#!/bin/sh
RED='\033[0;31m'
GREEN='\033[0;32m'
NC='\033[0m'
echo -e "${NC}STARTING REBUILD PROCESS${GREEN}"
docker stop $(docker ps -aq)
docker rm $(docker ps -aq)
docker compose -f docker-compose.yaml -f docker-compose.local.yaml up --build -d
docker compose exec api composer update
echo -e ""
echo -e ""
key=$(awk -F= '/JWT_PASSPHRASE/ { print $2 }' api/.env)
rm -r api/config/jwt
mkdir -p api/config/jwt
chmod 777 api/config/jwt
echo -e "${GREEN}Used KEY for JWT: ${NC}" $key
echo -e ""
echo -e ""
docker compose exec api openssl genrsa -out config/jwt/private.pem -aes256 -
passout pass:$key 4096
docker compose exec api openssl rsa -pubout -in config/jwt/private.pem -passin
pass:$key -out config/jwt/public.pem
docker compose exec api chmod -R 777 config/jwt/
echo -e ""
echo -e ""
docker compose exec api php bin/console cache:clear
echo -e "${GREEN}Clearing cache - done${NC}"
echo -e " "
docker compose exec api chmod -R 777 var/cache
echo -e "${GREEN}Add rewrite rules 777 to var/cache - done${NC}"
echo -e " "
docker compose exec api chmod -R 777 var/log
echo -e "${GREEN}Add rewrite rules 777 to var/log - done${NC}"
```

```
echo -e " "
echo -e "${NC}FINISHED!${GREEN}"
```

За допомогою команди `docker ps -a` перевіряється чи всі контейнери працюють коректно. Приклад правильної роботи контейнерів можна побачити на рисунку 2.5.

```
nika@nika-VivoBook-15-ASUS-Laptop-X570UD:~/Документи/diploma/bw_mishchuk_ist-20-2$ docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
48474c68298    bw_mishchuk_ist-20-2-nginx         "/docker-entrypoint..." 8 days ago    Up 40 hours   0.0.0.0:80->80/tcp, :::80->80/tcp    eatify-nginx
656b95d72c40    phpmyadmin/phpmyadmin             "/docker-entrypoint..." 8 days ago    Up 40 hours   0.0.0.0:8080->80/tcp, :::8080->80/tcp eatify-phpmyadmin
2974e11345bc    bw_mishchuk_ist-20-2-api           "/docker-php-entrypoi..." 8 days ago    Up 40 hours   9000/tcp                           eatify-api
437a12d7e5b6    dunglas/mercure:v0.14             "/usr/bin/caddy run ..." 8 days ago    Up 40 hours   443/tcp, 2019/tcp, 443/udp, 0.0.0.0:8083->80/tcp, :::8083->80/tcp eatify-mercurehub
98086a8c08f1    mysql:8.0.20                       "docker-entrypoint.s..." 8 days ago    Up 40 hours   33060/tcp, 0.0.0.0:13306->3306/tcp, :::13306->3306/tcp eatify-mysql
9319cca402ad    bw_mishchuk_ist-20-2-frontend      "docker-entrypoint.s..." 8 days ago    Up 40 hours   0p 40 hours                       eatify-frontend
```

Рисунок 2.5 – Приклад правильної роботи контейнерів

Структура проекту організована за допомогою Docker, що дозволяє ізолювати різні компоненти системи:

- **api**: Директорія для бекенд частини проекту, що містить вихідний код, конфігурації та залежності API.
- **frontend**: Директорія для фронтенд частини проекту, що містить вихідний код, конфігурації та залежності клієнтської частини.
- **docker**: Директорія для конфігурацій Docker, що містить Dockerfiles та інші конфігураційні файли для створення контейнерів.

Структуру проекту можна побачити на рисунку 2.6.

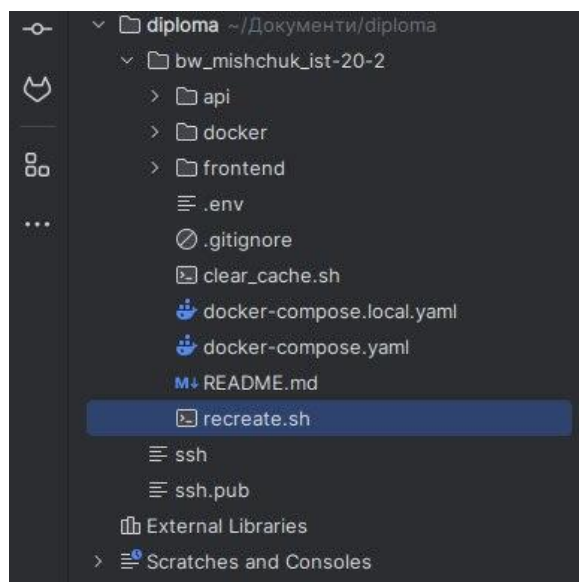


Рисунок 2.6 – Структура проекту

Хешування паролю та відправка повідомлення на пошту

Одним з ключових аспектів роботи системи є забезпечення безпеки паролів користувачів шляхом їх хешування перед збереженням у базі даних та автоматична відправка повідомлення на електронну пошту користувача після успішної реєстрації. Це реалізовано за допомогою класу `UserEntityListener`, який використовує події Doctrine для виконання цих дій.

Опис основних етапів коду:

1. Імпорт необхідних модулів:

```
use App\Entity\User;
use Doctrine\ORM\EntityManagerInterface;
use Exception;
use Hackzilla\PasswordGenerator\Generator\ComputerPasswordGenerator;
use Psr\Log\LoggerInterface;
use Symfony\Bridge\Twig\Mime\TemplatedEmail;
use Symfony\Component\Mailer\Exception\TransportExceptionInterface;
use Symfony\Component\Mailer\MailerInterface;
use Symfony\Component\Mime\Address;
use Symfony\Component>PasswordHasher\Hasher\UserPasswordHasherInterface;
```

Цей розділ коду імпортує необхідні модулі та класи, що використовуються для роботи з сутністю користувача, управління пароллями, логуванням, відправкою електронних листів та взаємодією з ORM[12].

2. Конструктор

```
public function __construct(
    UserPasswordHasherInterface $passwordHasher,
    MailerInterface             $mailer,
    LoggerInterface             $logger,
    EntityManagerInterface      $entityManager,
)
{
    $this->passwordHasher = $passwordHasher;
    $this->generator = new ComputerPasswordGenerator();
    $this->generator->setLength(16);
    $this->logger = $logger;
    $this->entityManager = $entityManager;
    $this->mailer = $mailer;
}
```

Конструктор класу ініціалізує його властивості, передані як залежності. Це дозволяє використовувати механізми хешування паролів, генерації паролів, логування, управління сутностями та відправки електронних листів у методах класу.

3. Метод `prePersist`

```
public function prePersist(User $user): void
```

```

{
    $password = $this->generator->generatePassword();

    $user->setGeneratedPassword($password);

    $user->setPassword($this->passwordHasher->hashPassword($user, $password));

    $user->setEmailActivationToken(bin2hex(random_bytes(15)));
}

```

Цей метод викликається перед тим, як новий користувач буде збережений у базі даних. Він виконує наступні дії:

- Генерує новий пароль за допомогою ComputerPasswordGenerator.
- Встановлює згенерований пароль у властивість generatedPassword об'єкта користувача.
- Хешує згенерований пароль та встановлює його у властивість password об'єкта користувача.
- Генерує токен активації електронної пошти та встановлює його у властивість emailActivationToken об'єкта користувача.

4. Метод postPersist:

