

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ

к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "бакалавр"

на тему: «Розробка інформаційної системи управління рестораном»

Виконав: студент групи

_____ *Козлов О.Ю.*
(підпис)

Керівник роботи:

(посада, науковий ступінь вчене звання)

_____ *Павленко А.Ю.*
(підпис)

м. Миколаїв – 2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 126 "Інформаційні системи та технології"
Освітня програма "Інформаційні системи та технології"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

к.т.н, доц. НУК Книрик Н.Р.
" ____ " _____ 2024 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ на здобуття ступеня вищої освіти "бакалавр"

Студенту Козлову Олександрю Юрійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка інформаційної системи управління рестораном»

Керівник роботи Павленко Альона Юріївна

Затверджені наказом ректора № 243-уч від "18" березня 2024 року.

2. Термін подання роботи: 20 червня 2024 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем,

4. Перелік питань, що належать до розробки (найменування розділів) дослідження та аналіз предметної галузі, розробка проекту інформаційної підсистеми для менеджменту ресторану, реалізація проекту інформаційної системи для управління рестораном, охорона праці.

5. Перелік презентаційних матеріалів вступ, мета та актуальність роботи, схема запропонованої концепції, діаграма варіантів використання, концептуальна модель, фізична модель, діаграма компонентів, діаграма класів, діаграма переходу станів, діаграма розгортання системи, результати розробки, висновок.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Павленко А. Ю. старший викладач	05.02.2024	22.02.2024
2	Павленко А. Ю. старший викладач	22.02.2024	21.03.2024
3	Павленко А. Ю. старший викладач	21.03.2024	25.04.2024
4	Павленко А. Ю. старший викладач	25.04.2024	07.06.2024

7. Дата видачі завдання 05 лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі	05.02.2024	
2	Постановка задачі	29.02.2024	
3	Розробка концепції	21.03.2024	
4	Розробка проекту веб-сайту	11.04.2024	
5	Реалізація проекту	23.05.2024	
6	Розробка питань з охорони праці	05.06.2024	
7	Розробка графічних матеріалів	12.06.2024	
8	Подання роботи на перевірку на плагіат	10.06.2024	
9	Подання роботи рецензенту	15.06.2024	
10	Подання роботи на захист	20.06.2024	

Студент

(підпис)

Козлов О.Ю.

(ПІБ)

Керівник роботи

(підпис)

Павленко А.Ю.

(ПІБ)

АНОТАЦІЯ

Обсяг пояснювальної записки складає сторінок. Робота складається з 71 сторінки, 24 ілюстрацій, 13 таблиць, 3 додатків та 21 джерела у списку літератури.

Об'єктом дослідження є персонал ресторану. Предмет дослідження - це робочі процеси, пов'язані з обслуговування клієнтів, управлінням рестораном та внутрішньою комунікацією. Метою проекту покращення та прискорення процесів комунікації за допомогою автоматизації процесів роботи в команді.

У рамках дослідження використовувалися методи аналізу вимог користувачів, проектування системи та інтерфейсу, розробки програмного забезпечення та тестування. Результати дослідження включають функціональну веб-платформу, що відповідає потребам користувачів.

Основні технологічні характеристики проекту включають використання мови програмування Java, фреймворку Spring, а також бази даних MariaDB.

Отримані результати можуть бути застосовані в сфері ресторанного бізнесу для покращення обслуговування клієнтів та збільшенню продуктивності персоналу. Висновки та прогнози щодо розвитку проекту підкреслюють важливість та перспективність подальшої розробки та вдосконалення інформаційної системи.

Ключові слова: ресторан, веб-платформа, меню, замовлення, розробка, технології, база даних.

ANNOTATION

The length of the explanatory note is pages. The paper consists of 71 pages, 24 illustrations, 13 tables, 3 appendices and 21 references.

The object of research is restaurant staff. The subject of the study is work processes related to customer service, restaurant management and internal communication. The aim of the project is to improve and accelerate communication processes by automating teamwork processes.

The research involved user requirements analysis, system and interface design, software development and testing. The research results include a functional web platform that meets the needs of users.

The main technological characteristics of the project include the use of the Java programming language, the Spring framework, and the MariaDB database.

The results obtained can be applied in the restaurant business to improve customer service and increase staff productivity. The conclusions and forecasts for the development of the project emphasise the importance and prospects for further development and improvement of the information system.

Keywords: restaurant, web platform, menu, order, development, technology, database.

ЗМІСТ

ВСТУП.....	8
РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ	9
1.1. Дослідження та аналіз предметної галузі	9
1.2. Аналіз існуючих рішень	11
1.3. Визначення проблем	15
1.4 Постановка задачі.....	16
1.5. Розробка концепцій рішення проблем	17
РОЗДІЛ 2. РОЗРОБКА ПРОЕКТУ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ ДЛЯ МЕНЕДЖМЕНТУ РЕСТОРАНУ	20
2.1 Загальносистемні рішення.....	20
2.1.1. Загальний опис системи	20
2.1.2. Розробка діаграми варіантів використання	21
2.2 Рішення з інформаційного забезпечення.....	23
2.3 Рішення з технічного забезпечення.....	28
2.4 Рішення з програмного забезпечення	29
2.4.1 Розробка схеми структури інформаційної системи.....	29
2.4.2 Розробка діаграми компонентів.....	32
2.4.3 Розробка діаграми переходів станів	33
2.4.4 Розробка діаграми розгортання підсистеми	35
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЕКТУ ІНФОРМАЦІЯ СИСТЕМИ ДЛЯ УПРАВЛІННЯ РЕСТОРАНОМ.....	36
3.1 Аргументація вибору засобів розробки	36
3.2 Розробка структури програмного забезпечення	37
3.3 Результат розробки програмного забезпечення.....	40
РОЗДІЛ 4 ОХОРОНА ПРАЦІ	43
4.1 Загальні питання охорони праці	43
4.2 Структура управління питаннями охорони праці на підприємстві	43
4.3 Шкідливі та небезпечні фактори, що супроводжують працю програміста ..	45
4.4 Ергономічні вимоги до робочого місця при роботі за комп'ютером	46
4.5 Нормативні вимоги до робочої площі на одного працюючого за комп'ютером	47

ВИСНОВОК.....	49
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	50
ДОДАТОК А – Технічне завдання.....	52
ДОДАТОК Б – ПРОГРАМНИЙ КОД.....	56
ДОДАТОК В – ІНСТРУКЦІЯ КОРИСТУВАЧА.....	66

ВСТУП

У світі, де технології проникають у всі аспекти нашого життя, програмне забезпечення стає необхідним інгредієнтом багатьох сфер, а ресторанний бізнес тут не є винятком. Його розвиток потребує вдосконалення управління та оптимізації робочих процесів, і розробка системи управління для ресторанів стає вельми актуальною.

Мета дипломної роботи полягає в створенні цифрової системи управління рестораном, яка надасть зручні і ефективні інструменти для замовлення страв, керування столами та аналізу статистики замовлень. Система спростить процес обслуговування клієнтів, підвищить точність розрахунків та підтримає управління рестораном.

Головним завданням додатку буде – автоматизація всіх внутрішніх процесів, адже достатньо велика структура персоналу створює багато внутрішніх процесів взаємодії в команді.

У подальших розділах кваліфікаційної роботи буде розглянуто процеси аналізу вимог, проектування, розробки та тестування системи управління рестораном. Визначено основні потреби та вимоги, розроблено структуру бази даних та логіку роботи системи, і висвітлено методи тестування, які допоможуть забезпечити функціональність та надійність системи.

Висновки роботи містять оцінку досягнутих результатів, аналіз переваг та недоліків системи управління рестораном, а також пропозиції щодо подальшого розвитку та вдосконалення. Також наведено список джерел, що були використані під час роботи, що підтверджує достовірність та наукову обґрунтованість проекту.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ТА АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1. Дослідження та аналіз предметної галузі

Чіткий поділ відповідальності між робітниками у ресторанному бізнесі є невід'ємною складовою успішної діяльності закладу. Цей аспект впливає на всі аспекти роботи, від організації робочих процесів до підвищення продуктивності та якості обслуговування.