```

public function postPersist(User $user): void
{
    $email = (new TemplatedEmail())
        ->from(new Address('eatify@gmail.com', 'Eatify'))
        ->to($user->getEmail())
        ->subject('Реєстрація на сайті')
        ->htmlTemplate('emails/user_registration.html.twig')
        ->context(
            [
                'user' => $user,
                'password' => $user->getGeneratedPassword()
            ]
        );
    $this->mailer->send($email);
    $this->entityManager->flush();
}

```

Цей метод викликається після того, як новий користувач був збережений у базі даних. Він виконує наступні дії:

- Створює електронний лист за допомогою TemplatedEmail, який містить інформацію про користувача та згенерований пароль.
- Відправляє електронний лист за допомогою MailerInterface.

- Зберігає зміни у базі даних.

ВИСНОВКИ ДО ДРУГОГО РОЗДІЛУ

У процесі проєктування автоматизованої системи управління закладом харчування було визначено бізнес-вимоги та вимоги користувачів, які впливають на функціональні можливості програмного забезпечення. З цією метою були створені детальні варіанти використання (use cases), які показують взаємодію між різними ролями користувачів - від адміністраторів до клієнтів.

Для візуалізації основних функціональних можливостей системи було розроблено діаграми використання та об'єктно-орієнтовану модель системи у вигляді діаграми класів. Ці інструменти дозволили чітко визначити структуру та взаємодію між різними компонентами програми.

Завершальним етапом була реалізація функціоналу системи з використанням сучасних технологій, включаючи контейнеризацію компонентів за допомогою Docker. Це забезпечило гнучкість управління системою та легкість розширення її функціональності.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		39

РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З АВТОМАТИЗОВАНОЮ СИСТЕМОЮ УПРАВЛІННЯ ЗАКЛАДОМ ХАРЧУВАННЯ

3.1 Порядок встановлення та налаштування параметрів системи

Діаграма розгортання відображає обчислювальні вузли під час роботи програми, компоненти та об'єкти, що виконуються на цих вузлах. За її допомогою можна провести аналіз необхідної апаратної конфігурації, на якій працюватимуть окремі процеси системи, а також описати їх взаємодію між собою та іншими пристроями.

На рисунку 3.1 показана змодельована діаграма розгортання для поставленої задачі.

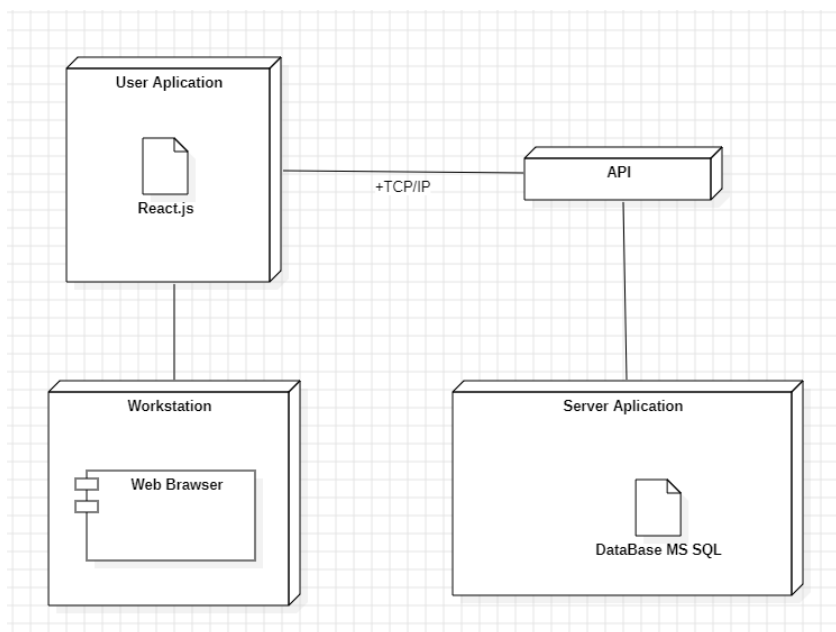


Рисунок 3.1. Діаграма розгортання системи

Ця діаграма розгортання системи показує взаємодію між користувацькою аплікацією, робочою станцією та серверною аплікацією. Користувацька аплікація, яка базується на React.js, комунікує з API через протокол TCP/IP. Робоча станція з веб-браузером також може взаємодіяти з цим API. Серверна аплікація пов'язана з базою даних MS SQL. Ця діаграма складається з трьох компонентів: “User Application”, “Workstation” та “Server Application”. Компонент

“User Application” базується на React.js та комунікує з API через протокол TCP/IP. Компонент “Workstation” з веб-браузером також може взаємодіяти з цим API. Компонент “Server Application” пов’язаний з базою даних MS SQL.

3.2 Структура інтерфейсу системи

При вході на сайт користувача зустрічає ненав’язлива головна сторінка. В правому верхньому кутку є кнопка ‘Авторизація’ для вже зареєстрованих користувачів та ‘Реєстрація’ для нових користувачів. Головна сторінка зображена на рисунку 3.2.

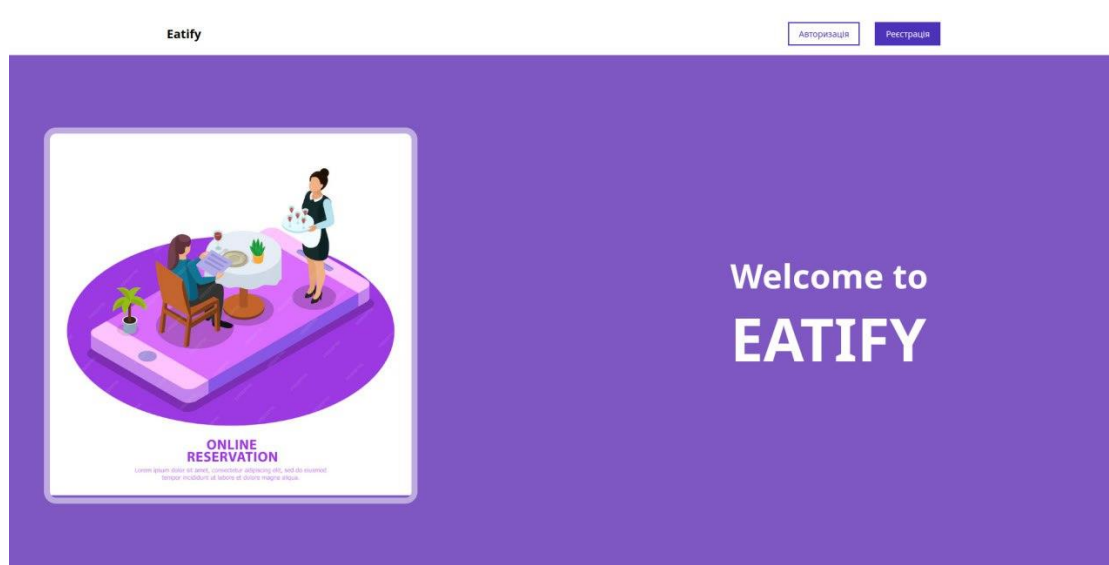
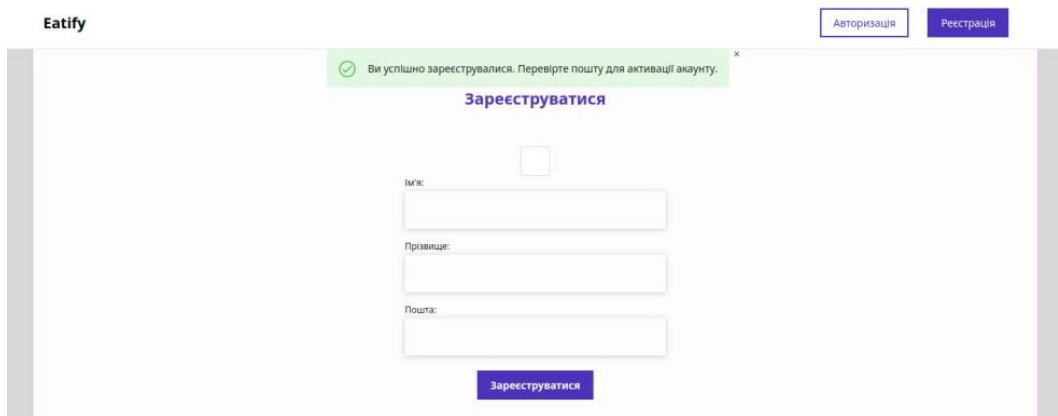


Рисунок 3.2 – Головна сторінка

Розглянемо процес реєстрації нового користувача натиснувши кнопку ‘Реєстрація’ (рисунок 3.3).

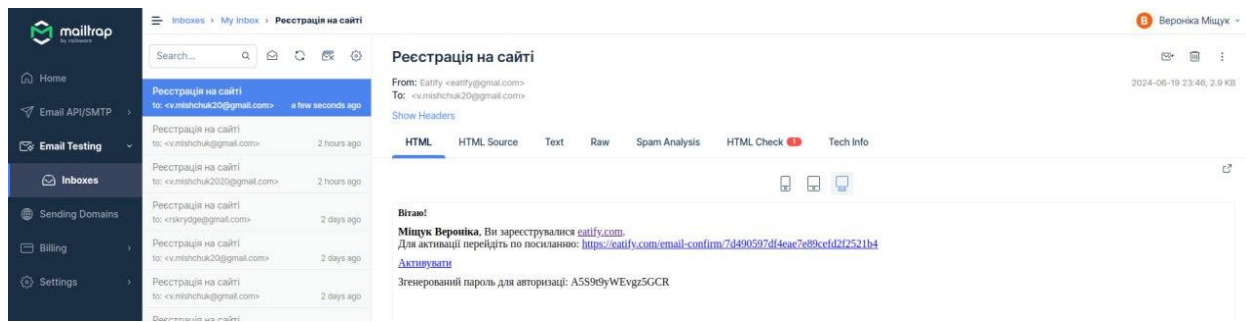
Рисунок 3.3 – Сторінка реєстрації

Після заповнення даних користувач відправляє форму та отримує повідомлення про успішну реєстрацію та необхідність підтвердити пошту для активації акаунту (рисуюнок 3.4).



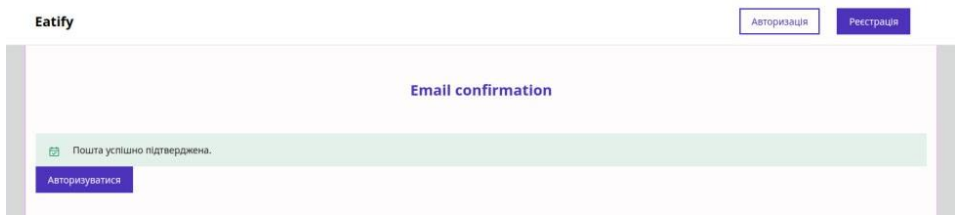
Рисуюнок 1.4 – Повідомлення після відправки форми реєстрації

Наступним етапом реєстрації є підтвердження пошти через SMTP-сервіс Mailtrap (рисуюнок 3.5).



Рисуюнок 3.5 – Повідомлення на пошті для активації акаунту

Необхідно перейти по посиланню, яке прийшло на пошту для активації акаунту (рисуюнок 3.6).



Рисуюнок 3.6 – Повідомлення про успішне підтвердження пошти

Переходимо на сторінку авторизації, заповнюємо поля 'Email' та в полі 'Пароль' вводимо пароль, згенерований в листі про активацію акаунту на пошті (рисуюнок 3.7).

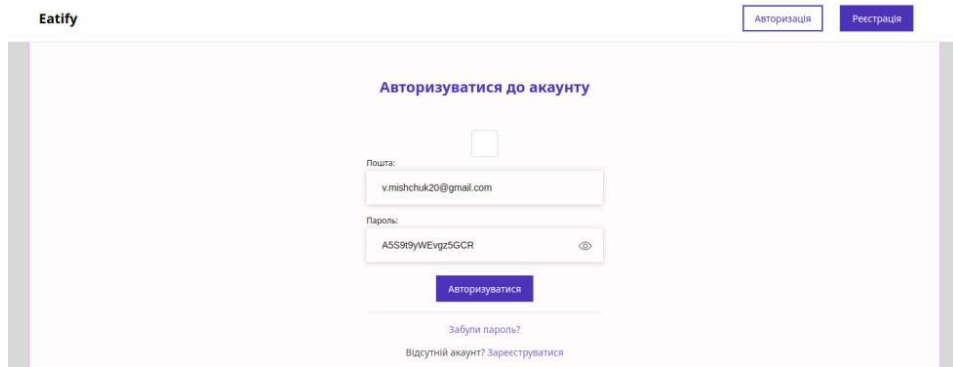


Рисунок 3.7 – Сторінка авторизації користувача

В базі даних можна побачити успішне додавання нового користувача із захешованим паролем та автоматичний запис його ролі – ROLE_CLIENT (рисунок 3.8).

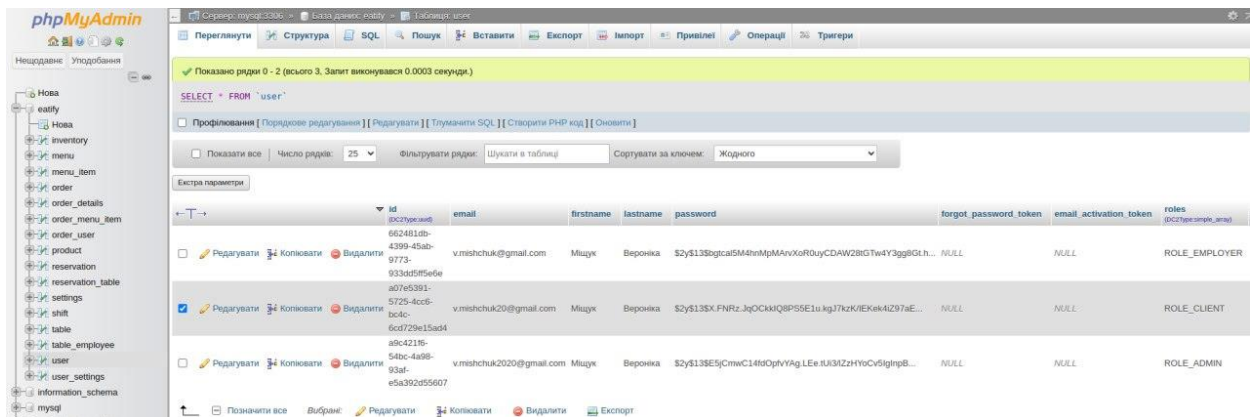


Рисунок 3.8 – Перевірка успішної реєстрації користувача в базі даних

Користувач входить в свій акаунт, де може змінювати такі дані як ім'я та прізвище (рисунок 3.9).

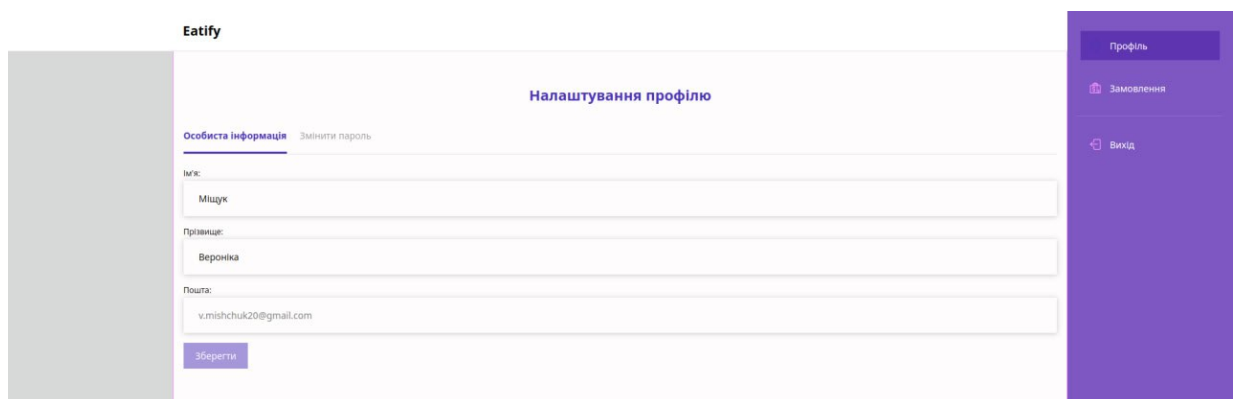


Рисунок 3.9 – Сторінка налаштування профілю користувача

Також на сторінці налаштування користувача є вкладка зі зміною пароля (рисунок 3.10).

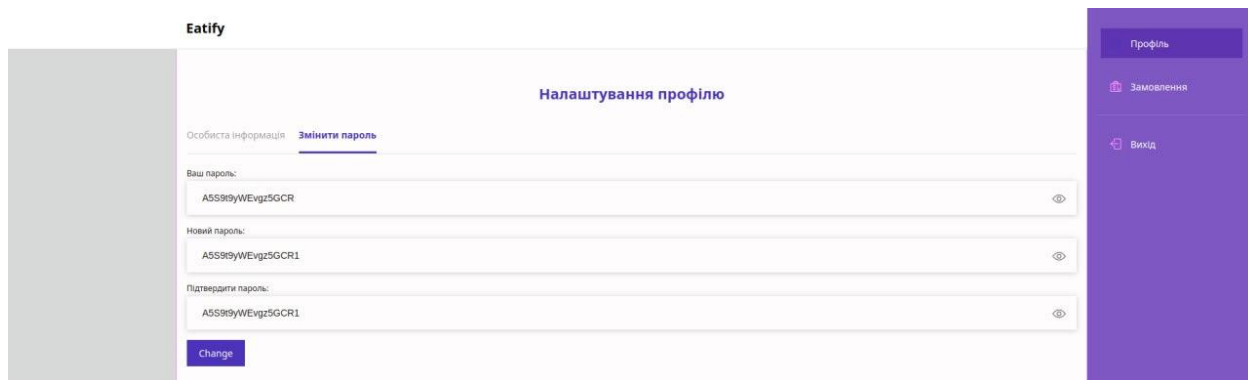


Рисунок 3.10 – Сторінка зміни пароля

Також у користувача є вкладка 'Замовлення', де він може створювати замовлення. Проте для створення замовлення потрібно мати заповнене стравами меню, це може робити тільки Адмін (рисунок 3.11).



Рисунок 3.11 – Сторінка замовлень користувача

Для заповнення меню, зайдемо в акаунт адміна, де одразу можна побачити всі доступні вкладки. Першою є вкладка зі списком всіх користувачів, можливістю редагування та додавання їх даних (рисунок 3.12).

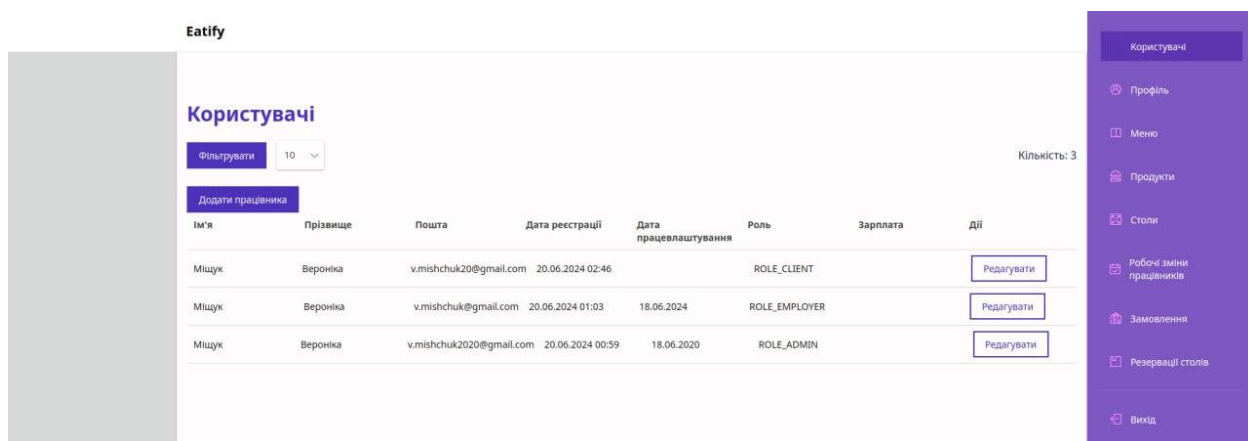


Рисунок 3.12 – Сторінка 'Користувачі' в адмін-панелі

Додавання нового користувача (рисунки 3.13 – 3.14).

Еатифі

Головна / Користувачі / Додавання користувача

Додати користувача

Пошта *

v.mishc@gmail.com

Ім'я *

Міщук

Прізвище *

піка

Роль:

Офіціант

Дата прийняття на роботу:

13-06-2024

Заробітна плата, ₴ *

4

ЗБЕРЕГТИСЯ

Користувачі

Профіль

Меню

Продукти

Столи

Робочі зміни працівників

Замовлення

Резервації столів

Вихід

Рисунок 3.13 – Сторінка додавання нового користувача

Еатифі

Головна / Користувачі / Користувачі

Користувачі

Фільтрувати 10

Додати працівника

Кількість: 4

Ім'я	Прізвище	Пошта	Дата реєстрації	Дата працевлаштування	Роль	Зарплата	Дії
Міщук	піка	v.mishc@gmail.com	20.06.2024 02:52	13.06.2024	ROLE_WAITER	4.00000000	Редагувати
Міщук	Вероніка	v.mishchuk2@gmail.com	20.06.2024 02:46		ROLE_CLIENT		Редагувати
Міщук	Вероніка	v.mishchuk@gmail.com	20.06.2024 01:03	18.06.2024	ROLE_EMPLOYER		Редагувати
Міщук	Вероніка	v.mishchuk2020@gmail.com	20.06.2024 00:59	18.06.2020	ROLE_ADMIN		Редагувати

Рисунок 2.14 – Сторінка з успішно доданим користувачем
Редагуємо дані користувача (рисунки 3.15 – 3.16).

Еатифі

Головна / Користувачі / Редагування користувача

Редагувати користувача

Пошта *

v.mishc@gmail.com

Ім'я *

Міщук

Прізвище *

піка

Роль:

Офіціант

Дата прийняття на роботу:

13-06-2024

Заробітна плата, ₴ *

5000

ЗБЕРЕГТИ

Користувачі

Профіль

Меню

Продукти

Столи

Робочі зміни працівників

Замовлення

Резервації столів

Вихід

Рисунок 3.15 – Сторінка редагування даних користувача

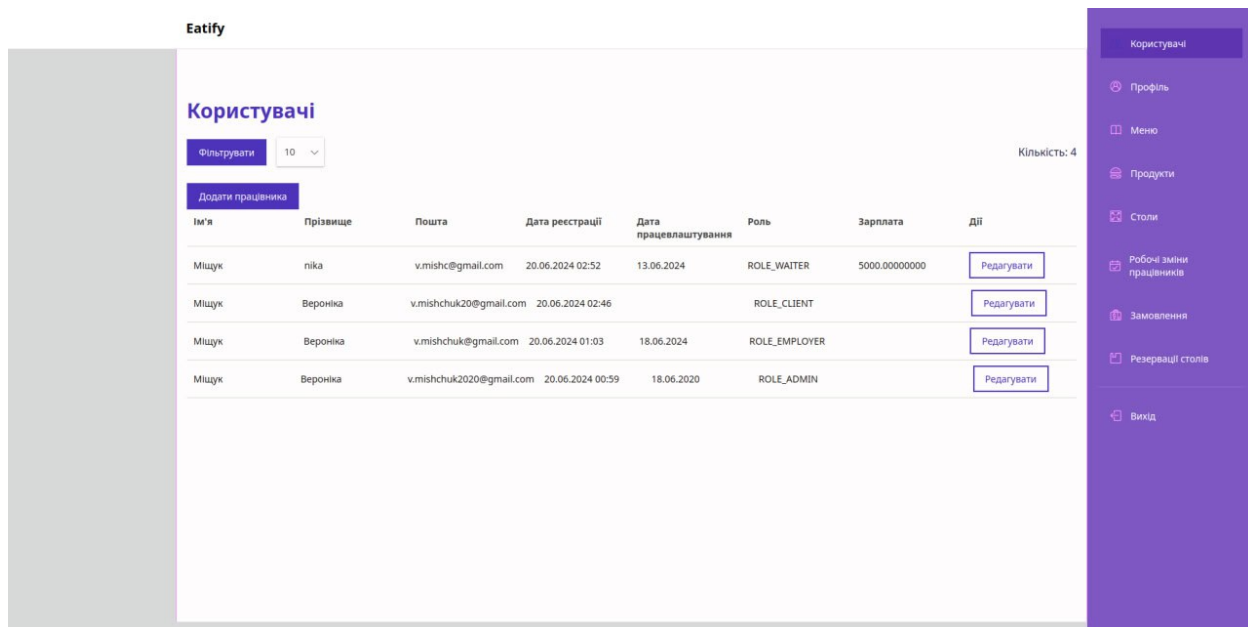


Рисунок 3.16 – Сторінка з успішно відредагованими даними користувача
Адміністратор, як і всі користувачі може налаштовувати свій профіль
(рисунки 3.17 – 3.18).

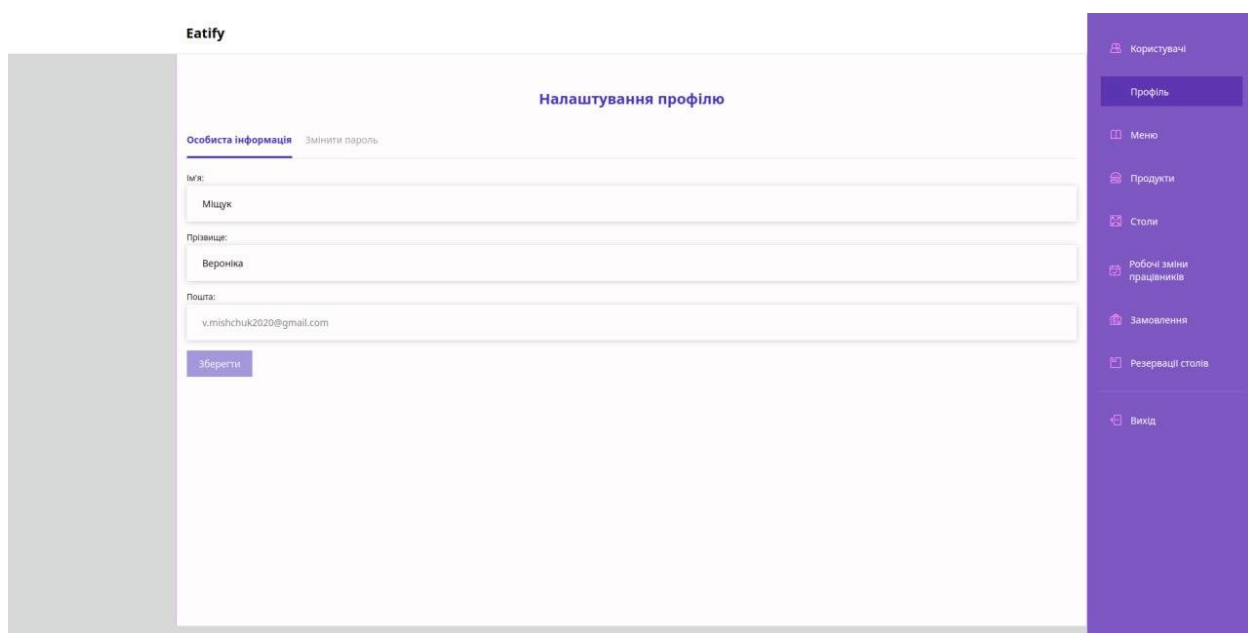


Рисунок 3.17 – Сторінка налаштування профілю

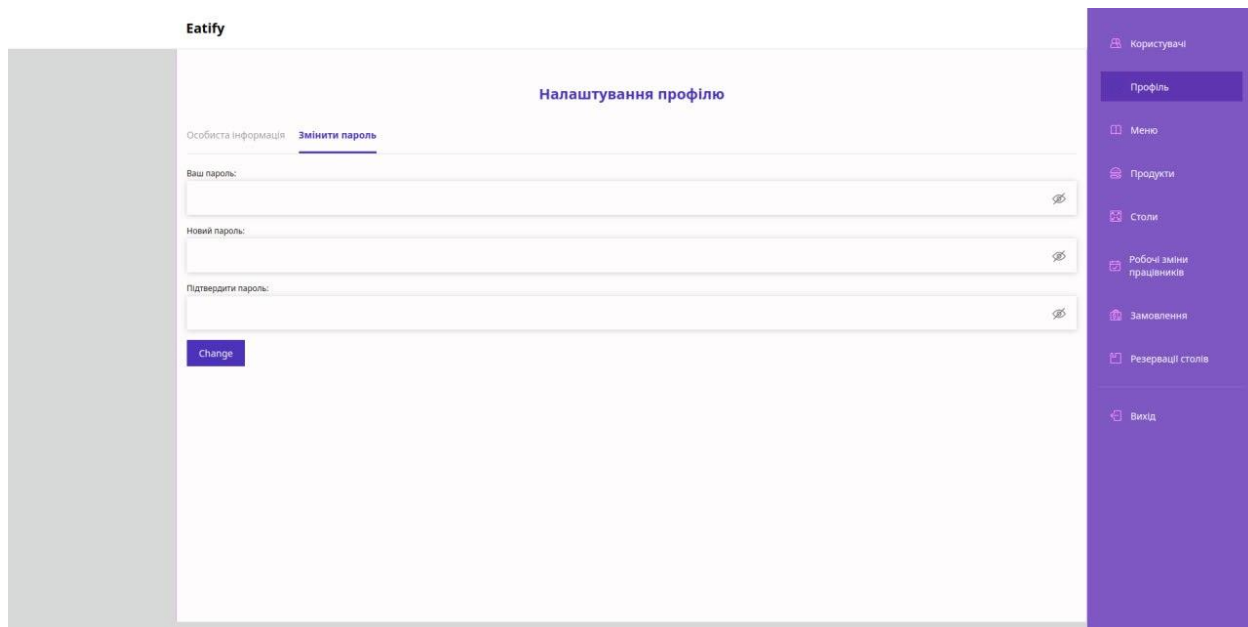


Рисунок 3.18 – Сторінка зміни паролю

На сторінці продуктів бачимо продукти з усіма властивостями, а також можливістю додавати нові продукти, редагувати та видаляти їх (рисунок 3.19).

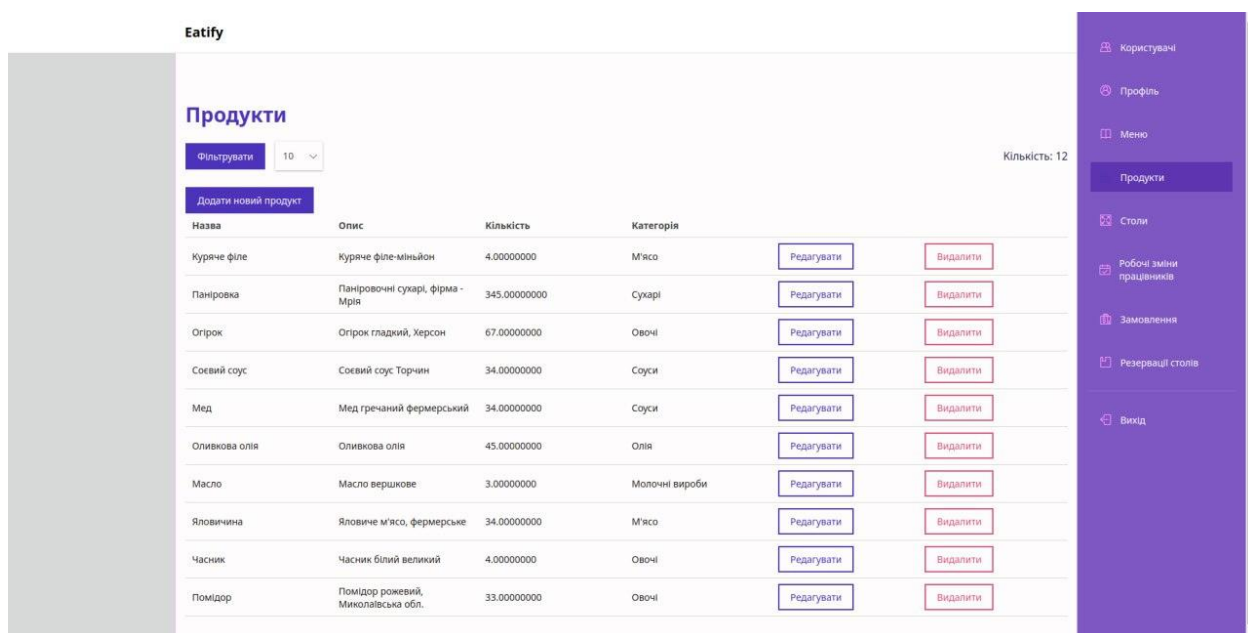


Рисунок 3.19 – Сторінка 'Продукти'

Додавання нового продукту (рисунки 3.20 – 3.21).

Eatify

Home / Account / Menu / Product / ProductCreate

Додати продукт

Назва продукту *

Шпінат

Категорія *

Овочі

Опис *

Шпінат бебі

Кількість на склад *

2

ЗБЕРЕГТИ

Користувачі

Профіль

Меню

Продукти

Столи

Робочі зміни працівників

Замовлення

Резервації столів

Вихід

Рисунок 3.20 – Додавання нового продукту

Eatify

Продукти

Фільтрувати

10

Додати новий продукт

Назва	Опис	Кількість	Категорія		
Куряче філе	Куряче філе-мінйон	4.00000000	М'ясо	Редагувати	Видалити