Основна структура персоналу зображена на рис. 1.1

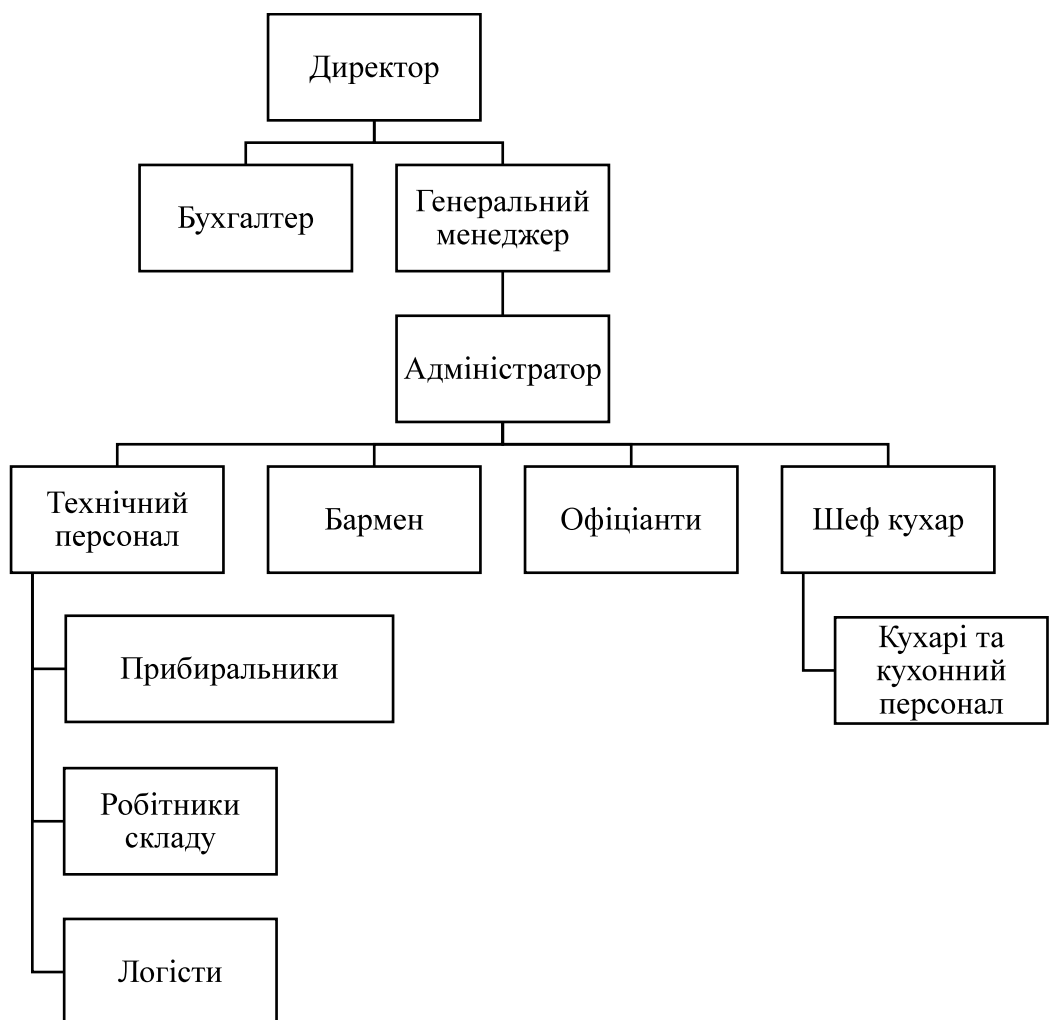


Рисунок 1.1 - Кадрова структура підприємства

Кожен працівник, знаючи свої чітко визначені обов'язки, може ефективно зосередитися на їх виконанні, що сприяє уникненню конфліктів та

покращує комунікацію всередині команди. Це також допомагає зменшити ризик помилок та підвищує загальний рівень продуктивності. Відповідальний підхід до розподілу завдань створює основу для успішної командної роботи та сприяє досягненню спільних цілей ресторану.

Основні задачі кожного робітника у табл.1.1.

Таблиця 1.1 – Основні задачі для співробітників

Посада	Основні щоденні задачі
Генеральний менеджер	- Планування та керівництво роботою всього ресторану
	- Вирішення поточних питань і проблем
	- Контроль за дотриманням стандартів і політики закладу
Адміністратор	- Організація роботи персоналу та розподіл обов'язків
	- Контроль за бронюванням столиків та обробкою замовлень
	- Вирішення будь-яких непередбачених ситуацій
Шеф-кухар	- Розробка щоденного меню з урахуванням сезонності
	- Перевірка якості і свіжості продуктів
	- Організація та контроль за роботою кухонного персоналу
Бармен	- Підготовка спеціальних напоїв та коктейлів
	- Збереження порядку та чистоти у барній зоні
	- Обслуговування гостей, консультація щодо напоїв
Офіціанти	- Прийом та обслуговування гостей згідно з встановленими стандартами
	- Підготовка залу до роботи та забезпечення порядку
Кухарі та кухонний персонал	- Підготовка страв згідно зі стандартами та замовленнями
	- Дотримання стандартів гігієни та безпеки на кухні
Прибиральники	- Прибирання кухні, залу та інших приміщень
	- Видалення сміття та забезпечення чистоти
Технічний персонал	- Технічне обслуговування та ремонт обладнання
	- Перевірка роботи систем опалення, кондиціонування, водопостачання
	- Реагування на будь-які аварійні ситуації

1.2. Аналіз існуючих рішень

Дослідження проблем схожих додатків у конкурентів може допомогти у вдосконаленні власного продукту.

Poster — система закладового обліку, яка надає можливість автоматизувати керування закладом [1].

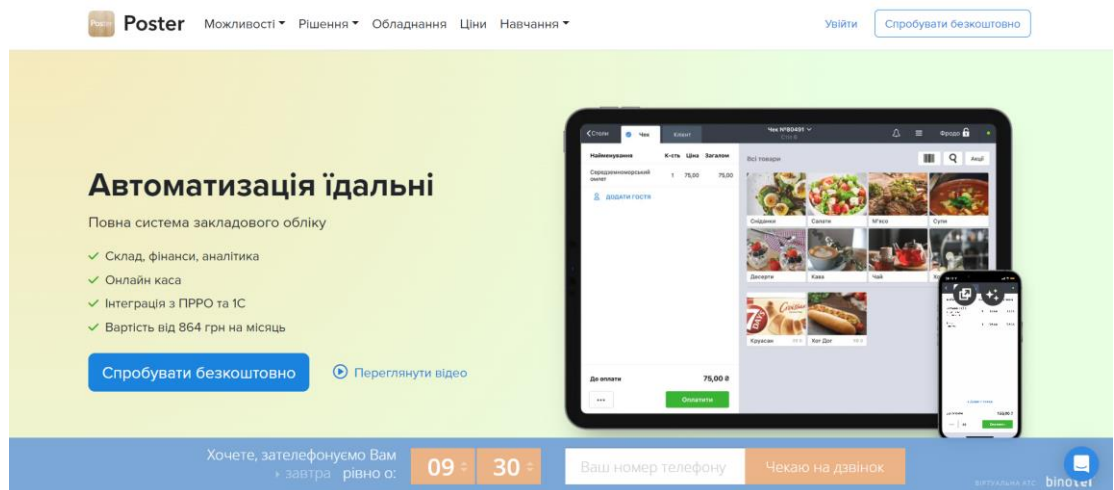


Рисунок 1.2 – Poster [1]

Переваги Poster:

- Інтеграція з ПРРО
- Програма лояльності клієнта
- Багато методів оплати замовлення
- Можливість контролювати декілька закладів з одного облікового запису
- Автоматизація доставки їжі

Недоліки Poster:

- Велика затримка відповідей від сервера
- Проблема з інтуїтивністю інтерфейсу користувача
- Відсутність можливості генерації звітів в Excel форматі

Також серед українських систем для закладів можна виділити Kavarr, що пропонує швидке комплексне рішення за невеликі гроші для кав'ярень.

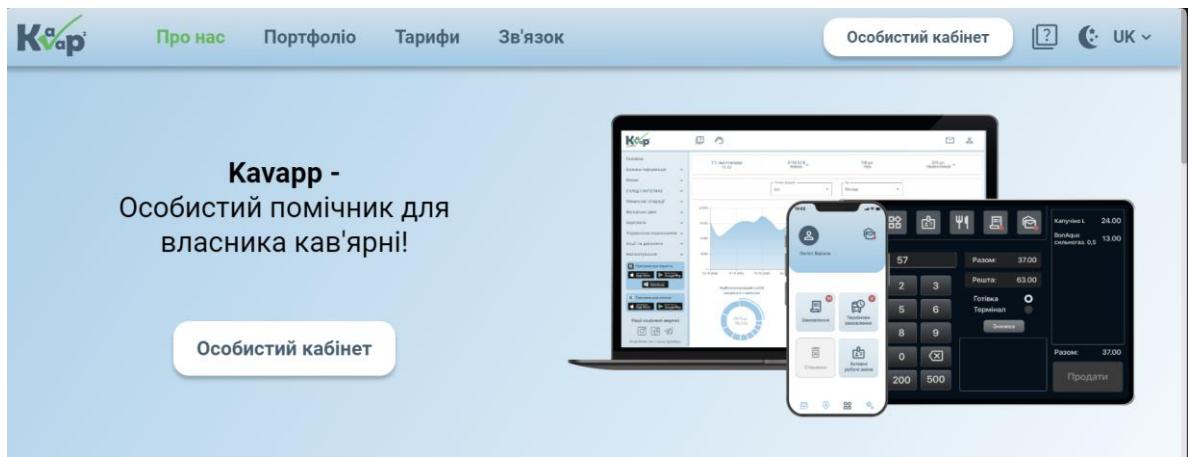


Рисунок 1.3 – KavaApp [2]

Серед своїх переваг виробник виділяє наступні:

- Низька ціна, в порівнянні з іншими сервісами. Можливість обрати оптимальний тариф для Вашого типу закладу
- Створення технологічних карток в необмеженій кількості
- Месенджер - внутрішнє листування між бариста, логістами та адміністраторами в робочий час, не відволікаючись від робочого процесу
- Висока сумісність програми з різними пристроями - система Kavapp підтримує пристрої на ОС Android, iOS та Windows
- Робота з ПРРО - відкривайте фіскальні зміни та фіскалізуєте продані чеки в податковій прямо з робочого пристрою, відповідно до чинного законодавства
- Розрахунок зарплати, нарахування штрафів, премій, авансів
- Контроль роботи бариста, логіста та рух товару на складі

Також серед переваг можемо виділити доступну ціну (від 99 грн/місяць на 1 торгову точку) та комплексність рішення.

Споживачі виявляють такі недоліки при користуванні як незручний інтерфейс та велику кількість системних збоїв, що негативно впливають на роботу закладів. Також це рішення не підходить для більших систем, наприклад, кафе, їдальні, ресторанів.

Найвідомішим в Україні є сервіс R_keeper, що застосовується у більш ніж 65 тисячах ресторанах. Це досить складна система, яку доцільно розібрати детальніше, щоб перейняти досвід великих гравців на ринку.

R_keeper пропонує повну автоматизацію бізнеса, детальніше на рис. 1.4



Рисунок 1.4 – R-keeper [3]

Система має 6 основних модулів: каса офіціанта, станція менеджера, мобільний офіціант, дисплей для кухні, CRM-система та Dashboard. Усі вони формують єдину екосистему ресторану.

Каса та мобільний офіціант – це прилад та додаток, що замінюють звичні блокноти з ручкою для офіціантів. Там у інтуїтивно зрозумілому меню робітник може швидко оформити замовлення прямо у систему зі свого телефону або спеціального комп'ютера, який можна встановити, наприклад, на барі



Рисунок 1.5 – Додаток офіціанта та Каса офіціанта [3]

Станція менеджера представляє собою більш складну управлінську систему керування закладом. Завдяки ній можна зручно налаштувати касову станцію для роботи з потоком гостей, керувати доступом персоналу до касових операцій, своєчасно формувати звіти з роботи ресторану.

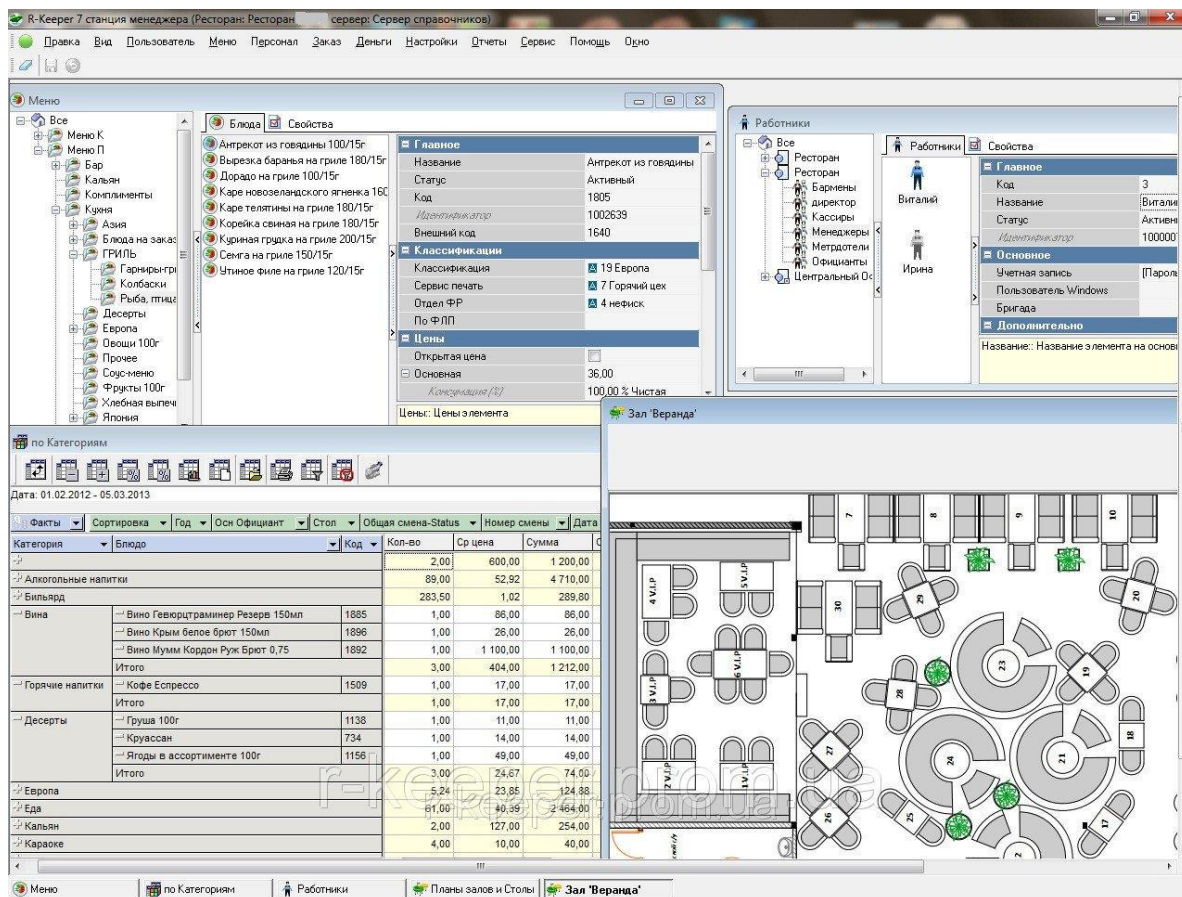


Рисунок 1.6 – Система менеджера [3]

CRM та Dashboard дають можливість маркетологам та керівництву досліджувати інформацію про фінансові показники, статистику, приймати управлінські рішення щодо розширення ресторану, зміни меню, цін, тощо.

Також важливим елементом є додаток для кухні, що передає на неї інформацію про замовлення, а також передає інформацію про готовність офіціантам.

Разом ці модулі утворюють велику еко-систему, що покриває усі аспекти керування рестораном. Великою проблемою є запровадження системи у закладах, адже її обслуговування та встановлення потребує значних капіталовкладень.

Дослідження та усунення типових проблем у продукті може підвищити конкурентоздатність додатку у порівнянні з аналогами у його цінovій категорії. Варто дослідити досвід найкращих, наприклад, r-keeper та перенести його у бюджетну та доступнішу версію для малого та середнього бізнесу України.

1.3. Визначення проблем

Нижче декілька типових проблем, які можуть виникати у подібних додатках та як їх можна усунути:

1. Погана швидкодія: Багато додатків мають проблеми зі швидкістю завантаження сторінок або відгуків на дії користувачів. Розробка оптимізованих алгоритмів та використання кешування може допомогти вирішити цю проблему.

2. Неінтуїтивний інтерфейс користувача: Інтерфейси, які складно розуміти або використовувати, можуть відлякати користувачів та знижувати їхню лояльність. Проведення тестування користувачів та використання найкращих практик UX/UI дизайну допоможе створити більш інтуїтивний і привабливий інтерфейс.

3. Недостатня безпека даних: Проблеми з безпекою даних, такі як витоки особистої інформації або вразливості в системі, можуть нашкодити репутації додатку. Регулярне тестування на наявність вразливостей та використання найкращих практик забезпечення безпеки даних допоможе захистити інформацію користувачів.

4. Обмежені функціональні можливості: Якщо додаток не надає достатньо функціональності для зручного та повного задоволення потреб користувачів, це може призвести до втрати клієнтів. Проведення аналізу

потреб користувачів та постійне вдосконалення функціоналу допоможе створити конкурентоздатний продукт.

5. Поганий сервіс підтримки користувачів: Недоступність або неефективність служби підтримки може викликати незадоволення серед користувачів. Забезпечення швидкої та якісної підтримки користувачів через різні канали зв'язку може вирішити цю проблему.

Незважаючи на швидкий розвиток технологій у всіх сферах життя, багато ресторанів залишаються вірними застарілим методам управління, таким як використання блокнотів, журналів або таблиць Excel для ведення внутрішньої політики та контролю за операціями.

Отже, перехід до сучасної цифрової системи управління для ресторанів малого бізнесу є важливим. Нова система має забезпечити зручність, ефективність та точність управління, що допоможе підвищити якість обслуговування, збільшити прибуток та задовольнити потреби як співробітників, так і гостей ресторанів.

1.4 Постановка задачі

На основі інформації аналізу предметної області, перевагах та недоліках аналогів і їх типових проблем сформовано такі вимоги до функціоналу цифрового додатку для автоматизації процесів замовлення та обслуговування відповідно користувача:

1. Адміністратор

1.1. Авторизація та аутентифікація.

1.2. Реєстрація нових користувачів.

1.3. Перегляд, редагування, створення та видалення категорій.

1.4. Перегляд, редагування, створення та видалення меню.

1.5. Перегляд, редагування, створення замовлень.

1.6. Перегляд, редагування, створення та видалення столиків.

- 1.7.Перегляд та редагування зарплат.
- 1.8.Перегляд та редагування статистики.
- 2. Кухар
 - 2.1.Авторизація та аутентифікація.
 - 2.2.Перегляд категорій.
 - 2.3.Перегляд меню.
 - 2.4.Перегляд та редагування замовлень.
- 3. Робітник залу
 - 3.1.Авторизація та аутентифікація.
 - 3.2.Перегляд категорій
 - 3.3.Перегляд меню
 - 3.4.Перегляд, створення та редагування замовлень.

Ці вимоги дозволять створити цифровий додаток, який відповідає потребам ресторану та допоможе оптимізувати його робочі процеси.

1.5. Розробка концепцій рішення проблем

Ресторани, які не мають цифрових рішень, які автоматизують внутрішні процеси, в більшості випадків використовують концепцію зображену на рисунку 1.7.

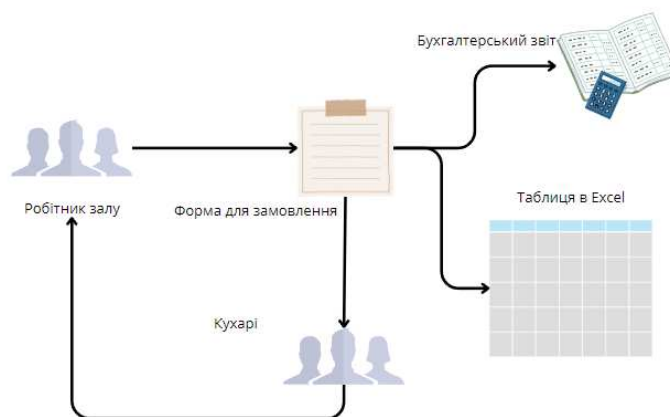


Рисунок 1.7 – Концепція внутрішніх процесів ресторану без автоматизації

Можна побачити, що зараз офіціанти вручну заносять дані про замовлення в форму, підраховують вручну чек, відносять цю форму кухарам та після цього вносять інформацію про замовлення в бухгалтерський звіт та таблицю в Excel. Звісно таблиця в Excel полегшить задачі аудиту та введення даних займає велику частину часу. Також офіціант має відносити форму замовлення на кухню, протягом певного часу має уточнювати чи готове його замовлення чи ні у поварів, що займає не мало часу. Також офіціанту потрібно запам'ятовувати зайняті столики, чи тримати це в якихось нотатках.