Паніровка	Паніровочні сухарі, фірма - Мрія	345.00000000	Сухарі	Редагувати	Видалити
Огірок	Огірок гладкий, Херсон	67.00000000	Овочі	Редагувати	Видалити
Соевий соус	Соевий соус Торчин	34.00000000	Соуси	Редагувати	Видалити
Шпінат	Шпінат бебі	2.00000000	Овочі	Редагувати	Видалити
Мед	Мед гречаний фермерський	34.00000000	Соуси	Редагувати	Видалити
Оливкова олія	Оливкова олія	45.00000000	Олія	Редагувати	Видалити
Масло	Масло вершкове	3.00000000	Молочні вироби	Редагувати	Видалити
Яловичина	Яловиче м'ясо, фермерське	34.00000000	М'ясо	Редагувати	Видалити
Часник	Часник білий великий	4.00000000	Овочі	Редагувати	Видалити

Кількість: 13

Користувачі

Профіль

Меню

Продукти

Столи

Робочі зміни працівників

Замовлення

Резервації столів

Вихід

Рисунок 3.22 – Сторінка з успішно доданим продуктом
Редагування продукту (рисунки 3.23 – 3.24).

Eatify

Home / Account / Menu / Product / ProductCreate

Редагувати

Назва продукту *

Шпінат

Категорія *

Овочі

Опис *

Шпінат бебі

Кількість на склад *

2

ЗБЕРЕГТИ

Користувачі

Профіль

Меню

Продукти

Столи

Робочі зміни працівників

Замовлення

Резервації столів

Вихід

Рисунок 3.23 – Редагування продукту

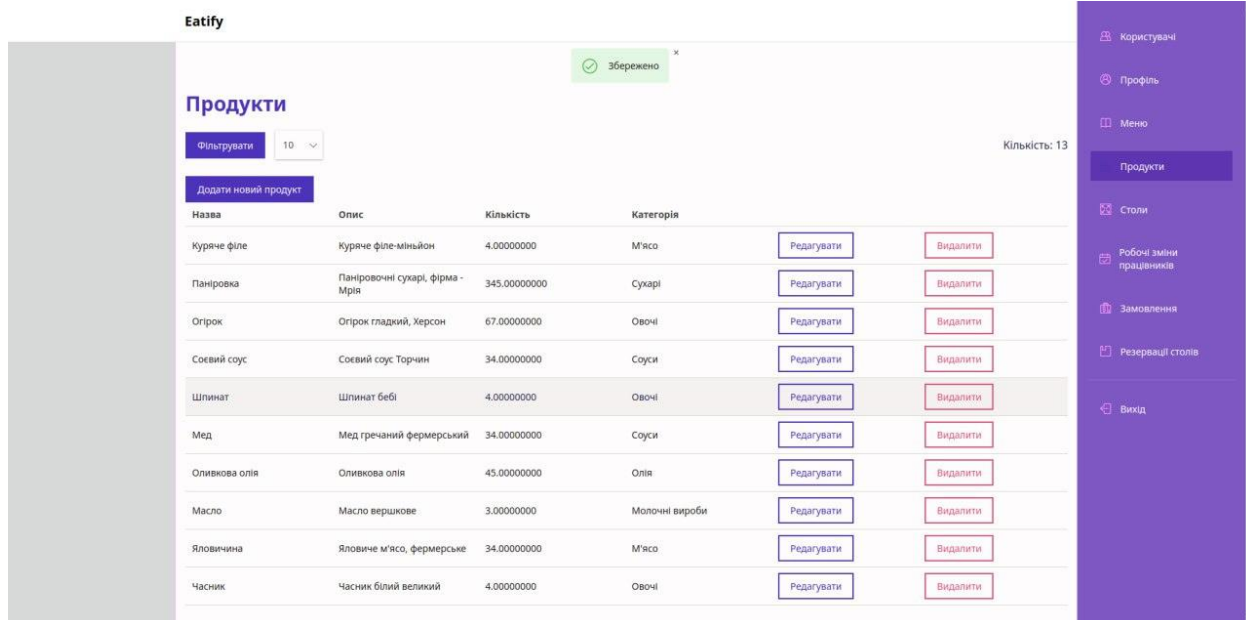


Рисунок 3.24 – Сторінка з успішно відредагованим продуктом
Видалення продукту (рисунки 3.25 – 3.26).

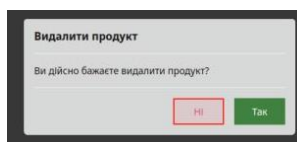


Рисунок 3.25 – Вікно підтвердження видалення продукту

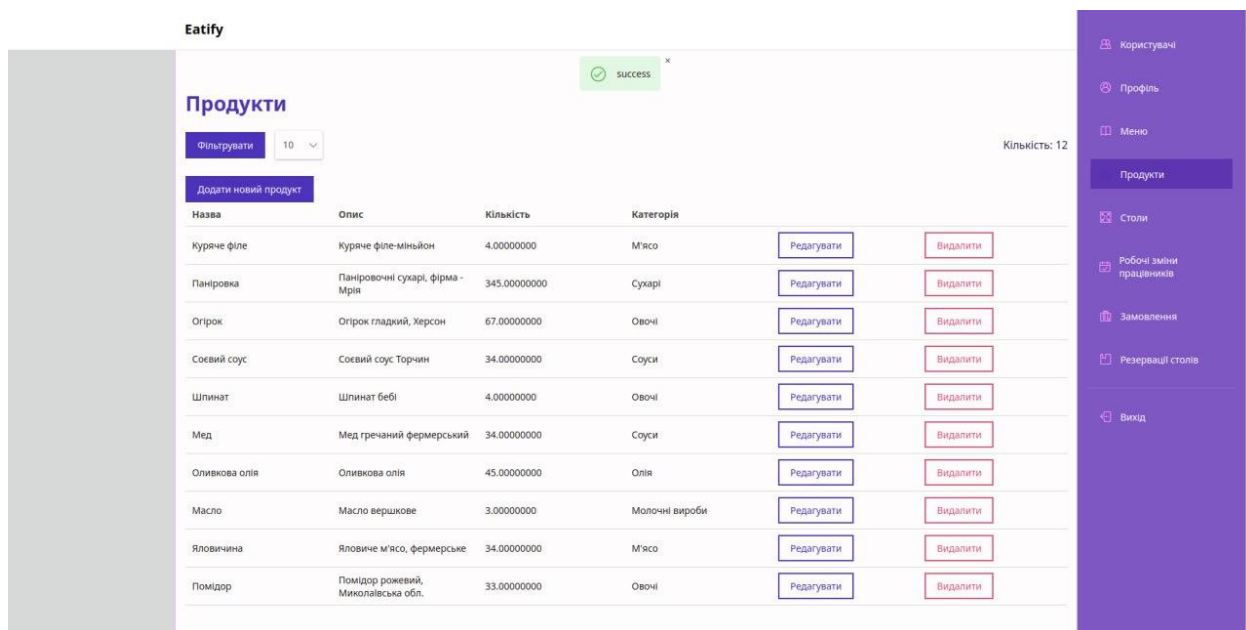


Рисунок 3.26 – Сторінка з успішно видаленим продуктом
Наступною є сторінка меню (рисунок 3.27).

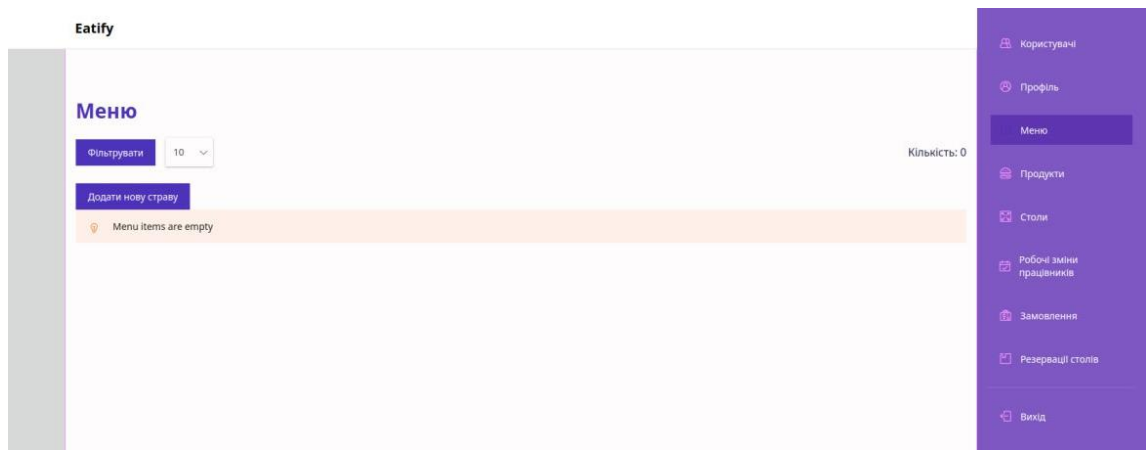


Рисунок 3.27 – Сторінка 'Меню'

Додаємо страву в меню з уже створеними раніше продуктами (рисунки 3.28 – 3.29).

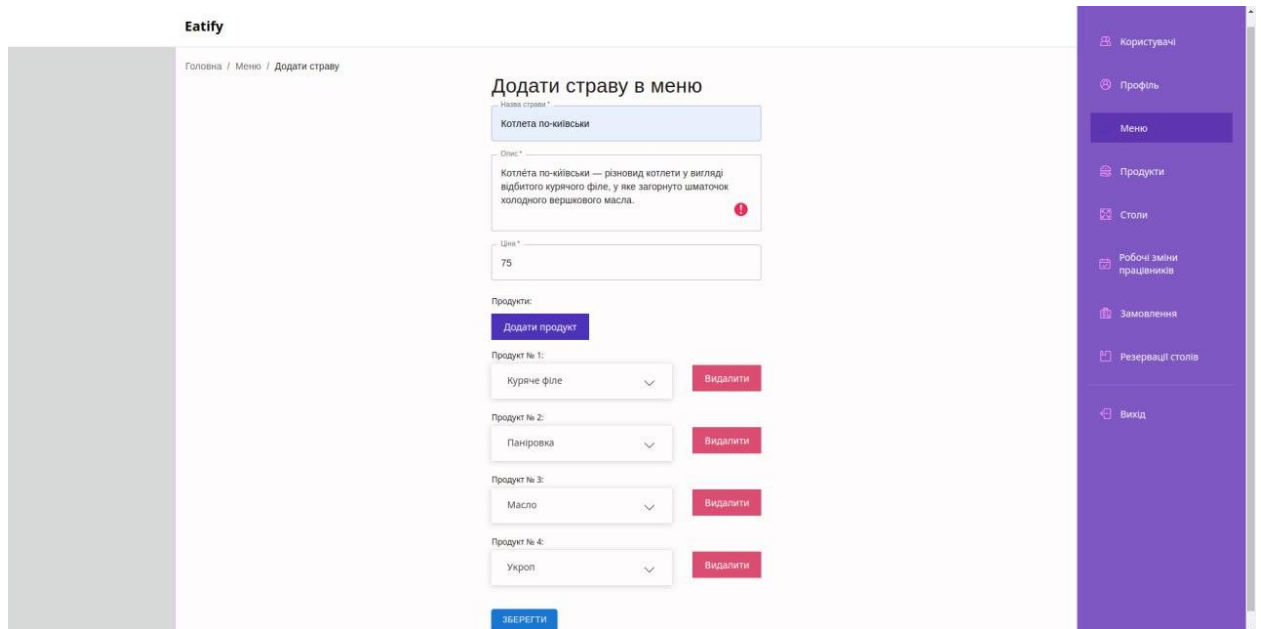


Рисунок 3.28 – Сторінка додавання страви в меню

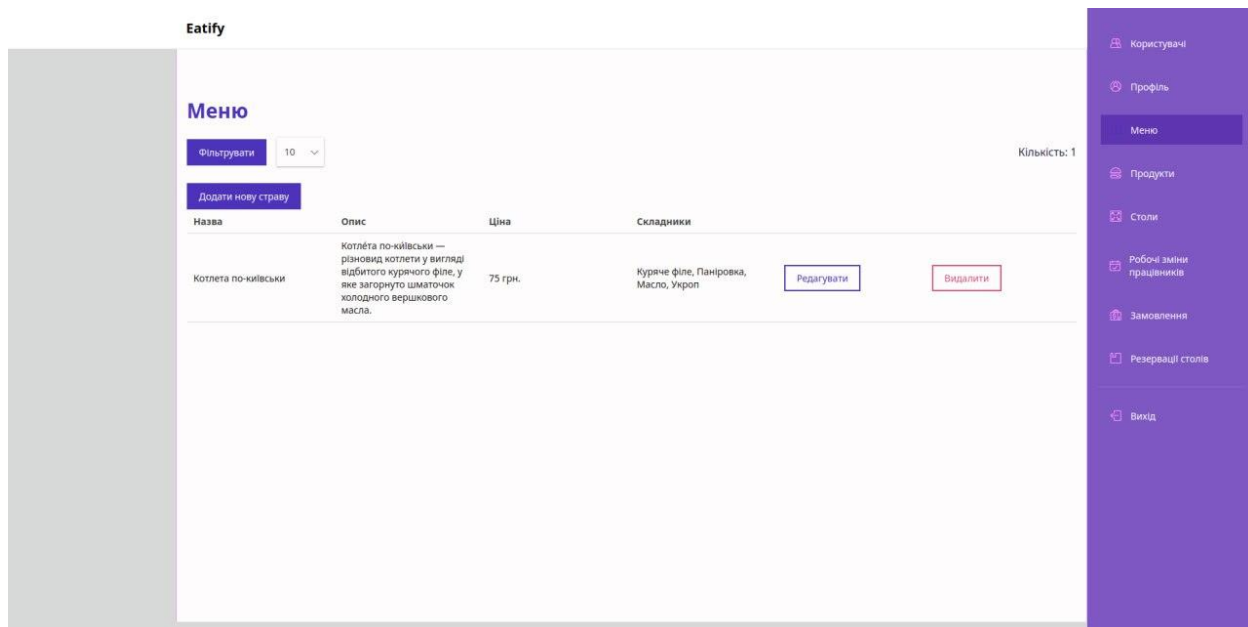


Рисунок 3.29 – Сторінка з успішно доданою стравою в меню
Редагування інформації про страву (рисунки 3.30 – 3.33).

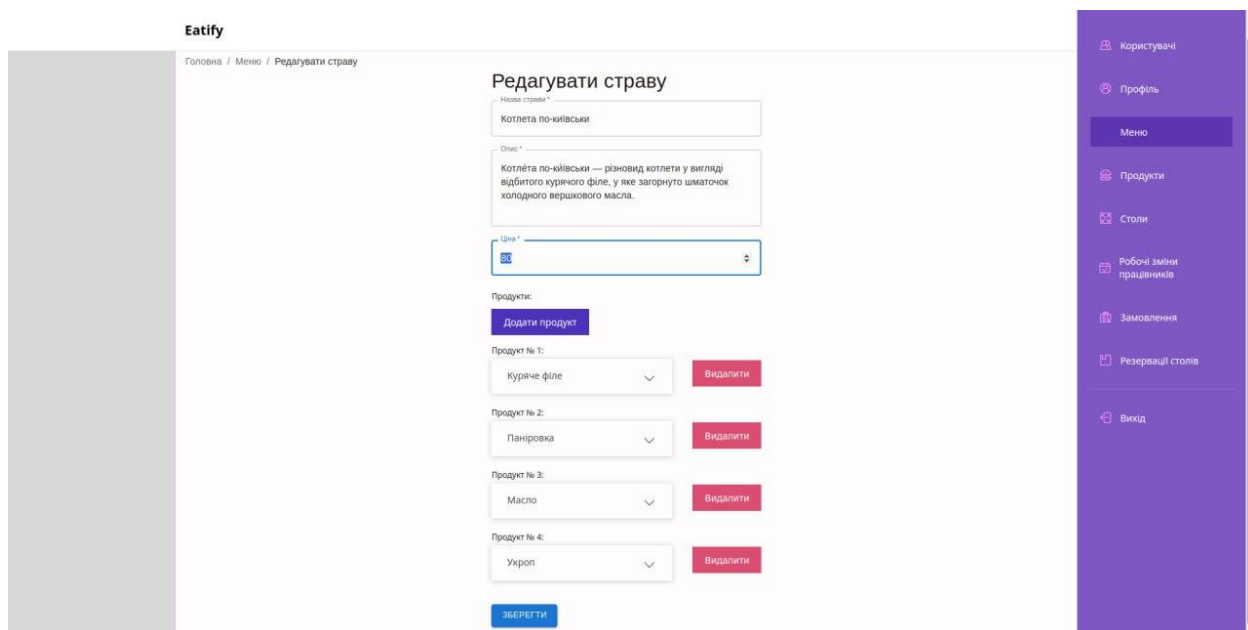


Рисунок 3.30 – Сторінка редагування страви в меню

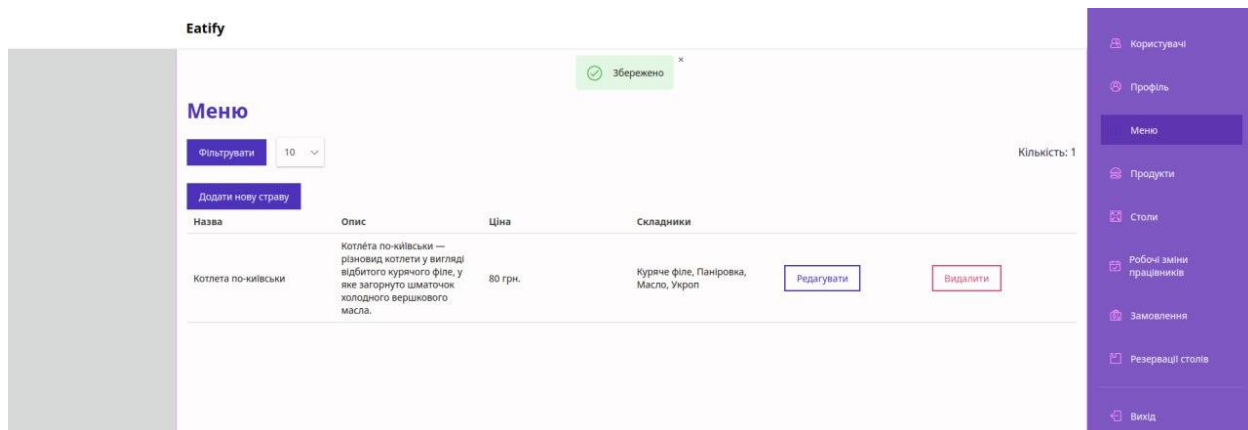


Рисунок 3.31 – Сторінка з успішно відредагованою стравою в меню

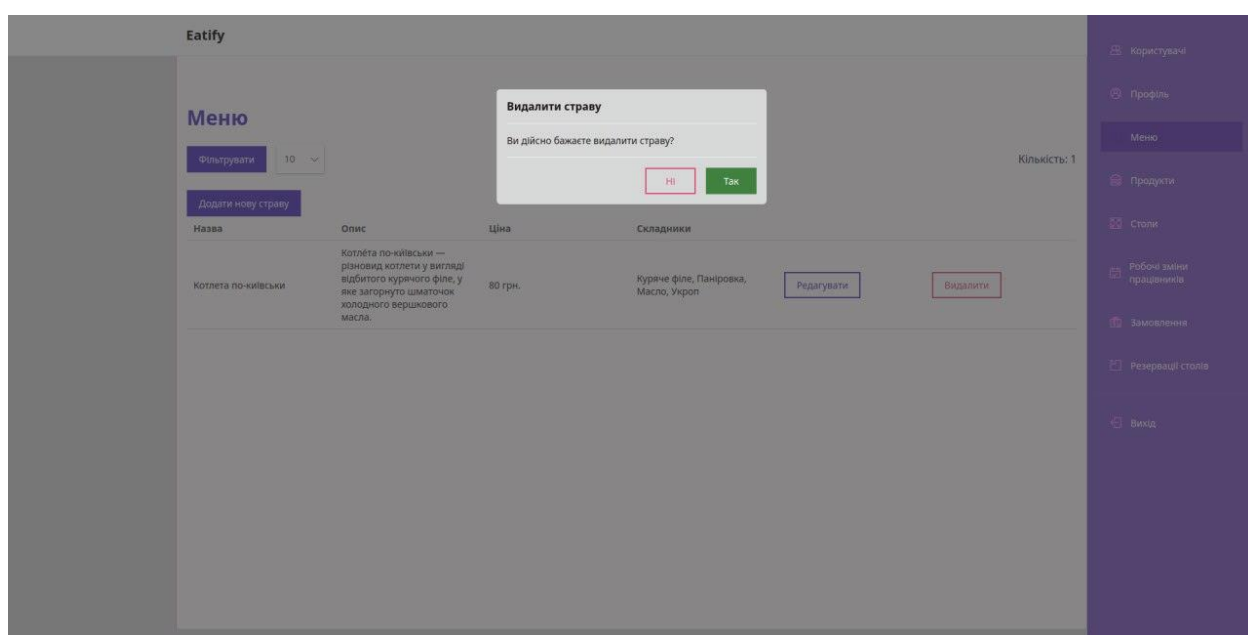


Рисунок 3.32 – Вікно підтвердження видалення страви



Рисунок 3.33 – Сторінка з успішно видаленою стравою в меню
Додавання, редагування та видалення столика (рисунки 3.34 – 3.40).

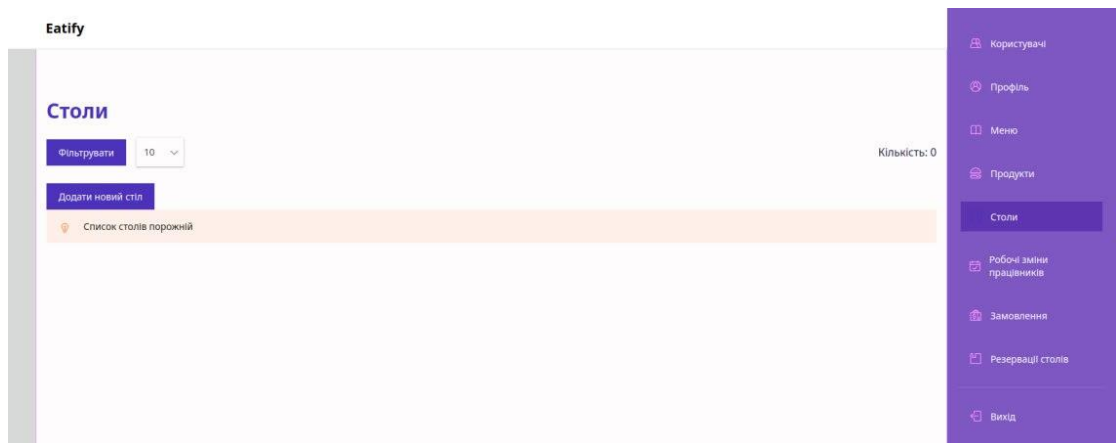


Рисунок 3.34 – Сторінка 'Столи'

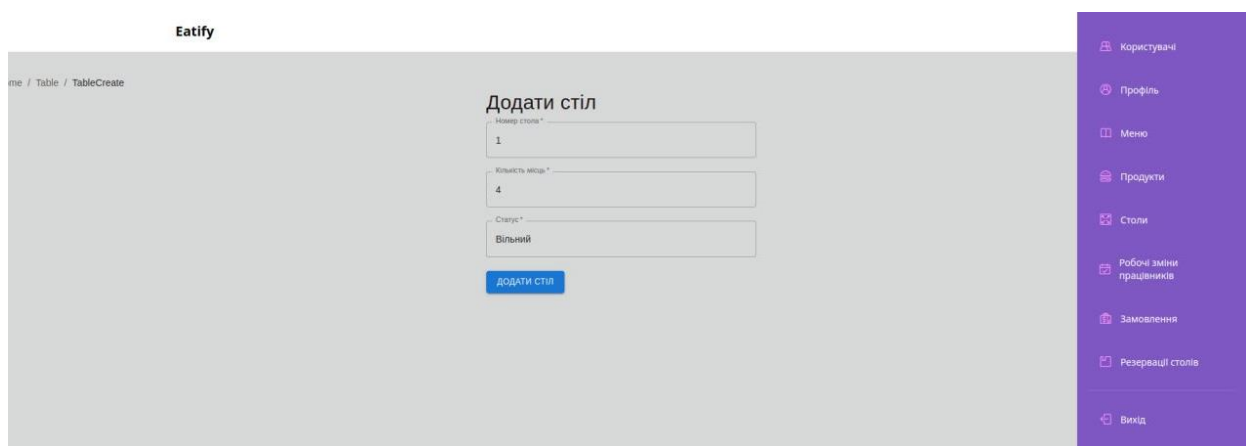


Рисунок 3.35 – Сторінка додавання стола

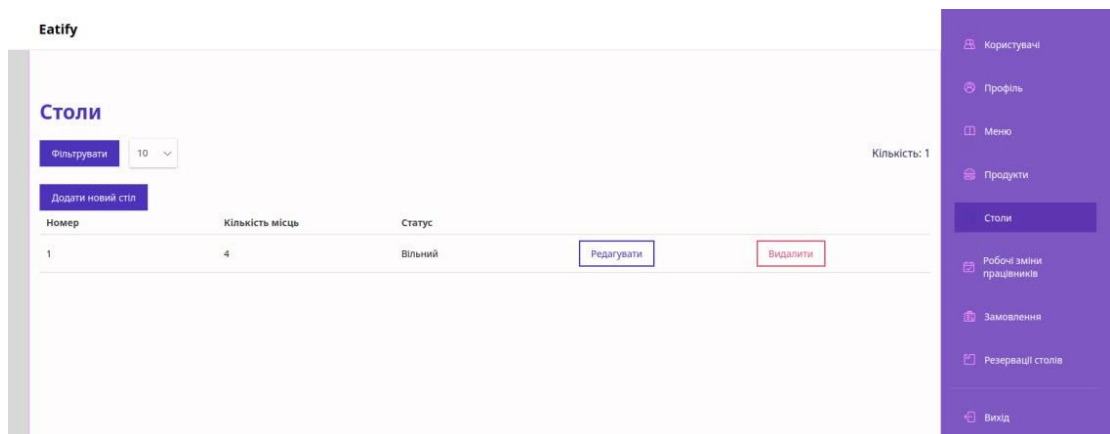


Рисунок 3.36– Сторінка з успішно доданим столом

Eatify

Додати стіл

Номер столу *

1

Кількість місць *

4

Статус *

Зайнятий

ДОДАТИ СТИЛ

Користувачі

Профіль

Меню

Продукти

Столи

Робочі зміни працівників

Замовлення

Резервації столів

Вихід

Рисунок 4.37 – Сторінка редагування стола

Eatify

Столи

Фільтрувати 10

Додати новий стіл

Номер

Кількість місць

Статус

1

4

Зайнятий

Редагувати

Видалити

Кількість: 1

Користувачі

Профіль

Меню

Продукти

Столи

Робочі зміни працівників

Замовлення

Резервації столів

Вихід

Рисунок 5.38 – Сторінка з успішно відредагованим столом

Eatify

Столи

Фільтрувати 10

Додати новий стіл

Номер

Кількість місць

Статус

1

4

Зайнятий

Редагувати

Видалити

Кількість: 1

Користувачі

Профіль

Меню

Продукти

Столи

Робочі зміни працівників

Замовлення

Резервації столів

Вихід

Рисунок 6.39– Вікно підтвердження видалення стола

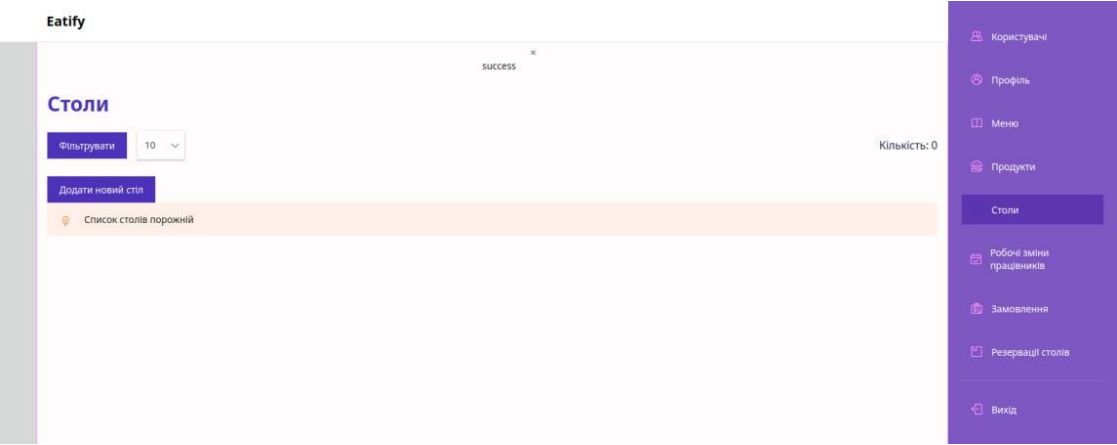


Рисунок 7.40 – Сторінка з успішно видаленим столом

Додавання, редагування та видалення робочих змін працівника (рисунки 3.41 – 3.46).

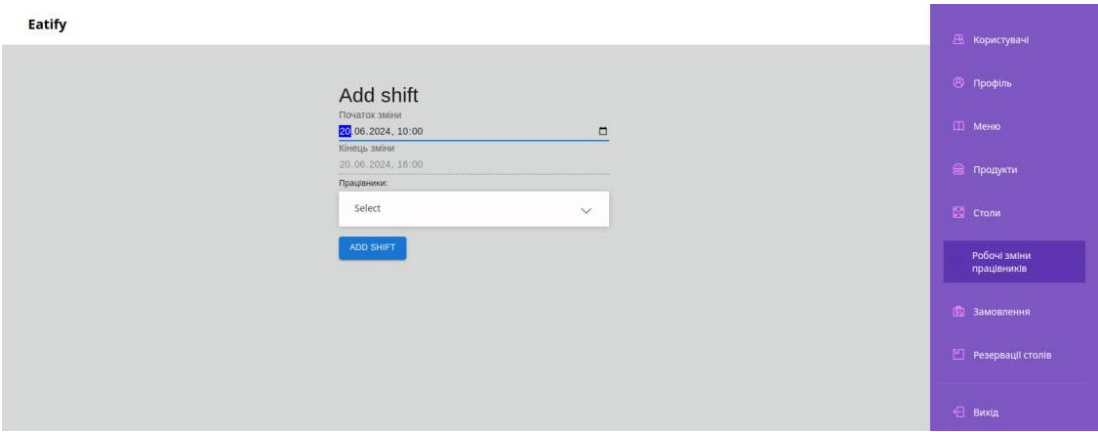


Рисунок 8.41 – Сторінка додавання робочої зміни

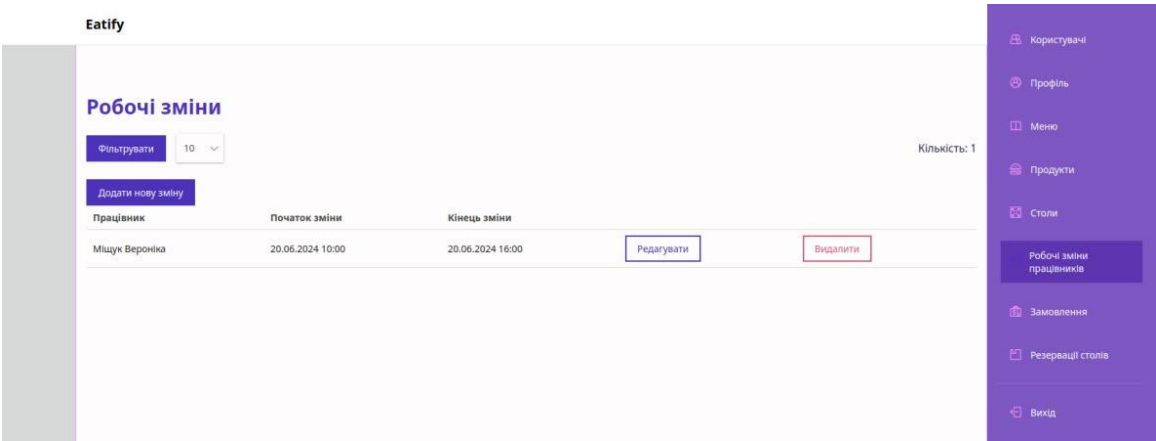


Рисунок 3.42 – Сторінка з успішно доданою робочою зміною

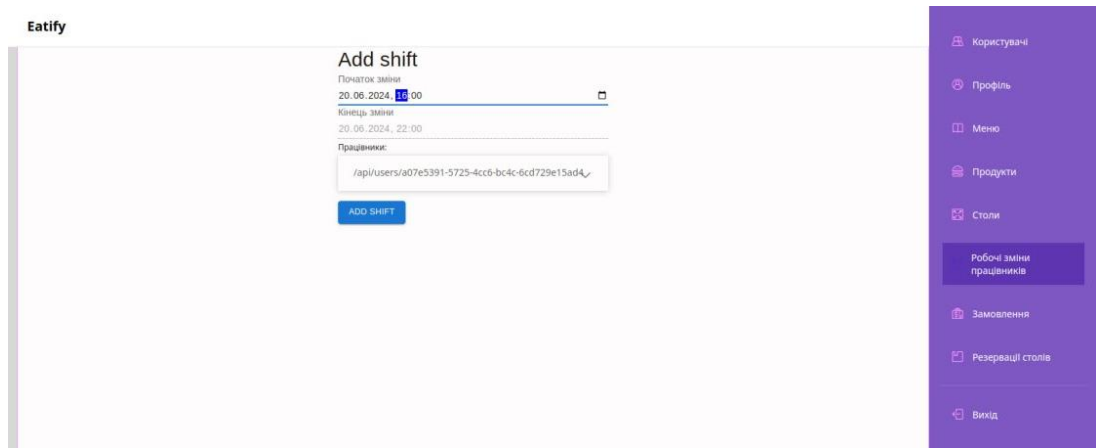


Рисунок 3.43 – Сторінка редагування робочої зміни

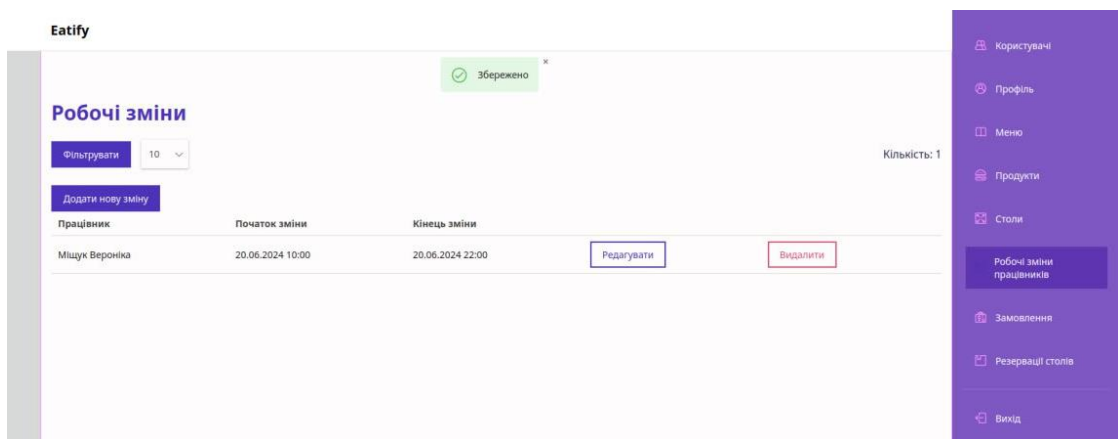


Рисунок 3.44 – Сторінка з успішно відредагованою робочою зміною

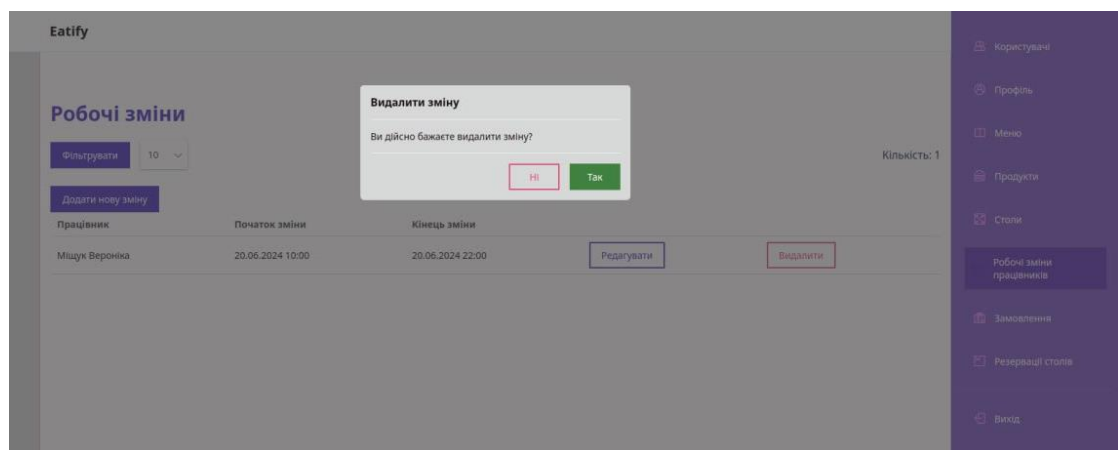


Рисунок 3.45 – Вікно підтвердження видалення робочої зміни

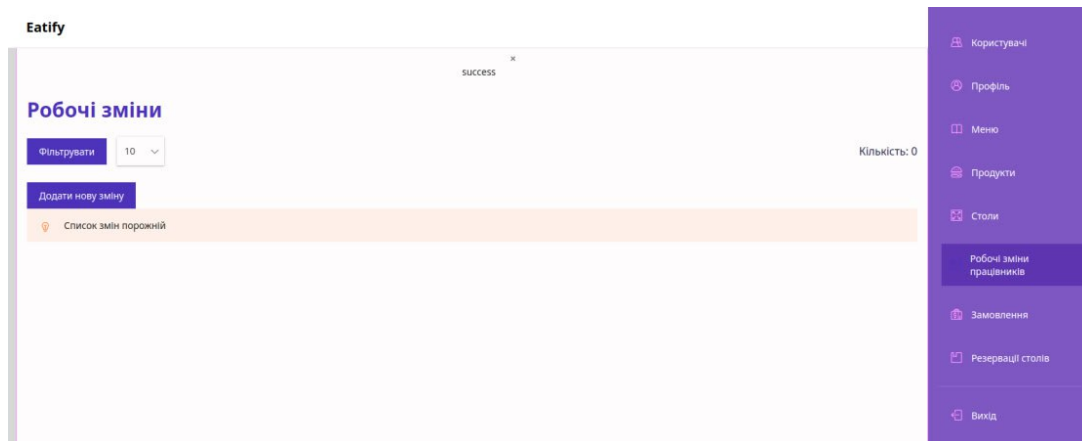


Рисунок 3.46 – Сторінка з успішно робочою зміною

Додавання, редагування та видалення резервації резервацій (рисунки 3.47 – 3.53).

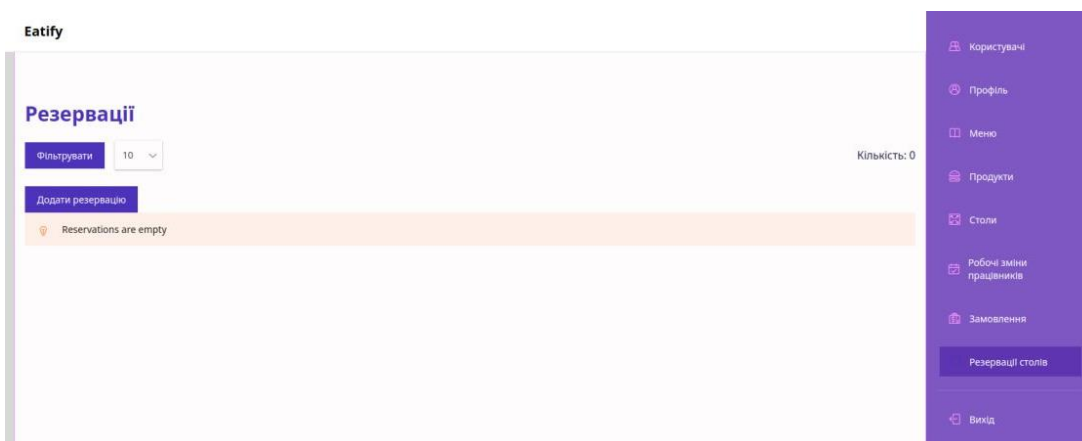


Рисунок 3.47 – Сторінка 'Резервації'

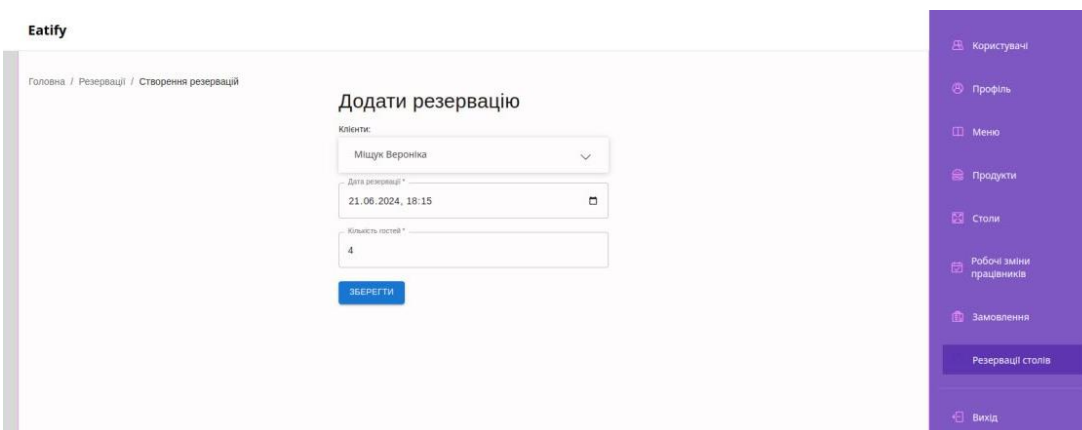


Рисунок 3.48 – Сторінка додавання резервації

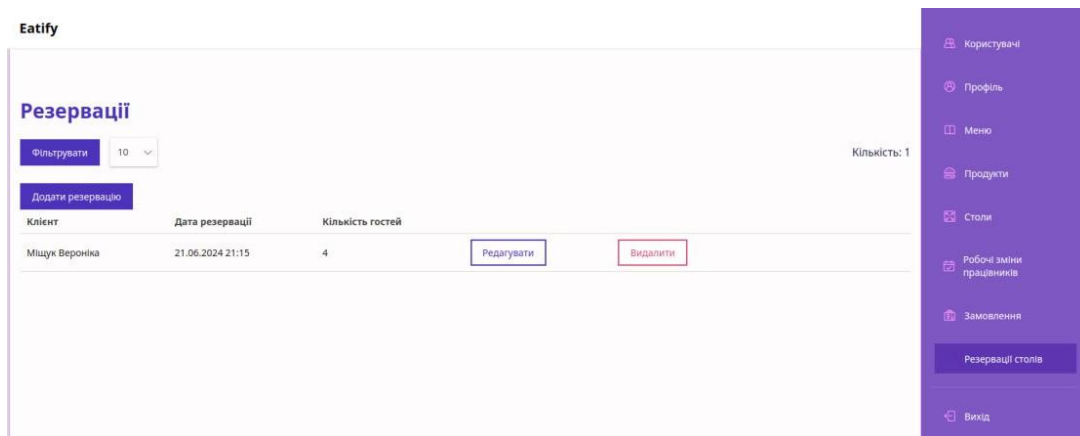


Рисунок 3.49 – Сторінка з успішно доданою резервацією

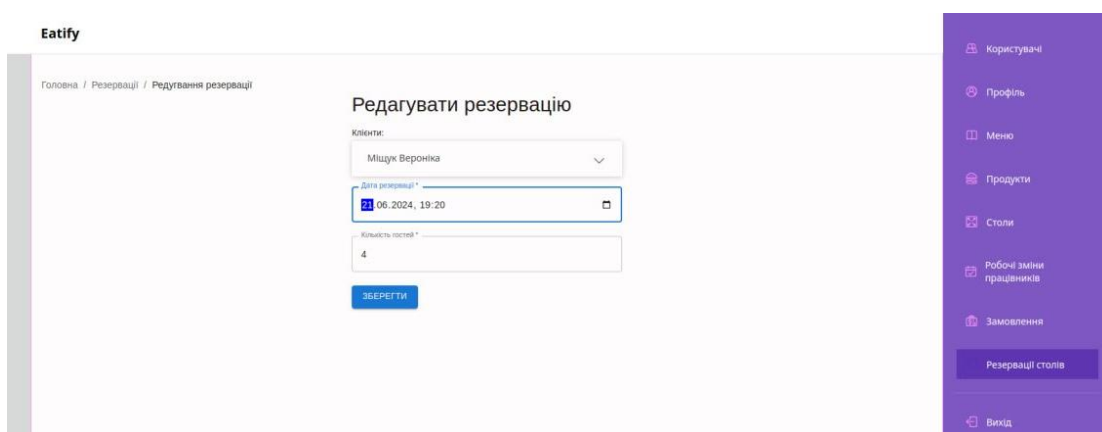


Рисунок 3.50 – Сторінка редагування резервації

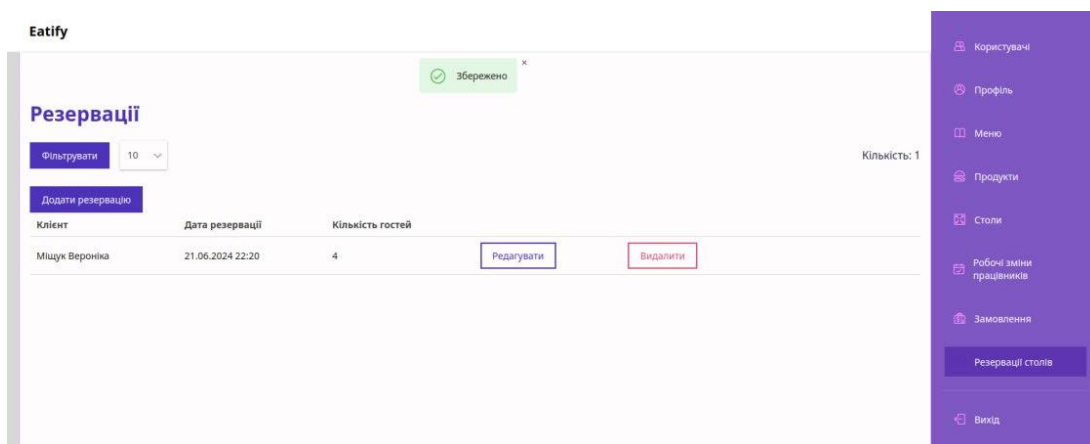


Рисунок 3.51 – Сторінка з успішно відредагованою резервацією

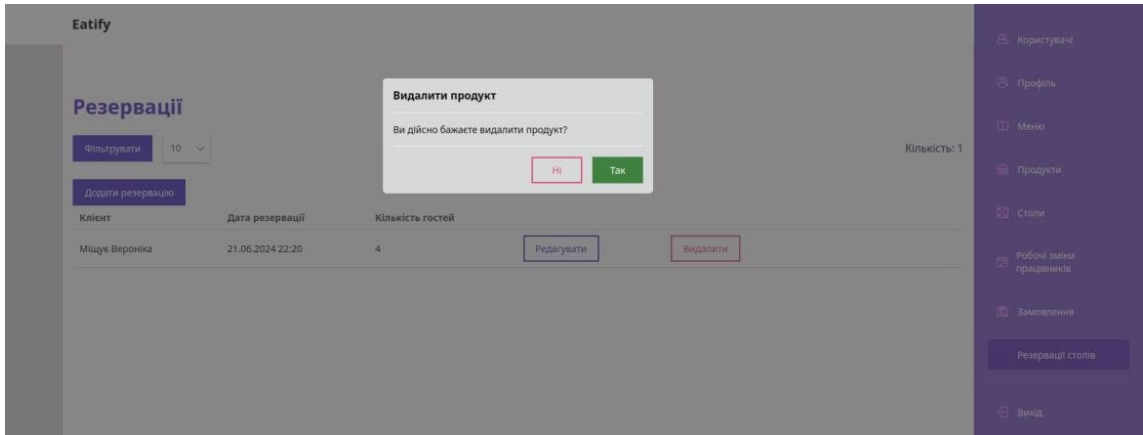


Рисунок 3.52 – Вікно підтвердження видалення резервації

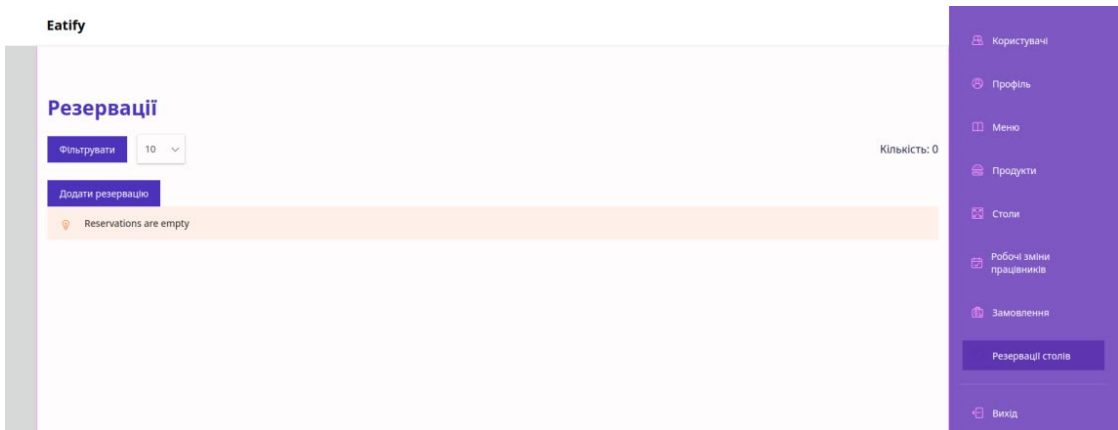


Рисунок 3.53 – Сторінка з успішно видаленою резервацією

По прикладу додаємо страв в меню (рисунок 3.54).

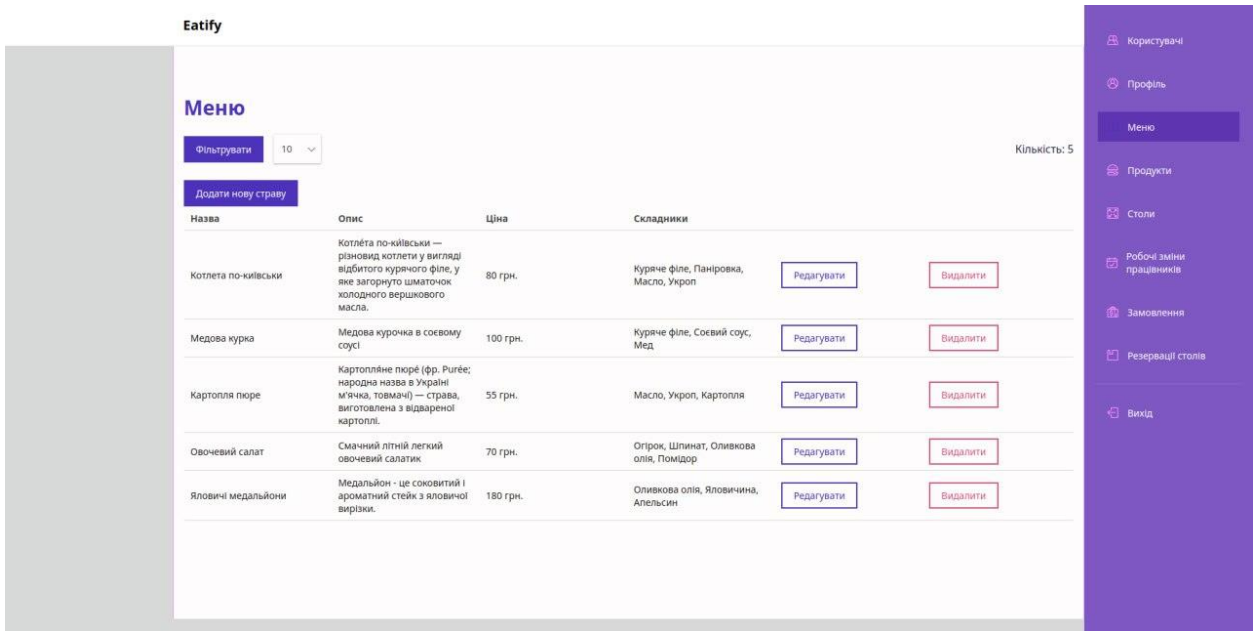


Рисунок 3.54 – Додані страви в меню

Заходимо в акаунт клієнта та створюємо нове замовлення (рисунок 3.55).



Рисунок 3.55 – Сторінка 'Замовлення'

Обираємо страви з меню (рисунок 3.56).

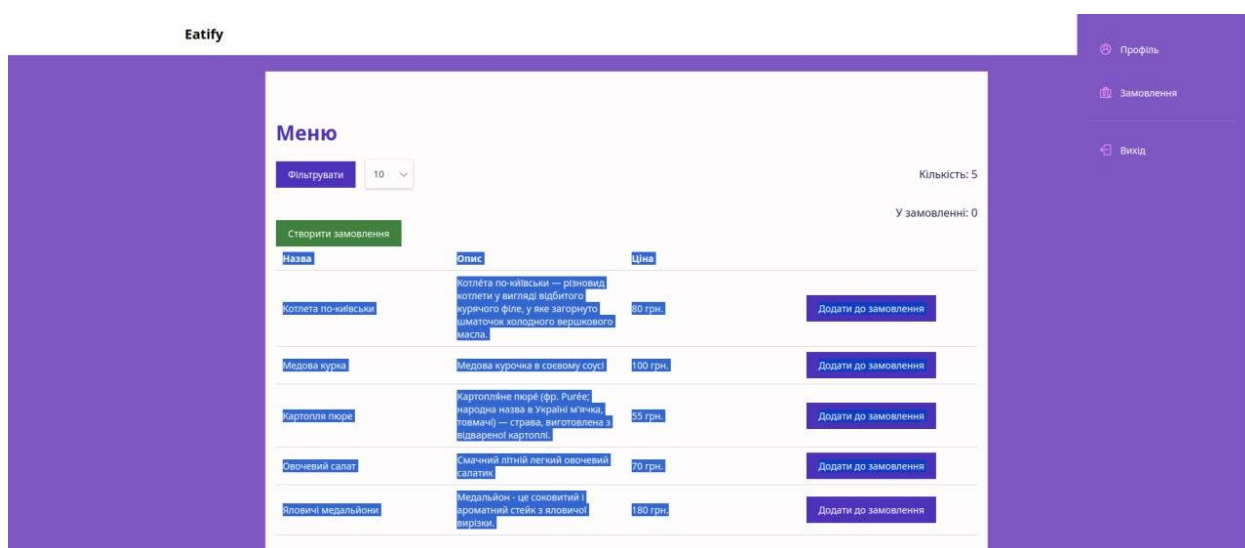


Рисунок 3.56 – Сторінка 'Меню'

Підтверджуємо замовлення (рисунок 3.57).



Рисунок 3.57 – Підтвердження замовлення

Замовлення з'являється в кабінеті користувача (рисунок 3.58).

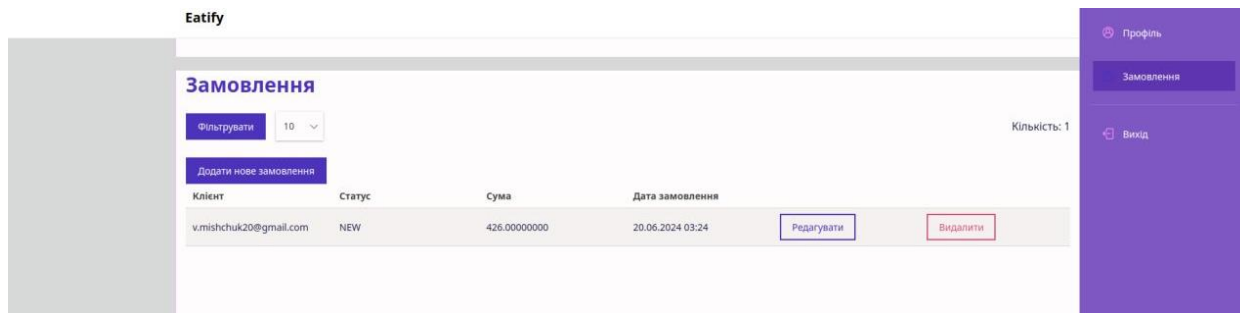


Рисунок 3.58 – Сторінка із щойно створеним замовленням

У адміна на панелі також додалось новостворене замовлення (рисунок 3.59).

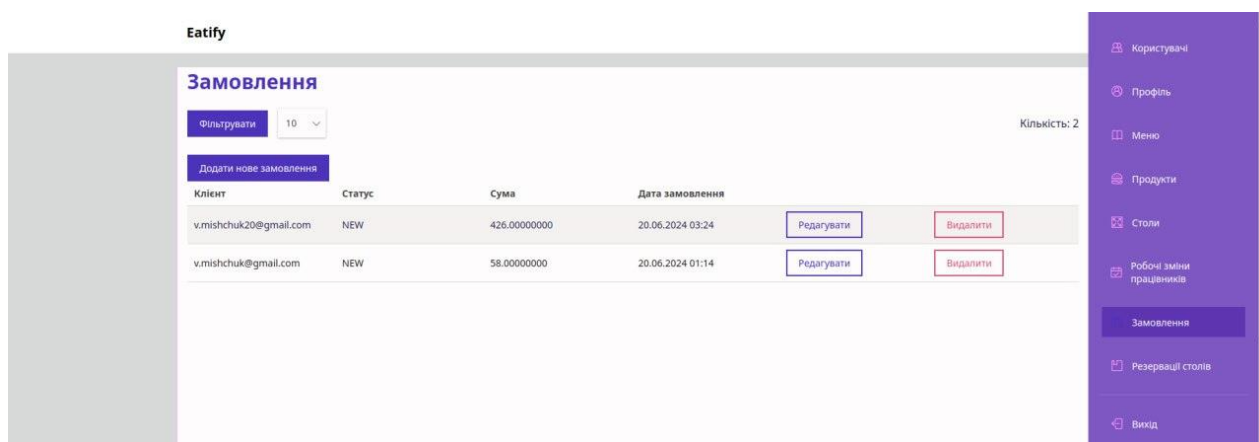


Рисунок 3.59 – Щойно створене замовлення в адмін-панелі

3.3 Тестування системи

Для перевірки загальної роботи додатку було використано техніки мануального тестування. Критичні помилки було виявлені та виправлені.

Перевірка на мобільну адаптацію сайту (рисунок 3.60).

Рисунок 3.60 – Мобільна адаптація сайту

Перевірка виводу помилок при неправильному вводі паролю при вході (рисунок 3.61).

Рисунок 3.61 – Вивід помилки при неправильному вводі пароля
Тестування можливості зміни пароля (рисунок 3.62 – 3.65).

Рисунок 3.62 – Сторінка 'Забули пароль?'

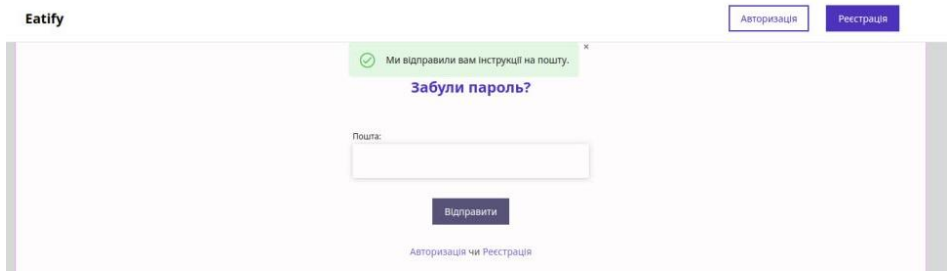


Рисунок 3.63 – Відправка інструкції по відновленню пароля

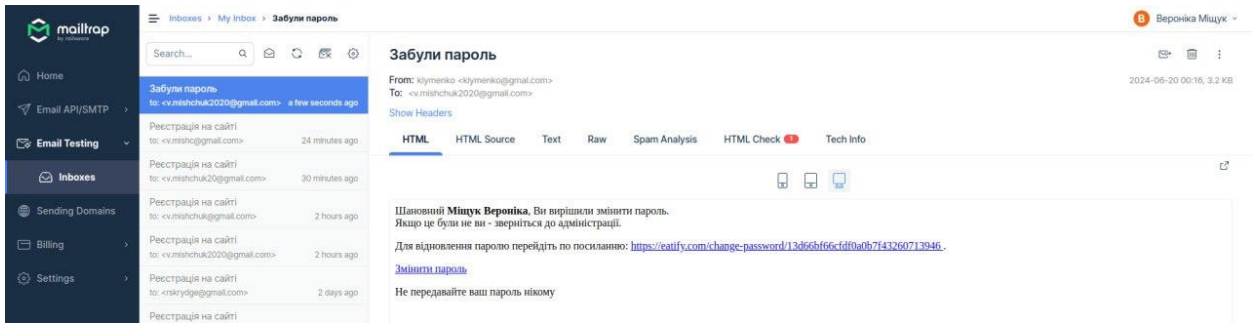


Рисунок 3.64 – Повідомлення на пошті про відновлення пароля

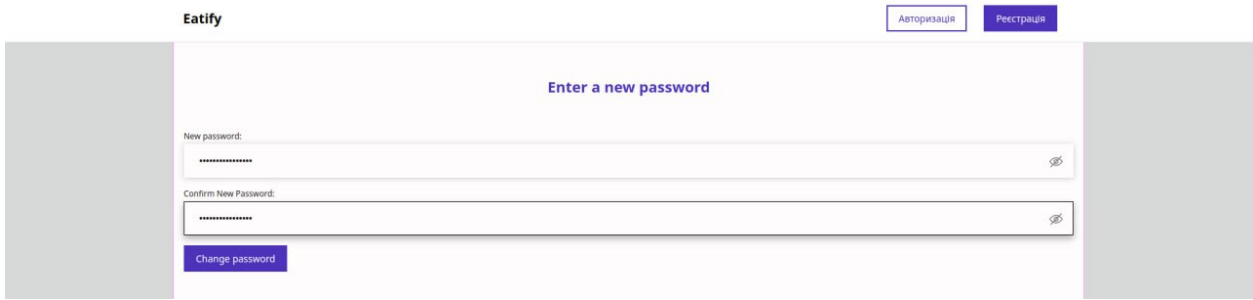


Рисунок 3.65 – Форма зі зміною пароля

Додатково, було підтверджено, що при невірному видаленні даних виводиться вікно з підтвердженням перед видаленням. Ці кроки демонструють впевненість у функціональності та коректності роботи системи після її розробки та впровадження.

ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ

У розділі детально розглянуто інтерфейс та процеси роботи з автоматизованою системою управління закладом харчування. Починаючи з діаграми розгортання, яка ілюструє архітектурні компоненти системи, показано взаємодію між користувацькою аплікацією, робочою станцією та серверною аплікацією. Це дозволило нам проаналізувати потрібну апаратну конфігурацію та забезпечити взаємодію компонентів через TCP/IP.

Описавши структуру інтерфейсу, показано, як користувачі взаємодіють з системою через веб-інтерфейс. Від реєстрації нових користувачів до управління профілями та створенням замовлень, де враховано різноманітні потреби різних категорій користувачів. Крім того, продемонстровано адміністративні можливості, такі як управління продуктами, стравами та резерваціями, що забезпечують ефективне функціонування закладу.

Тестування системи підтвердило її надійність та стабільність після внесених коригувань, включаючи виправлення виявлених помилок та перевірку доступності та адаптивності для мобільних пристроїв. Цей розділ підкріплює впевненість у функціональності системи та готовності до її впровадження.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		64

ВИСНОВКИ

Дана кваліфікаційна робота є результатом аналізу вимог користувачів та ретельної розробки веб-орієнтованої системи. Під час цього процесу не лише визначено основні функції та вимоги до додатку, а й враховано сучасні тренди у веб-розробці та потреби користувачів.