На основі поставлених задач та проблем концепції, що наявна зараз розроблено концепцію інформаційної системи яка представлена на рис 1.8.

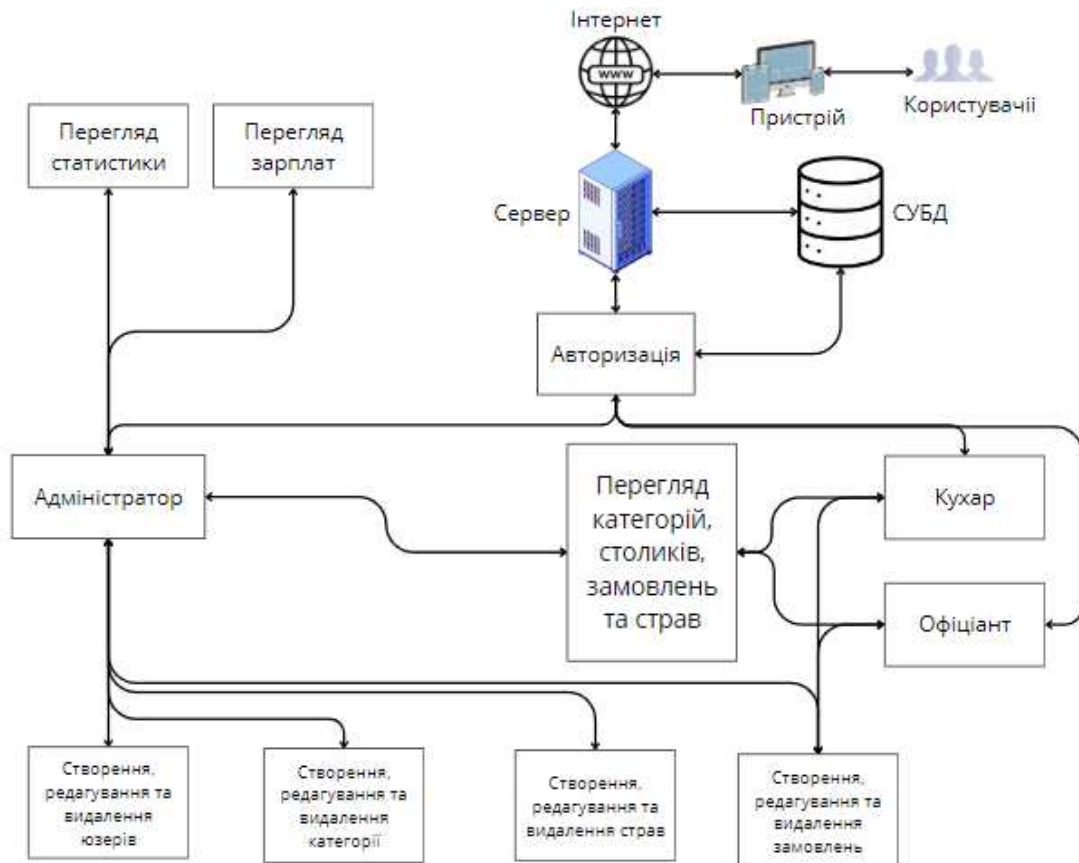


Рисунок 1.8 – Концепція інформаційної системи

В розробленій концепції від офіціантів потрібно лише створити замовлення, кухари автоматично отримають замовлення і візьмуть його в роботу. Статус столика буде змінено автоматично, в бухгалтерський облік

добавиться сума замовлення та його склад. Також автоматично буде змінена кількість інгредієнтів в складу, що є дуже зручним. Ці зміни допоможуть заощадити 90% часу офіціанта, а отже підвищити його продуктивність. Всі ці зміни позитивно впливатимуть на ведення внутрішніх справ в ресторані.

РОЗДІЛ 2. РОЗРОБКА ПРОЕКТУ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ ДЛЯ МЕНЕДЖМЕНТУ РЕСТОРАНУ

2.1 Загальносистемні рішення

2.1.1. Загальний опис системи

Система управління рестораном дозволяє персоналу ресторану автоматизувати більшу частину їх роботи та зберігати всі дані в одному місці.

Мета цього додатку вирішити довготривалі та багато затратні процеси всередині команди ресторану. Дана система дозволяє спростити комунікаційний процес між персоналом та даватиме можливість кожному користувачу приділяти увагу саме на якості обслуговування клієнтів.

Дані в застосунку мають бути розміщені в інтуїтивному та зручному для користувача вигляді. Користувачі залежно від своєї ролі можуть мати різні можливості такі як: перегляд, додавання, редагування та видалення інформації.

Сторінки, які мають бути наявними у додатку:

- Інформація про страви, можливість створення, редагування, видалення;
- Інформація про категорії, можливість створення, редагування, видалення;
- Інформація про столики;
- Інформація про страви, можливість створення, редагування, видалення;
- Інформація про замовлення, можливість створення, редагування, видалення;
- Інформація про статистику ресторану;

— Інформація про зарплати працівників.

2.1.2. Розробка діаграми варіантів використання

Діаграма варіантів використання фіксує функціональні можливості та вимоги системи за допомогою акторів і випадків використання. Випадки використання моделюють послуги, завдання та функції, які має виконувати система. Варіанти використання представляють функціональні можливості високого рівня та те, як користувач працюватиме з системою. Варіанти використання є ядром concepts of Unified Modeling мова моделювання.

Варіанти використання використовуються для представлення функціональних можливостей високого рівня та того, як користувач працюватиме з системою. Варіант використання представляє окрему функціональність системи, компонента, пакета або класу. Він позначається овальною фігурою з назвою варіанта використання, написаною всередині овальної форми.

Актор використовується в діаграмах варіантів використання. Актор - це суб'єкт, який взаємодіє з системою. Користувач - найкращий приклад актора. Актор — це об'єкт, який ініціює варіант використання поза межами його застосування. Це може бути будь-який елемент, який може ініціювати взаємодію з варіантом використання. Один актор може бути пов'язаний із кількома варіантами використання в системі [4].

Головними акторами у системі будуть: адміністратор, шеф-кухар, офіціант, бармен, кухар, бухгалтер. Кожен з них буде мати свій функціонал.

Короткий опис акторів представлений в табл. 2.1.

Таблиця 2.1. Короткий опис акторів

Роль	Актор	Опис
Адмін	Адміністратор	Має можливість перегляду, створення, редагування та видалення замовлень та страв. Також має можливість перегляду, створення, редагування та видалення категорій та юзерів. Має можливість перегляду столиків. Крім того, є можливість перегляду статистики та зарплат працівників.

Продовження таблиці 2.1

	Шеф-кухар	Має можливість перегляду, створення, редагування та видалення замовлень. Також має можливість перегляду, створення, редагування та видалення категорій. Має можливість перегляду страв їх редагування, створення та видалення. Має можливість перегляду столиків.
	Бухгалтер	Має можливість перегляду статистики ресторану. Також є можливість перегляду, редагування зарплати співробітників.
Офіціант	Офіціант	Має можливість перегляду, створення, редагування та видалення замовлень. Крім того, має права на перегляд: категорій, страв, столиків.
	Бармен	Має можливість перегляду, створення, редагування та видалення замовлень. Також має можливість на перегляд: категорій, страв, столиків.
Кухар	Кухар	Має можливість перегляду, створення, редагування та видалення замовлень. Крім того, має права на перегляд: категорій, страв, столиків.

В таблиці 2.1 можна побачити, що актори згруповані відповідно до ролі, яка буде для них створена в додатку та вони будуть мати доступи до функціоналу відповідно прав наданих цій ролі.

Проаналізувавши опис кожного з акторів було розроблено діаграму варіантів використання. Розглянемо діаграму варіантів використання зображену на рисунку 2.1.

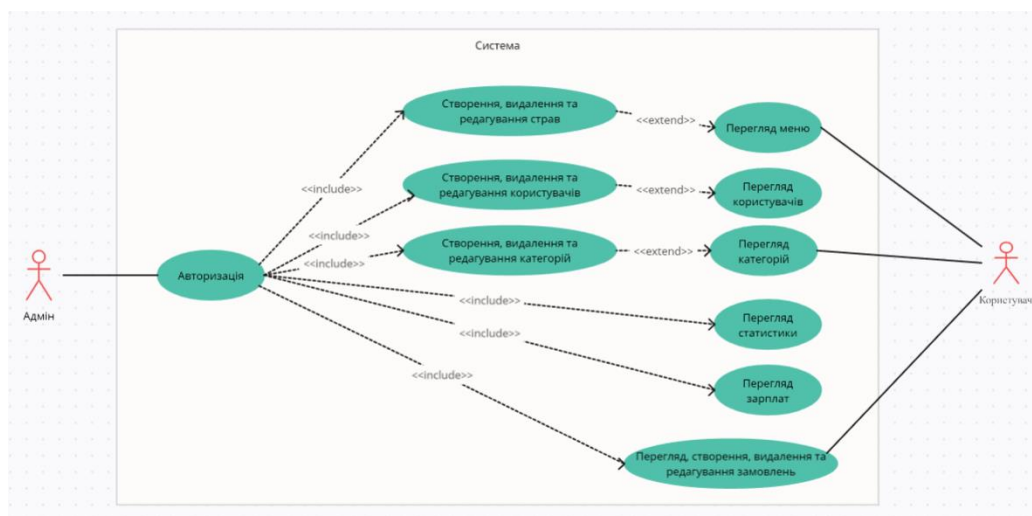


Рисунок 2.1 – Діаграма варіантів використання

2.2 Рішення з інформаційного забезпечення

Визначимо структуру та організацію бази даних, необхідної для ефективного функціонування платформи. У цьому підрозділі представлено концептуальну, логічну та фізичну моделі даних, а також проведено аналіз нормалізації бази даних на нормальні форми.