Особливу увагу приділено проектуванню архітектури системи та її компонентів, забезпечивши оптимальну взаємодію між ними. Встановлення чітких залежностей та структурування бази даних дозволило забезпечити ефективне управління даними та високу швидкодію системи.

Процес тестування виявився важливим етапом, де перевірено не лише функціональність програмного забезпечення, а й його надійність та зручність використання. Застосування різних методів тестування, включаючи автоматизоване та мануальне тестування, дозволило виявити та виправити потенційні проблеми ще на стадії розробки.

У результаті було створено готовий до використання програмний продукт з інтуїтивно зрозумілим інтерфейсом, який відповідає потребам наших користувачів. План подальшого розвитку системи забезпечить її вдосконалення та адаптацію до змінюючихся умов ринку.

Цей проект підвищив навички у всіх етапах розробки веб-додатків. Він став важливим етапом у професійній підготовці, дозволяючи вдосконалити знання та досвід у сфері веб-розробки та інформаційних технологій.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		65

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Коротун О.В., Сугоняк І.І., Міщук В.С. Проектування та використання бази даних у сфері обслуговування. Тези IV Всеукраїнської науково-технічної конференції «Комп'ютерні технології: інновації, проблеми, рішення», 18-20 листопада 2021 року. Житомир: “Житомирська політехніка”, 2021. С. 50-51
2. Міщук В.С., Марчук Г.В. Огляд сучасних інструментів для розробки веб-орієнтованих систем управління закладами харчування на базі PHP з використанням фреймворку SYMFONY. Тези XIV Міжнародної науково-технічної конференції «Інформаційно-комп'ютерні технології - 2024», 28-29 березня 2024 року. Житомир: “Житомирська політехніка”, 2024. С. 108-109
3. Міщук В.С., Вольський Р.А. Оптимізація розробки програмного забезпечення у сфері управління закладом харчування за допомогою систем контролю версій. Тези XVII Міжнародної науково-практичної конференції «Інтегровані інтелектуальні робототехнічні комплекси ІРТК-2024», 21-22 травня 2024 року. Київ: “Національний авіаційний університет”, 2024. С. 394-395
4. Entity Framework Core [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: <https://learn.microsoft.com/en-us/ef/core/>.
5. Маслова, В. М. Управління персоналом: монографія / за ред. В. М. Маслова. К.: Центр учбової літератури, 2014. 492 с
6. Курс php : У 4 т. / Акад. прав. наук України ред. О.Д. Святоцького. – К.: Вид. дім «І Юра», 2008. – Т. 1-4.
7. React.JS / Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/>
8. Бізнес-планування : навч. посіб. / Т. Г. Васильців, Я. Д. Качмарик, В. І. Блонська, Р. Л. Лупак. – К. : Знання, 2013. – 173 с.

9. Батенко Л. П. Управління проектами : [навч. посібник] / Батенко Л. П., Афанасьєв М. В., Гонтарева І. В. – К. : КНЕУ, 2004. – 231 с.
10. Демченко А. В. Система управління IoT ресторану «Антрекот» : автореф. дип. роботи на здобуття освітнього ступеня «бакалавр» : напрям підготовки 6.050102 «Комп’ютерна інженерія» / А. В. Демченко, ЧНУ ім. Петра Могили. - Миколаїв, 2019. - 12 с.
11. Шишкіна К. М. Розробка електронного меню для ресторанів / К. М. Шишкіна // Молодий вчений. - 2014. - №4. - С. 24-27.
12. Symfony / Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://symfony.com.ua/>
13. Symfony - Open-source PHP framework [Електронний ресурс] – Режим доступу до ресурсу: <https://symfony.com/>
14. Git - Про систему контролю версій [Електронний ресурс] – Режим доступу до ресурсу: <https://git-scm.com/book/uk/v2>
15. UML Activity Diagram Tutorial [Електронний ресурс] – Режим доступу до ресурсу: <https://www.lucidchart.com/pages/uml-activity-diagram>
16. Class Diagram [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>
17. Sanders W. Learning PHP Design Patterns / William Sanders. – United States of America, 2013. – 340 с. – (O’Reilly Media).
18. Грицунов О. В. Інформаційні системи та технології. Навчальний посібник. — Х.: ХНАМГ, 2010. — 222 с.
19. А.Зандстра, Ф.Матт. PHP Objects, Patterns, and Practice, 2013. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.apress.com/la/book/9781430260325>
20. Activity Diagram [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>

ДОДАТКИ

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг index.js:

```
import React from "react";
import ReactDOM from "react-dom";
import { BrowserRouter } from "react-router-dom";
```

```
import App from "./App";
import reportWebVitals from "./reportWebVitals";
```

```
ReactDOM.render(
  <BrowserRouter>
    <React.StrictMode>
      <App />
    </React.StrictMode>
  </BrowserRouter>,
  document.getElementById("root")
);
```

Лістинг App.js:

```
import React, { createContext, Suspense, useEffect, useState } from "react";
import { Redirect, Switch, useHistory } from "react-router-dom";
import { Helmet, HelmetProvider } from "react-helmet-async";
import axios from "axios";
import Header from "./components/elements/header/Header";
import AutoLogoutModal from "./components/elements/modal/AutoLogoutModal";
import PageSpinner from "./components/elements/spinner/PageSpinner";
import { getExpirationDate, isExpired } from "./utils/checkExpiredToken";
import RequestConfig from "./utils/requestConfig";
import getUserInfo from "./utils/getUserInfo";
import eventBus from "./utils/eventBus";

import LoadRoute from "./components/loadRouter/LoadRoute";
import routes from "./routes/routes";
import userRoutesConcat from "./routes/userRoutes";

import ".i18n";
import i18n from "i18next";
import { defaultLanguage, helmetHtmlByLang, supportedLanguages } from "./utils/consts";

import { ThemeProvider } from "styled-components";
import { mainTheme } from "./components/styles/theme/theme";

import { GlobalStyle } from "./components/styles/globalStyle";
import "./assets/fonts/default-icons/style.css";
import "./assets/css/rc-select.css";
import "rc-dialog/assets/index.css";
import "rc-notification/assets/index.css";
import "rc-dropdown/assets/index.css";
import "rc-checkbox/assets/index.css";
import "rc-switch/assets/index.css";
import "rc-pagination/assets/index.css";
import "rc-tabs/assets/index.css";
import "rc-drawer/assets/index.css";
import "rc-tooltip/assets/bootstrap.css";

export const AppContext = createContext({});

function App () {
```

```

window.localStorage.removeItem("appLang");

const history = useHistory();

const [locale, setLocale] = useState("");
const [visible, setVisible] = useState(false);
const [authenticated, setAuthenticated] = useState(localStorage.getItem("token") &&
!isExpired(getExpirationDate(localStorage.getItem("token"))));

const theme = "mainTheme";
const mountedComponent = true;
// const [theme, mountedComponent] = getTheme();

const currentUrl = new URL(document.location);
let lngFromUrl = defaultLanguage;

Object.keys(supportedLanguages).forEach(language => {
  let matches = currentUrl.href.match((^${currentUrl.origin.replaceAll("/",
"\\"\/")}${supportedLanguages[language].regex}));
  if (i18n.options.supportedLngs.includes(language) && matches && matches[0] !== "" && matches[1] !== "") {
    lngFromUrl = language
  }
})

const isValidToken = () => {
  axios.post("/api/is-valid-token", {}, RequestConfig(true)).catch(() => {
    localStorage.removeItem("token");
    setAuthenticated(false);
    history.push("/login");
  });
};

const handleOnIdle = () => {
  eventBus.on("logout", (data) => {
    if (data?.expired) {
      setVisible(true);
    }

    localStorage.removeItem("clientId");
    localStorage.removeItem("token");

    setAuthenticated(false);
    history.push("/");
  });
};

useEffect(() => {
  if (localStorage.getItem("token") !== null) {
    isValidToken();
  }
  handleOnIdle();
}, []);

const authRouteRender = () => {
  if (!authenticated) {
    return (
      routes.map((route, i) => {
        <LoadRoute key={i} {...route} />
      ))
    );
  }
};

```

```

    } else {
      return (
        userRoutesConcat.map((route, i) => (
          <LoadRoute key={i} {...route} />
        ))
      );
    }
  };
};

```

```
const themeMode = mainTheme;
```

```

if (!mountedComponent) {
  return <div />;
}

```

Лістинг HomePage.jsx:

```

import React, { useState } from "react";
import styled from "styled-components";
import MenuContainer from "../../components/menu/public/MenuContainer";

```

```
// Styled Components для стилізації елементів
```

```

const StyledHomePage = styled.div`
  background-color: #7e57c2;
  color: #fff; /* Білий колір тексту */
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
`;

```

```

const Title1 = styled.h1`
  font-size: 4rem;
  margin-bottom: 0;
`;

```

```

const Title2 = styled.h1`
  font-size: 7rem;
  margin-left: 0;
`;

```

```

const ImagesContainer = styled.div`
  display: flex;
  justify-content: space-between;
  align-items: center;
  width: 80%;
  margin-top: 120px;
  margin-bottom: 120px;
  margin-right: 250px;
`;

```

```

const ImagesContainer2 = styled.div`
  //display: flex;
  justify-content: space-between;
  align-items: center;
  width: 80%;
`;

```

```

const ImageWrapper = styled.div`
  border: 10px solid rgba(255, 255, 255, 0.5);
  border-radius: 20px;

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		71

```

        overflow: hidden;
    `;

const Image = styled.img`
    width: 600px;
    height: auto;
`;

const HomePage = () => {

    return (
        <StyledHomePage>
            <ImagesContainer>
                <>
                    <ImageWrapper>
                        <Image
                            src="https://img.freepik.com/free-vector/online-reservation-isometric-composition-waiter-during-customer-service-mobile-device-screen-white-purple_1284-31769.jpg?w=826&t=st=1718830015~exp=1718830615~hmac=b9e8b824808a9c5aaedb24bbc17b8edcc67018ea26c941357519bb8702fa9864"
                            alt="Image 1"
                        />
                    </ImageWrapper>
                    <div>
                        <Title1>Welcome to</Title1>
                        <Title2>EATIFY</Title2>
                    </div>
                </>
            </ImagesContainer>
            <ImagesContainer2>
                <MenuContainer />
            </ImagesContainer2>
        </StyledHomePage>
    );
};

export default HomePage;

```

Лістинг User.php:

```

<?php

namespace App\Entity;

use ApiPlatform\Core\Annotation\ApiFilter;
use ApiPlatform\Core\Annotation\ApiResource;
use ApiPlatform\Core\Bridge\Doctrine\Orm\Filter\RangeFilter;
use ApiPlatform\Core\Bridge\Doctrine\Orm\Filter\SearchFilter;
use App\Action\user\ChangePasswordAction;
use App\Action\user\ChangePasswordUserAction;
use App\Action\user\ConfirmEmailAction;
use App\Action\user\ForgotPasswordAction;
use App\Entity\Order\Order;
use App\Entity\Reservation\Reservation;
use App\Entity\Settings\UserSettings;
use App\Entity\Table\Table;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;
use Doctrine\ORM\Mapping as ORM;
use Ramsey\Uuid\Uuid;
use Ramsey\Uuid\UuidInterface;

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		72