Концептуальна модель бази даних - це абстрактне представлення структури даних та їх взаємозв'язків у базі даних без деталей щодо фізичної реалізації. Вона описує сутності, атрибути і зв'язки між ними у вигляді концепційних сутностей, що допомагає розуміти логічну структуру даних та їх взаємодії в системі. Концептуальна модель бази даних забезпечує загальний огляд і визначає основні елементи, які будуть використовуватися в подальшій проектуванні та розробці бази даних. Вона є важливим інструментом для аналізу та спілкування між розробниками, замовниками та іншими зацікавленими сторонами в процесі розробки бази даних [5].

На рисунку 2.2 показано концептуальну модель даних. Виділено наступні сутності:

1. **Страви та Категорії (Menu_Items та Categories):** Страви поділяються на категорії для легшого пошуку та організації в меню ресторану. Кожна категорія має назву та унікальний ідентифікатор.

2. **Інгредієнти (Ingredients):** Інгредієнти, які використовуються для приготування страв, зазвичай мають назву, кількість, одиницю виміру та кількість калорій.

3. **Складові продукти (Ingredient_Items):** Ця сутність відображає зв'язок між стравами та інгредієнтами, вказуючи кількість кожного інгредієнта для конкретної страви.

4. **Працівники (Employees):** Профілі працівників ресторану зазначають ім'я, прізвище, роль у системі, інформацію для входу в систему (ім'я користувача та пароль).

5. **Столики (Tables):** Кожен стіл в ресторані має унікальний ідентифікатор та статус, який вказує, чи він вільний, зайнятий або заброньований.

6. **Замовлення (Orders):** Інформація про замовлення, включаючи ідентифікатор, дату та час створення, список елементів замовлення, статус замовлення та працівника, який обробляє замовлення.

7. **Статуси замовлення та столиків (Statuses та Table_Statuses):** Дозволяє відстежувати стан замовлення та статус столика (наприклад, "очікує підтвердження", "виконується", "завершено" для замовлень та "вільний", "зайнятий", "заброньований" для столиків).

8. **Ролі працівників (Roles):** Різні ролі працівників у системі, наприклад, адміністратор, офіціант, кухар.

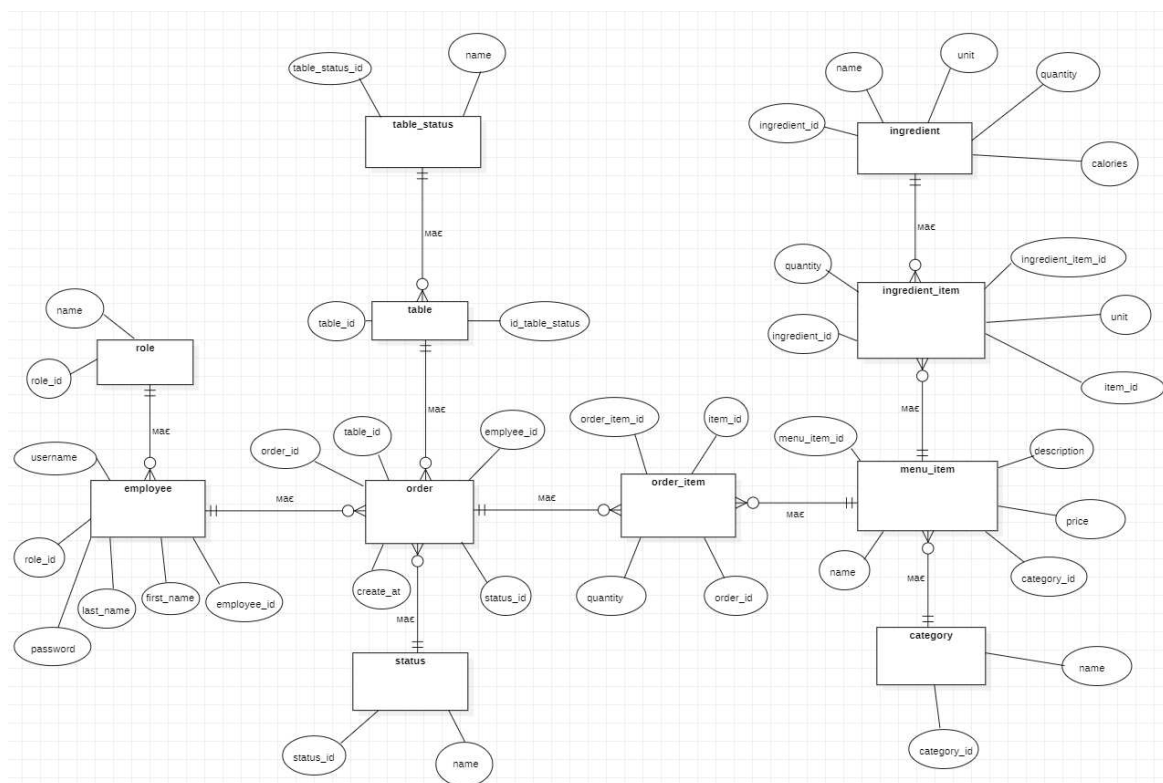


Рисунок 2.2 – Концептуальна модель бази даних

Логічна модель бази даних - це подальший етап після концептуальної моделі, де концепції та зв'язки концептуальної моделі перетворюються на конкретні структури даних, такі як таблиці, колонки та зв'язки між ними. Логічна модель визначає структуру даних в рамках обраної системи

управління базами даних (СУБД) і надає детальніше уявлення про те, як будуть зберігатися та оброблятися дані в межах бази даних. Вона описує таблиці, їх атрибути, типи даних та зв'язки між таблицями у вигляді схеми бази даних, що є основою для розробки фізичної реалізації бази даних. Логічна модель дозволяє уточнити дизайн бази даних та підготувати її для подальшої імплементації у вибраній СУБД [5].

На рисунку 2.2 представлена логічна модель нашої бази даних, де кожна сутність представлена в формі таблиці. Також представлені зв'язки між таблицями за допомогою зовнішніх ключів.

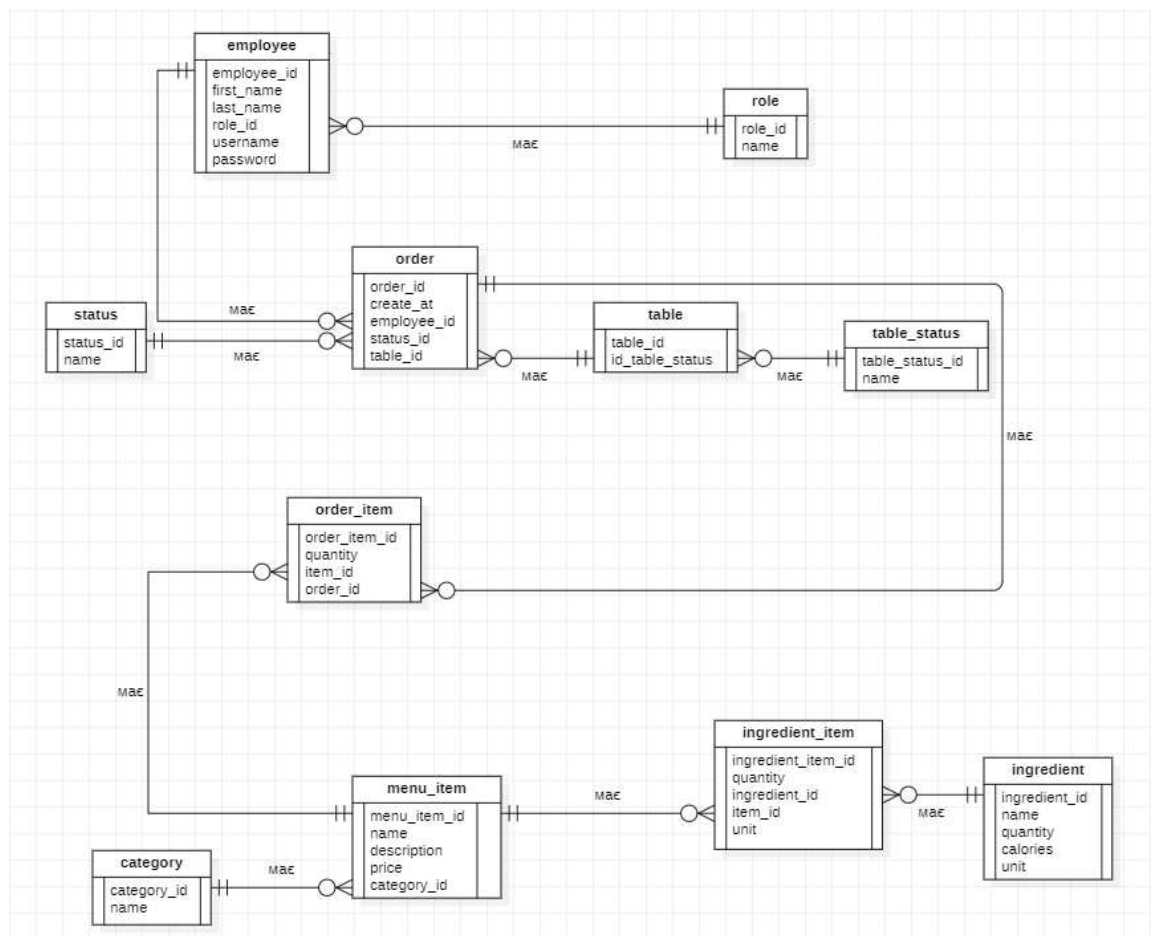


Рисунок 2.3 – Логічна модель бази даних

Для кожної таблиці чітко визначається первинний ключ, який є унікальним ідентифікатором, що однозначно розпізнає кожен рядок у таблиці. Також можуть присутні зовнішні ключі, які встановлюють зв'язки між

таблицями. Зовнішній ключ - це стовпець у одній таблиці, що посилається на первинний ключ у іншій таблиці.