```

use Symfony\Bridge\Doctrine\Validator\Constraints\UniqueEntity;
use Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface;
use Symfony\Component\Security\Core\User\UserInterface;
use Symfony\Component\Serializer\Annotation\Groups;
use Symfony\Component\Validator\Constraints as Assert;

```

```

#[ApiResponse(
    collectionOperations: [
        'get' => [
            'security' => "is_granted('ROLE_ADMIN')",
            'method' => "GET",
            'normalization_context' => ['groups' => ['get:collection:user']]
        ],
        'post' => [
            'method' => "POST",
            'denormalization_context' => ['groups' => ['post:collection:user']],
            'normalization_context' => ['groups' => ['get:item:user']]
        ],
        'forgotPassword' => [
            'method' => "POST",
            'path' => "/users/forgot-password",
            'denormalization_context' => ['groups' => ['post:collection:user']],
            'normalization_context' => ['groups' => ['get:collection:user']],
            'controller' => ForgotPasswordAction::class
        ],
        'changePassword' => [
            'method' => "POST",
            'path' => "/users/change-password",
            'denormalization_context' => ['groups' => ['post:collection:user']],
            'normalization_context' => ['groups' => ['get:collection:user']],
            'controller' => ChangePasswordAction::class,
        ],
        'confirmEmail' => [
            'method' => "GET",
            'path' => "/users/confirm-email/{token}",
            'controller' => ConfirmEmailAction::class,
            'normalization_context' => ["groups" => ["get:collection:confirm-email-user"]],
        ]
    ],
    itemOperations: [
        'get' => [
            'method' => "GET",
            'normalization_context' => ['groups' => ['get:item:user']]
        ],
        'put' => [
            'method' => "PUT",
            'security' => "is_granted('ROLE_ADMIN')",
            'denormalization_context' => ['groups' => ['put:item:user']],
            'normalization_context' => ['groups' => ['get:item:user']]
        ],
        'changePasswordUser' => [
            'method' => "PUT",
            'path' => "/users/change-password/{id}",
            'security' => "object == user",
            'controller' => ChangePasswordUserAction::class,
            'denormalization_context' => ['groups' => ['put:item:user']],
            'normalization_context' => ['groups' => ['get:item:user']]
        ]
    ],
    attributes: [
        'order' => ['createDate' => 'DESC']
    ]
)]

```

```

    ],
    mercure: [
        "normalization_context" => ["groups" => ["get:item:user"]]
    ]
}]
#[ApiFilter(SearchFilter::class, properties: [
    'email' => 'partial',
    'firstname' => 'partial',
    'lastname' => 'partial',
    'roles' => 'partial',
    'isEnabled' => 'exact'
])]
#[ApiFilter(RangeFilter::class, properties: ['createDate'])]
/ * @ORM\Entity(repositoryClass="App\Repository\UserRepository")
* @UniqueEntity(
*     fields = {"email"},
*     errorPath = "email",
*     message = "This user already exist, try login"
* )
* /
class User implements UserInterface, PasswordAuthenticatedUserInterface
{
    public const EMPLOYER = 'ROLE_EMPLOYER';
    public const CLIENT = 'ROLE_CLIENT';
    public const ADMIN = 'ROLE_ADMIN';

    /
    * @var UuidInterface
    * @ORM\Id()
    * @ORM\Column(type="uuid")
    * @Groups({
    *     "get:collection:user",
    *     "get:item:user",
    *     "get:collection:reservation",
    *     "get:item:reservation",
    *     "get:collection:shift",
    *     "get:item:shift"
    * })
    * /
    private UuidInterface $id;

    /
    * @Assert\NotBlank()
    * @Assert\Email()
    * @ORM\Column(type="string", length=255, unique=true)
    * @Groups({
    *     "get:collection:user",
    *     "get:item:user",
    *     "post:collection:user",
    *     "get:collection:reservation",
    *     "get:item:reservation",
    *     "get:collection:shift",
    *     "get:item:shift",
    *     "put:item:user",
    *     "get:collection:order",
    *     "get:item:order"
    * })
    * @Assert\Regex(
    *     pattern="/^[_a-z0-9-]+\.[_a-z0-9-]*@[a-z]+\.[a-z]+\.[a-z]{2,}$/",
    *     match=true,
    *     message="Пошта має невірні значення"
    * )

```

```

* )
*/
private string $email;

/
* @ORM\Column(type="string", length=255)
* @Groups({
*     "get:collection:user",
*     "get:item:user",
*     "post:collection:user",
*     "get:collection:reservation",
*     "get:item:reservation",
*     "get:collection:shift",
*     "get:item:shift",
*     "put:item:user"
* })
* @Assert\Regex(
*     pattern="/^[\\'\\\"-a-zA-Zа-ягіїєёышщъхротьюА-ЯГІІЄЁЫ ]+$/",
*     match=true,
*     message="Ім'я має невірні значення"
* )
*/
private string $firstname;

/
* @ORM\Column(type="string", length=255)
* @Groups({
*     "get:collection:user",
*     "get:item:user",
*     "post:collection:user",
*     "get:collection:reservation",
*     "get:item:reservation",
*     "get:collection:shift",
*     "get:item:shift",
*     "put:item:user"
* })
* @Assert\Regex(
*     pattern="/^[\\'\\\"-a-zA-Zа-ягіїєёышщъхротьюА-ЯГІІЄЁЫ ]+$/",
*     match=true,
*     message="Прізвище має невірні значення"
* )
*/
private string $lastname;

/
* @ORM\Column(type="string", length=255)
*/
private string $password;

/
* @var string|null
*/
private ?string $generatedPassword;

/
* @var string|null
* @ORM\Column(type="string", length=100, nullable=true)
* @Groups({"put:collection:user"})
*/
private ?string $forgotPasswordToken;

```

```

/
* @var string|null
* @ORM\Column(type="string", length=100, nullable=true)
* @Groups({"get:collection:confirm-email-user"})
*/
private ?string $emailActivationToken;

/
* @ORM\Column(type="simple_array", nullable=true)
* @Groups({
*     "get:collection:user",
*     "get:item:user",
*     "post:collection:user",
*     "put:item:user"
* })
*/
private array $roles;

/
* @var bool|null
* @ORM\Column(type="boolean", options={"default" : 0})
* @Groups({
*     "put:item:user",
*     "get:collection:user",
*     "get:item:user"
* })
*/
private ?bool $isEnabled;

/
* @var bool|null
* @ORM\Column(type="boolean", options={"default" : 0})
* @Groups({
*     "put:item:user",
*     "get:item:user",
*     "get:collection:user"
* })
*/
private ?bool $isBanned; /
* @var bool|null
* @ORM\Column(type="boolean", options={"default" : 0})
* @Groups({
*     "put:item:user",
*     "get:item:user",
*     "get:collection:user"
* })
*/
private ?bool $isDeleted;

/
* @ORM\OneToMany(targetEntity="App\Entity\Settings\UserSettings", mappedBy="user")
*/
private Collection $settings;

/
* @ORM\OneToMany(targetEntity=Shift::class, mappedBy="employee", orphanRemoval=true)
*/
private Collection $shifts;

/
* @ORM\OneToMany(targetEntity=Reservation::class, mappedBy="client")

```

```

*/
private Collection $reservations;

/
* @ORM\OneToMany(targetEntity=Order::class, mappedBy="client")
*/
private Collection $orders;

/
* @ORM\ManyToMany(targetEntity=Order::class, mappedBy="employees")
* @ORM\JoinTable(name="order_employee")
*/
private Collection $employeeOrders;

/
* @ORM\ManyToMany(targetEntity=Table::class, mappedBy="employees")
*/
private Collection $tables;

/
* @Assert\NotBlank()
* @ORM\Column(type="integer")
* @Groups({
*     "get:collection:user",
*     "get:item:user"
* })
*/
private int $createDate;

/
* @ORM\Column(type="decimal", precision=19, scale=8, nullable=true)
* @Groups({
*     "get:collection:user",
*     "get:item:user",
*     "put:item:user",
*     "post:collection:user"
* })
*/
private ?string $salary;

/
* @ORM\Column(type="integer", nullable=true)
* @Groups({
*     "get:collection:user",
*     "get:item:user",
*     "put:item:user",
*     "post:collection:user"
* })
*/
private ?int $hireDate;

/
* User constructor.
*/
public function __construct()
{
    $this->createDate = time();
    $this->id = Uuid::uuid4();
    $this->settings = new ArrayCollection();
    $this->roles = [self::CLIENT];
    $this->isEnabled = false;

```

```

$this->isBanned = false;
$this->isDeleted = false;
$this->shifts = new ArrayCollection();
$this->reservations = new ArrayCollection();
$this->orders = new ArrayCollection();
$this->employeeOrders = new ArrayCollection();
$this->tables = new ArrayCollection();
}

/
* @return UuidInterface|null
*/
public function getId(): ?UuidInterface
{
    return $this->id;
}

/
* @return string
*/
public function getEmail(): string
{
    return $this->email;
}

/
* @param string $email
* @return $this
*/
public function setEmail(string $email): self
{
    $this->email = $email;

    return $this;
}

/
* @return string
*/
public function getFirstname(): string
{
    return $this->firstname;
}

/
* @param string $firstname
* @return $this
*/
public function setFirstname(string $firstname): self
{
    $this->firstname = $firstname;

    return $this;
}

/
* @return string
*/
public function getLastname(): string
{
    return $this->lastname;
}

```

```

}

/
* @param string $lastname
* @return $this
*/
public function setLastname(string $lastname): self
{
    $this->lastname = $lastname;

    return $this;
}

/
* @return string|null
*/
public function getPassword(): ?string
{
    return $this->password;
}

/
* @param string $password
* @return $this
*/
public function setPassword(string $password): self
{
    $this->password = $password;

    return $this;
} /
* @return string|null
*/
public function getGeneratedPassword(): ?string
{
    return $this->generatedPassword;
}

/
* @param string|null $generatedPassword
* @return $this
*/
public function setGeneratedPassword(?string $generatedPassword): self
{
    $this->generatedPassword = $generatedPassword;

    return $this;
}

/
* @return array
*/
public function getRoles(): array
{
    return array_unique($this->roles);
}

/
* @param string $role
* @return bool
*/

```

```

public function checkRole(string $role): bool
{
    return in_array($role, $this->roles);
}

/
* @param array $roles
* @return $this
*/
public function setRoles(array $roles): self
{
    $this->roles = $roles;

    return $this;
}

/
* @param array $roles
*/
public function addRoles(array $roles)
{
    $this->roles = array_unique(array_merge($this->roles, $roles));
}

/
* @param UserSettings $settings
* @return $this
*/
public function addSettings(UserSettings $settings): self
{
    if (!$this->settings->contains($settings)) {
        $this->settings[] = $settings;
        $settings->setUser($this);
    }

    return $this;
}

/
* @param UserSettings $settings
*/
public function removeSettings(UserSettings $settings): void
{
    if ($this->settings->contains($settings)) {
        $this->settings->removeElement($settings);
        if ($settings->getUser() === $this) {
            $settings->setUser(null);
        }
    }
}

/
* @return string|null
*/
public function getSalt(): ?string
{
    return null;
}

/
* @return string

```

```

*/
public function getUserIdentifier(): string
{
    return $this->email;
}

/
* @return string
*/
public function getUsername(): string
{
    return $this->getUserIdentifier();
}

/
* @return void
*/
public function eraseCredentials()
{
    // TODO: Implement eraseCredentials() method.
}

/
* @return string|null
*/
public function getForgotPasswordToken(): ?string
{
    return $this->forgotPasswordToken;
}

/
* @param string|null $forgotPasswordToken
* @return User
*/
public function setForgotPasswordToken(?string $forgotPasswordToken): self
{
    $this->forgotPasswordToken = $forgotPasswordToken;

    return $this;
}

/
* @return string|null
*/
public function getEmailActivationToken(): ?string
{
    return $this->emailActivationToken;
}

/
* @param string|null $emailActivationToken
* @return User
*/
public function setEmailActivationToken(?string $emailActivationToken): self
{
    $this->emailActivationToken = $emailActivationToken;

    return $this;
}

/

```

```

* @return bool|null
*/
public function getIsEnabled(): ?bool
{
    return $this->isEnabled;
}

/
* @param bool|null $isEnabled
* @return User
*/
public function setIsEnabled(?bool $isEnabled): self
{
    $this->isEnabled = $isEnabled;

    return $this;
}

/
* @return bool|null
*/
public function getIsBanned(): ?bool
{
    return $this->isBanned;
}

/
* @param bool|null $isBanned
* @return User
*/
public function setIsBanned(?bool $isBanned): self
{
    $this->isBanned = $isBanned;

    return $this;
}

/
* @return bool|null
*/
public function getIsDeleted(): ?bool
{
    return $this->isDeleted;
}

/
* @param bool|null $isDeleted
* @return User
*/
public function setIsDeleted(?bool $isDeleted): self
{
    $this->isDeleted = $isDeleted;

    return $this;
} /
* @return Collection<int, Shift>
*/
public function getShifts(): Collection
{
    return $this->shifts;
}

```

```

/
* @param Shift $shift
* @return $this
*/
public function addShift(Shift $shift): self
{
    if (!$this->shifts->contains($shift)) {
        $this->shifts[] = $shift;
        $shift->setEmployee($this);
    }

    return $this;
}

/
* @param Shift $shift
* @return $this
*/
public function removeShift(Shift $shift): self
{
    if ($this->shifts->removeElement($shift)) {
        // set the owning side to null (unless already changed)
        if ($shift->getEmployee() === $this) {
            $shift->setEmployee(null);
        }
    }

    return $this;
}

/
* @return Collection<int, Reservation>
*/
public function getReservations(): Collection
{
    return $this->reservations;
}

/
* @param Reservation $reservation
* @return $this
*/
public function addReservation(Reservation $reservation): self
{
    if (!$this->reservations->contains($reservation)) {
        $this->reservations[] = $reservation;
        $reservation->setClient($this);
    }

    return $this;
}

/
* @param Reservation $reservation
* @return $this
*/
public function removeReservation(Reservation $reservation): self
{
    if ($this->reservations->removeElement($reservation)) {
        if ($reservation->getClient() === $this) {

```

```

        $reservation->setClient(null);
    }
}

return $this;
}

/
* @return Collection<int, Order>
*/
public function getOrders(): Collection
{
    return $this->orders;
}

/
* @param Order $order
* @return $this
*/
public function addOrder(Order $order): self
{
    if (!$this->orders->contains($order)) {
        $this->orders[] = $order;
        $order->setClient($this);
    }

    return $this;
}

/
* @param Order $order
* @return $this
*/
public function removeOrder(Order $order): self
{
    if ($this->orders->removeElement($order)) {
        // set the owning side to null (unless already changed)
        if ($order->getClient() === $this) {
            $order->setClient(null);
        }
    }

    return $this;
}

/
* @return Collection<int, Order>
*/
public function getEmployeeOrders(): Collection
{
    return $this->employeeOrders;
}

/
* @param Order $employeeOrder
* @return $this
*/
public function addEmployeeOrder(Order $employeeOrder): self
{
    if (!$this->employeeOrders->contains($employeeOrder)) {
        $this->employeeOrders[] = $employeeOrder;
    }
}

```

```

        $employeeOrder->addEmployee($this);
    }

    return $this;
}

/
* @param Order $employeeOrder
* @return $this
*/
public function removeEmployeeOrder(Order $employeeOrder): self
{
    if ($this->employeeOrders->removeElement($employeeOrder)) {
        $employeeOrder->removeEmployee($this);
    }

    return $this;
}

/
* @return Collection<int, Table>
*/
public function getTables(): Collection
{
    return $this->tables;
}

/
* @param Table $table
* @return $this
*/
public function addTable(Table $table): self
{
    if (!$this->tables->contains($table)) {
        $this->tables[] = $table;
        $table->addEmployee($this);
    }

    return $this;
}

/
* @param Table $table
* @return $this
*/
public function removeTable(Table $table): self
{
    if ($this->tables->removeElement($table)) {
        $table->removeEmployee($this);
    }

    return $this;
}

/
* @return int
*/
public function getCreateDate(): int
{
    return $this->createDate;
} /

```

```

* @return string|null
*/
public function getSalary(): ?string
{
    return $this->salary;
}

/
* @param string|null $salary
* @return $this
*/
public function setSalary(?string $salary): self
{
    $this->salary = $salary;

    return $this;
}

/
* @return int|null
*/
public function getHireDate(): ?int
{
    return $this->hireDate;
}

/
* @param int|null $hireDate
* @return $this
*/
public function setHireDate(?int $hireDate): self
{
    $this->hireDate = $hireDate;

    return $this;
}
}

```