На основі розробленої логічної моделі бази даних розроблено фізичну модель бази даних. Фізичну модель бази даних зображено на рисунку 2.4.

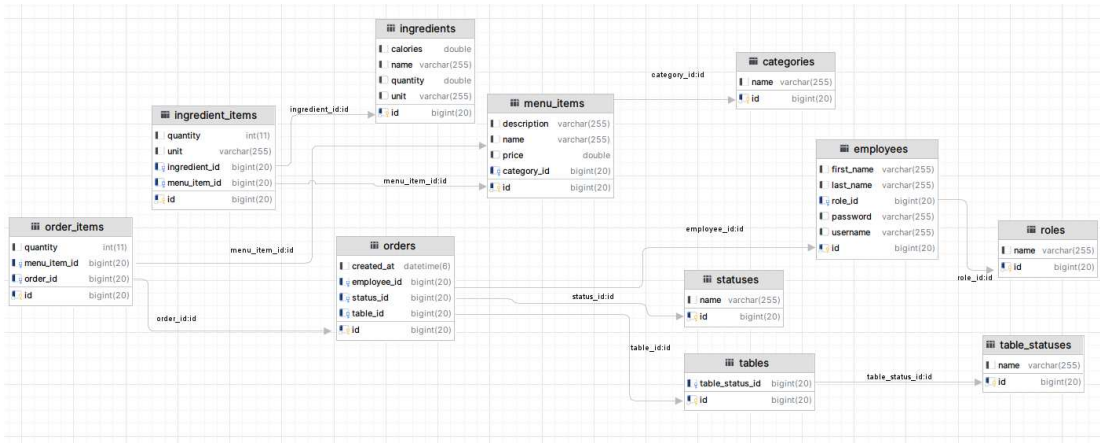


Рисунок 2.4 – Фізична модель бази даних

В таблицях 2.2 – 2.12 наведено характеристики атрибутів таблиць БД.

Таблиця 2.2. Схема відношення «Employee»

Ім'я поля	Тип	Обмеження цілісності	Ключі
employee_id	Long	Not Null	PK
first_name	String	Not Null	
last_name	String	Null	
role_id	Long	Not Null	FK (до Role)
username	String	Not Null	
password	String	Not Null	

Таблиця 2.3. Схема відношення «Category»

Ім'я поля	Тип	Обмеження цілісності	Ключі
category_id	Long	Not Null	PK
name	String	Not Null	

Таблиця 2.4. Схема відношення «Ingredient»

Ім'я поля	Тип	Обмеження цілісності	Ключі
ingredient_id	Long	Not Null	PK
name	String	Not Null	
quantity	Double	Not Null	
unit	String	Not Null	
calories	Double	Not Null	

Таблиця 2.5 Схеми відношення «Ingredient_Item»

Ім'я поля	Тип, довжина	Обмеження цілісності	Ключі
ingredient_item_id	Long	Not Null	PK
item_id	Long	Not Null	FK (до Menu_Item)
ingredient_id	Long	Not Null	FK (до Ingredient)
quantity	Integer	Not Null	
unit	String	Not Null	

Таблиця 2.6 Схеми відношення «Menu_Item»

Ім'я поля	Тип, довжина	Обмеження цілісності	Ключі
menu_item_id	Long	Not Null	PK
Name	String	Not Null	
description	String	Not Null	
Price	Double	Not Null	
category_id	Long	Not Null	FK (до Category)

Таблиця 2.7 Схеми відношення «Order»

Ім'я поля	Тип, довжина	Обмеження цілісності	Ключі
order_id	Long	Not Null	PK
table_id	Long	Null	FK (до Table)
created_at	Date	Not Null	
status_id	Long	Not Null	FK (до Status)
employee_id	Long	Not Null	FK (до Employee)

Таблиця 2.8 Схеми відношення «Status»

Ім'я поля	Тип, довжина	Обмеження цілісності	Ключі
status_id	Long	Not Null	PK
Name	String	Not Null	

Таблиця 2.9 Схеми відношення «Role»

Ім'я поля	Тип, довжина	Обмеження цілісності	Ключі
role_id	Long	Not Null	PK
Name	String	Not Null	

Таблиця 2.10 Схеми відношення «Table_Status»

Ім'я поля	Тип, довжина	Обмеження цілісності	Ключі
table_status_id	Long	Not Null	PK
Name	String	Not Null	

Таблиця 2.11 Схема відношення «Table»

Ім'я поля	Тип, довжина	Обмеження цілісності	Ключі
Id	Long	Not Null	PK
table_status_id	Long	Not Null	FK (до Table Status)

Таблиця 2.12. Схема відношення «Order_item»

Ім'я поля	Тип, довжина	Обмеження цілісності	Ключі
order_item_id	Long	Not Null	PK
quantity	Long	Not Null	
order_id	Long	Not Null	FK (до Order)
item_id	Long	Not Null	FK (до Menu_item)

2.3 Рішення з технічного забезпечення

Для забезпечення ефективної роботи інформаційної системи (ІС) для управління рестораном, необхідно врахувати низку вимог до технічного забезпечення. Ці вимоги визначають мінімальні характеристики апаратних та програмних компонентів, необхідних для коректного функціонування системи.

Мінімальні вимоги до серверного обладнання:

- Багатоядерний процесор з частотою не менше 2.4 ГГц.
- Оперативна пам'ять мінімум 8 ГБ.
- Жорсткий диск від 500 ГБ або SSD від 256 ГБ.
- Гігабітний Ethernet адаптер.
- Операційна система: Linux.

Мінімальні вимоги до робочих ноутбуків чи ПК:

- Двоядерний процесор з частотою не менше 2.0 ГГц.
- Оперативна пам'ять мінімум 4 ГБ.
- Жорсткий диск від 120 ГБ.
- Ethernet адаптер або Wi-Fi модуль.

- Операційна система Windows або Linux.

Мінімальні вимоги до мобільних пристроїв:

- Чотириядерний процесор.
- Оперативна пам'ять мінімум 2 ГБ.
- Накопичувач від 16 ГБ.
- Операційна система Android 8.0 і вище або iOS 12.

Загальні вимоги до мережевого обладнання:

- Маршрутизатор з підтримкою гігабітних швидкостей.
- Гігабітні комутатори з достатньою кількістю портів.
- Wi-Fi точка доступу з підтримкою стандарту 802.11ac або вище.

2.4 Рішення з програмного забезпечення

2.4.1 Розробка схеми структури інформаційної системи

Провівши детальний аналіз вимог та наявних технологій, для розробки інформаційної системи управління рестораном було обрано Spring MVC. Цей фреймворк забезпечує зручність у розробці веб-додатків, підтримуючи архітектурний шаблон MVC (Model-View-Controller), що дозволяє розділити логіку додатку на три взаємопов'язані компоненти: модель, уявлення та контролер. Такий підхід підвищує масштабованість, зручність підтримки та розширення системи. Використання Spring MVC у поєднанні з клієнт-серверною архітектурою забезпечує чіткий розподіл обов'язків між клієнтом та сервером, дозволяючи оптимізувати взаємодію та обробку даних. Це також сприяє більшій безпеці та продуктивності системи за рахунок централізованого управління логікою та доступом до бази даних[6].

Основні компоненти інформаційної підсистеми:

1. Веб-додаток (Spring MVC):

- 1.1.Інтерфейс для персоналу (офіціанти, адміністратори, кухарі) для створення та обробки замовлень.
- 1.2.Інтерфейс для адміністраторів для управління меню, перегляду звітів та аналітики.
- 1.3.Використання шаблонів Thymeleaf для відображення даних.
2. База даних:
 - 2.1.Центральне сховище даних, яке містить інформацію про замовлення, клієнтів, меню, інгредієнти та інші важливі дані.
 - 2.2.Використання ORM Hibernate для взаємодії з базою даних.
3. Система авторизації:
 - 3.1.Модуль для реєстрації та входу користувачів, реалізований за допомогою Spring Security.
 - 3.2.Розмежування прав доступу до різних частин системи на основі ролей (офіціант, адміністратор, кухар).
4. Функціональні модулі:
 - 4.1.Модуль управління замовленнями: створення, редагування та перегляд замовлень.
 - 4.2.Модуль управління меню: додавання, редагування та видалення пунктів меню.
 - 4.3.Модуль звітності: генерація фінансових та аналітичних звітів.
 - 4.4.Модуль інвентаризації: облік інгредієнтів та контроль запасів.

На рисунку 2.5 представлено схему взаємодії програмних модулів.

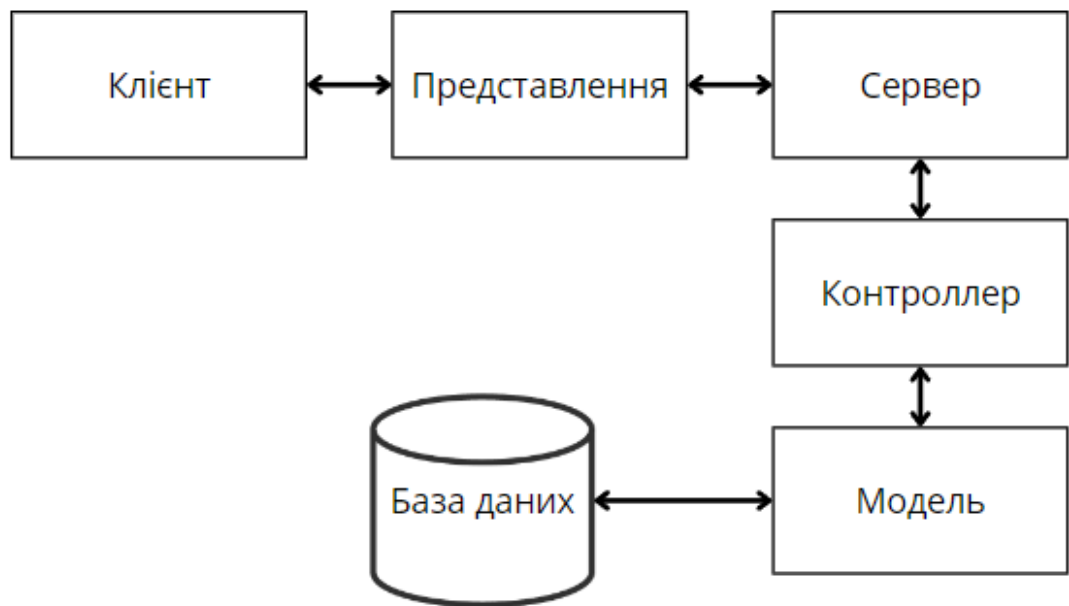


Рисунок 2.5 – Схема взаємодії програмних модулів

Опис взаємодії модулів:

Клієнт: Веб-браузер або мобільний додаток відправляє HTTP-запит до сервера. Клієнтська частина може бути реалізована з використанням технологій HTML, CSS, JavaScript (наприклад, React або Angular).

Уявлення (View): Відповідає за формування інтерфейсу користувача. Веб-додаток відображає дані, отримані від сервера, у зручному для користувача форматі (HTML, CSS, JavaScript).

Модуль управління замовленнями: дає можливість формувати та редагувати замовлення, додавати страви до замовлення, закріплювати його до столика та офіціанта, що його створив, ввести замовлення від моменту створення, до оплати замовлення.

Модуль управління меню: дає можливість перегляду меню, його редагування. Система сортує страви відповідно категорій для зручнішого перегляду.

Модуль звітності: система створює звітність про найпопулярніші страви. Також система відображає продуктивність кожного офіціанта.

Модуль інвентаризації: інформаційна система відображає запас продуктів на складі, автоматично змінює їх кількість відповідно до кількості для кожного блюда, при створенні замовлення .

База даних: База даних зберігає всі дані системи, включаючи інформацію про замовлення, клієнтів, меню та інші важливі дані. Для взаємодії з базою даних використовується ORM, який забезпечує абстракцію від безпосереднього використання SQL.

2.4.2 Розробка діаграми компонентів

Діаграми компонентів використовуються для моделювання великих систем на менші підсистеми, якими можна легко керувати. Діаграми компонентів UML використовуються для представлення різних компонентів системи. При моделюванні великих об'єктно-орієнтованих систем необхідно розбивати систему на керовані підсистеми [7].

На рисунку 2.6 зображено діаграму компонентів системи.

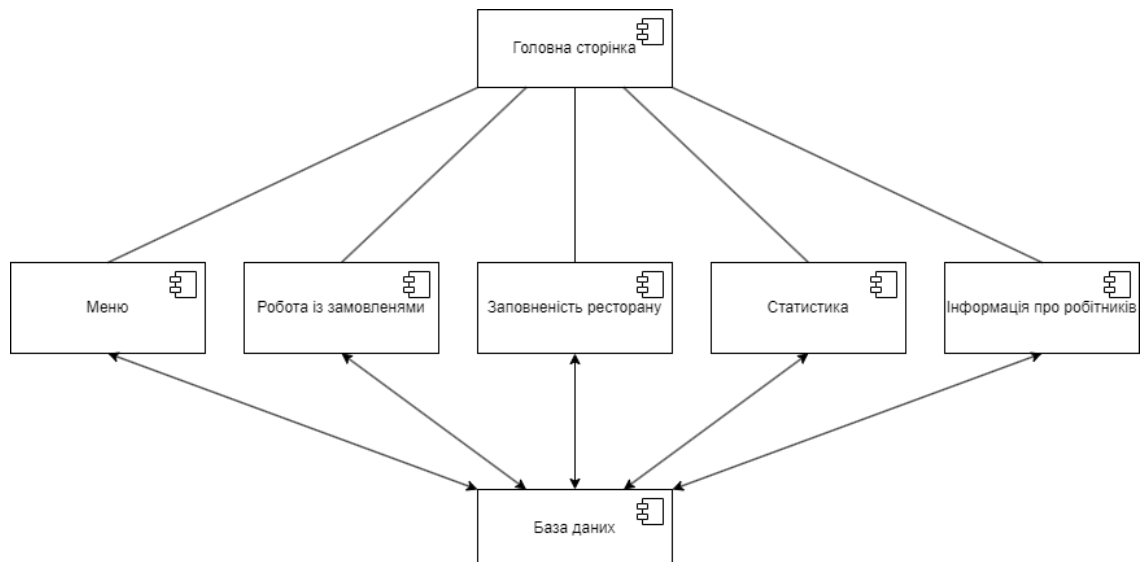


Рисунок 2.6 – Діаграма компонентів

2.4.3 Розробка діаграми переходів станів

Діаграма станів – це графічне уявлення процесу чи системи, що дозволяє уявити всі можливі стани та переходи з-поміж них. Вона є одним із ключових інструментів розробки та аналізу програмного забезпечення,

Основною метою створення діаграми станів є візуалізація логіки поведінки системи та визначення різних сценаріїв, які можуть виникнути у процесі її роботи. Це допомагає краще зрозуміти і пояснити, як система повинна поводитися в різних ситуаціях, а також виявити потенційні проблеми та покращити проект ще на ранній стадії розробки [8].

На рисунку 2.7 зображено діаграму переходів станів для адміна.

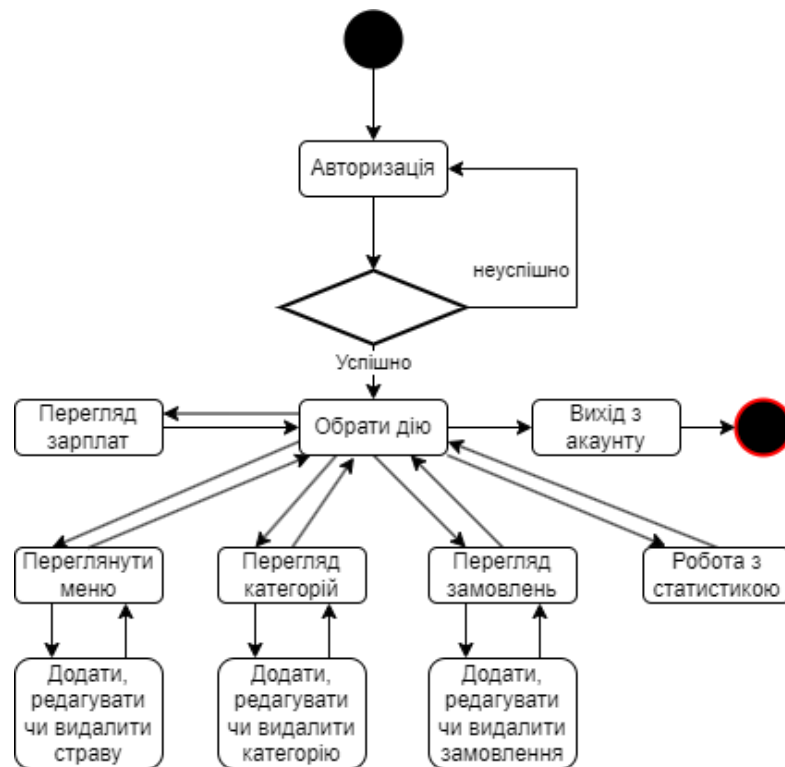


Рисунок 2.7 – Діаграма переходу станів для адміна

В адміністратора першим станом є авторизація, якщо користувач буде успішно авторизований, він зможе перейти до наступного стану вибору дії, в іншому випадку користувач залишиться в стані авторизації. Наступним станом може бути перегляд зарплат. Також одним зі станів може бути перегляд меню і в ньому вже можна перейти в стан редагування, видалення чи

додавання страв. Можна обрати перегляд категорій, з нього в свою чергу можна перейти в стан редагування, додавання чи видалення категорій. Користувач може перейти в стан перегляду замовлень, який надає можливість перейти в стан додавання, редагування чи видалення замовлень. Також адмін має можливість перейти в стан перегляду статистики. Останній стан це можливість виходу з акаунту.

На рисунку 2.8 зображена діаграма переходу станів для повара та офіціанта.

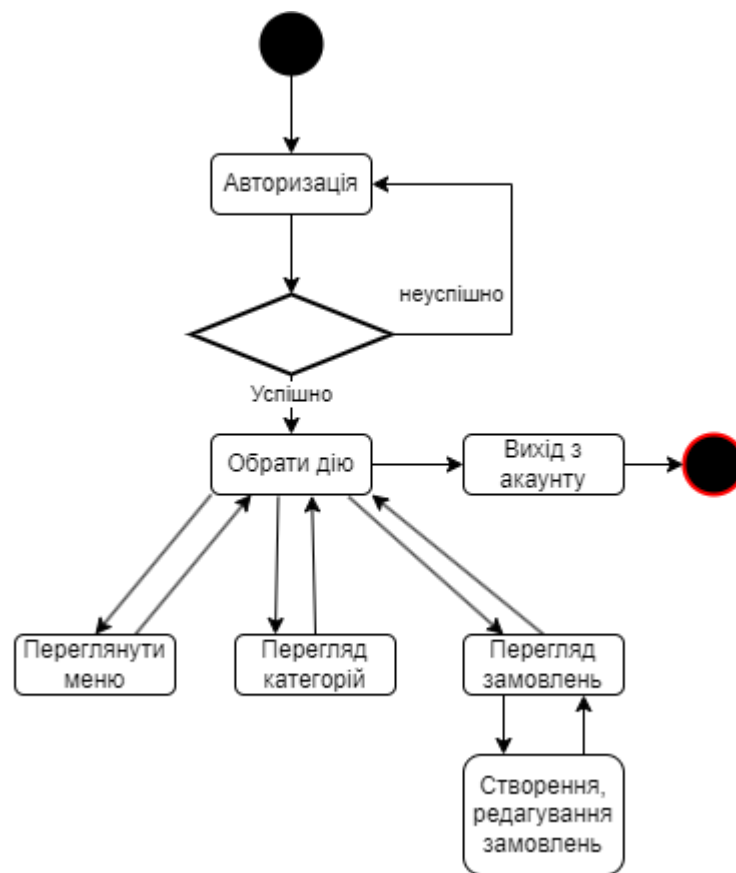


Рисунок 2.8 – Діаграма переходу станів для офіціанта та повара

В офіціанта чи повара першим станом є авторизація, якщо користувач буде успішно авторизований, він зможе перейти до наступного стану вибору дії, в іншому випадку користувач залишиться в стані авторизації. Наступним станом може бути перегляд меню або категорій. Також є можливість перейти в стан перегляду замовлень, що в свою чергу дає можливість перейти в стан

створення, редагувань замовлень. Офіціант чи повар також мають можливість обрати стан виходу із акаунту.

2.4.4 Розробка діаграми розгортання підсистеми

Діаграма розгортання — це тип діаграми, який визначає фізичне обладнання, на якому працюватиме програмна система. Він також визначає спосіб розгортання програмного забезпечення на базовому обладнанні. Він прив'язує частини програмного забезпечення системи до пристрою, який збирається її виконувати.

Діаграма розгортання відображає програмне забезпечення та архітектуру, створену в процесі розробки фізичної системи. У розподілених системах він моделює розподіл програмного забезпечення між фізичними вузлами [9].

Діаграма розгортання системи зображено на рисунку 2.9.

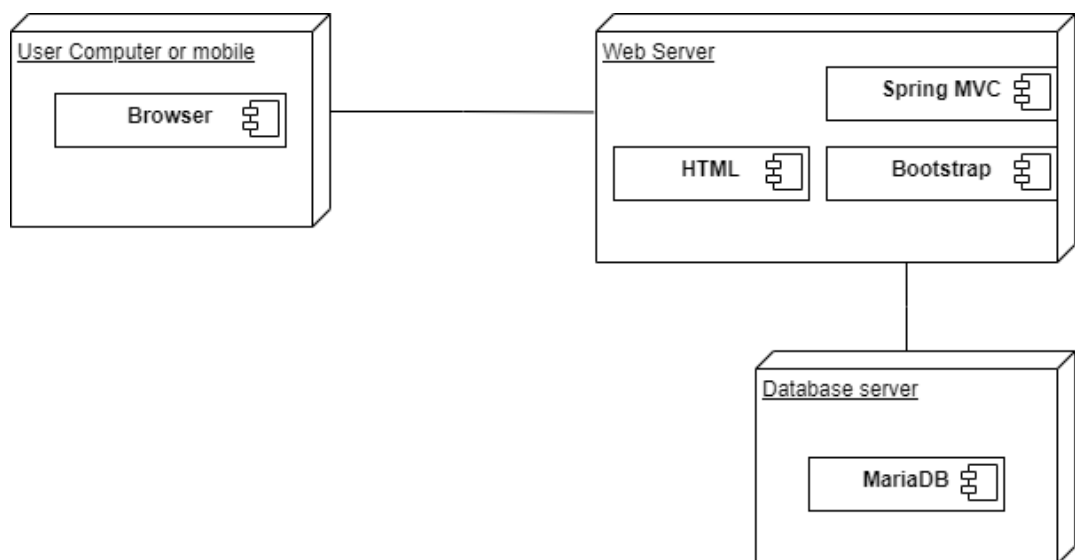


Рисунок 2.9 – Діаграма розгортання системи

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЕКТУ ІНФОРМАЦІЯ СИСТЕМИ ДЛЯ УПРАВЛІННЯ РЕСТОРАНОМ

3.1 Аргументація вибору засобів розробки

IntelliJ IDEA було обрано як середовище розробки (IDE) завдяки його потужним функціям для розробки на Java. Вона забезпечує інтеграцію з популярними фреймворками та інструментами, такими як Spring та JPA, а також має підтримку систем контролю версій, таких як Git. IntelliJ IDEA пропонує інтелектуальні підказки коду, автозавершення, аналіз коду в реальному часі та багато інших інструментів, що прискорюють розробку та зменшують кількість помилок [10].

Java є однією з найпопулярніших мов програмування для створення серверних додатків. Вона забезпечує високу продуктивність, безпеку та масштабованість. Крім того, Java має велику екосистему бібліотек та фреймворків, що спрощують розробку складних додатків. Зокрема, Java добре підходить для розробки корпоративних додатків завдяки своїй стабільності та кросплатформенності [11].

MariaDB було обрано як систему управління базами даних (СУБД) через її високу продуктивність, надійність та сумісність з MySQL. MariaDB є відкритим програмним забезпеченням, що робить її економічно вигідним вибором для багатьох проектів. Вона також підтримує сучасні можливості реляційних баз даних, включаючи транзакції та реплікацію [12].

phpMyAdmin є зручним інструментом для управління базами даних MariaDB через веб-інтерфейс. Він дозволяє легко виконувати різноманітні операції з базами даних, включаючи створення та редагування таблиць, запуск SQL-запитів, імпорт та експорт даних. phpMyAdmin підтримує широкий спектр функцій, що робить його корисним для адміністраторів баз даних [13].

Spring Framework було обрано для розробки цього проекту завдяки його модульності та можливості інтеграції з іншими технологіями. Spring забезпечує потужну інфраструктуру для створення Java-додатків, включаючи підтримку інверсії керування (IoC), аспектно-орієнтованого програмування (AOP), транзакцій та багато іншого. Spring також пропонує широкий набір модулів для інтеграції з іншими технологіями та фреймворками, такими як Spring Security та Spring Data JPA [14].

Spring Security було обрано для реалізації механізмів автентифікації та авторизації у проекті. Цей модуль забезпечує потужні інструменти для управління безпекою додатка, включаючи підтримку різних методів автентифікації, а також гнучке управління доступом на основі ролей та дозволів [15].

JPA використовується для управління персистенцією даних в додатку. JPA надає зручний API для роботи з базами даних, дозволяючи розробникам працювати з об'єктами Java, а не з SQL-запитами. Це сприяє кращій абстракції та полегшує роботу з базою даних [16].

Для створення веб-інтерфейсу було обрано HTML та Bootstrap. HTML є основною мовою розмітки для веб-сторінок, тоді як Bootstrap надає потужний фреймворк для швидкої розробки адаптивних веб-інтерфейсів. Bootstrap включає в себе набір готових компонентів та стилів, що значно прискорює процес розробки та забезпечує консистентний та професійний вигляд додатка [17].

3.2 Розробка структури програмного забезпечення

Перша сторінка для будь-якого користувача є сторінка авторизації. Сторінка авторизації має два поля – це логін та пароль. Дані для авторизації видаються адміністратором при створенні користувача.

Після авторизації, користувач направляється на головну сторінку, на якій буде зображено навігаційну панель, користувач може перейти на сторінки згідно свого доступу, до цих сторінок. Якщо користувач спробує відкрити сторінку сайту, до якої він не має доступу, він отримає попередження. На головній сторінці зображено дві таблиці. Перша таблиця вміщує всі замовлення зі статусом «Готується», друга таблиця за статусом «Готово». В таблиці розписана головна інформація про блюдо.

Якщо потрібно дізнатись більш детальну інформацію про замовлення, потрібно натиснути на ід замовлення. Схожий підхід використаний для блюда, якщо користувач захоче дізнатись більш детальну інформацію про блюдо, він має натиснути на його назву.

Всі користувач мають доступ до сторінки меню. Сторінка включає таблицю, яка відображає детальну інформацію про кожне блюдо, страви згруповано відповідно категорій для більш зручного пошуку страви. Для отримання технологічної карти блюда, потрібно натиснути на калорійність цього блюда, після чого відкриється нова сторінка з описом ваги, кількості та калорійності кожного інгредієнта цієї страви.

Наступна сторінка на навігаційній панелі – створення замовлення. Ця сторінка включає п'ять форм, їх кількість може змінюватись залежно від кількості блюд. Перша форма - це список, який відображає вільні столики. Друга форма – це список офіціантів, користувач який створює замовлення має обов'язково вибрати свою фамілію зі списку. Наступна форма надає можливість вибору статусу замовлення, за замовчуванням це статус «Готується». Остання форма – вибір блюда, та вказання, його кількості.

Вільні столики – наступна сторінка в навігаційній панелі. Вона вміщає в собі табличку з номерами столиків та їх статусами. Такий підхід надає зрозуміле пояснення наповненості залу.

Для адміністратора є доступною сторінка – «Зарплати». Ця сторінка відображає кожного працівника та його зарплату. Зарплата кожного офіціанта

підраховується згідно системи бонусів, що дає 30 гривень за кожне замовлення.

Наступна сторінка також доступна лише для адміністратора – це «Статистика». Вона дозволяє переглянути кількість закритих замовлень, їх суму, за вказаний період часу.

Остання сторінка є доступною для всіх користувачів – «Всі замовлення». Вони відображені в форматі таблиці, замовлення сортуються за датою, щоб переглянути детальну інформацію про склад замовлення, користувач має натиснути на ід замовлення.

Графічна частина інформаційної системи використовує спеціально підібрані кольорові гами для підвищення інтуїтивності роботи з інформаційною системою.

На рисунку 3.1 представлено діаграму класів модульного рівня інформаційної системи.

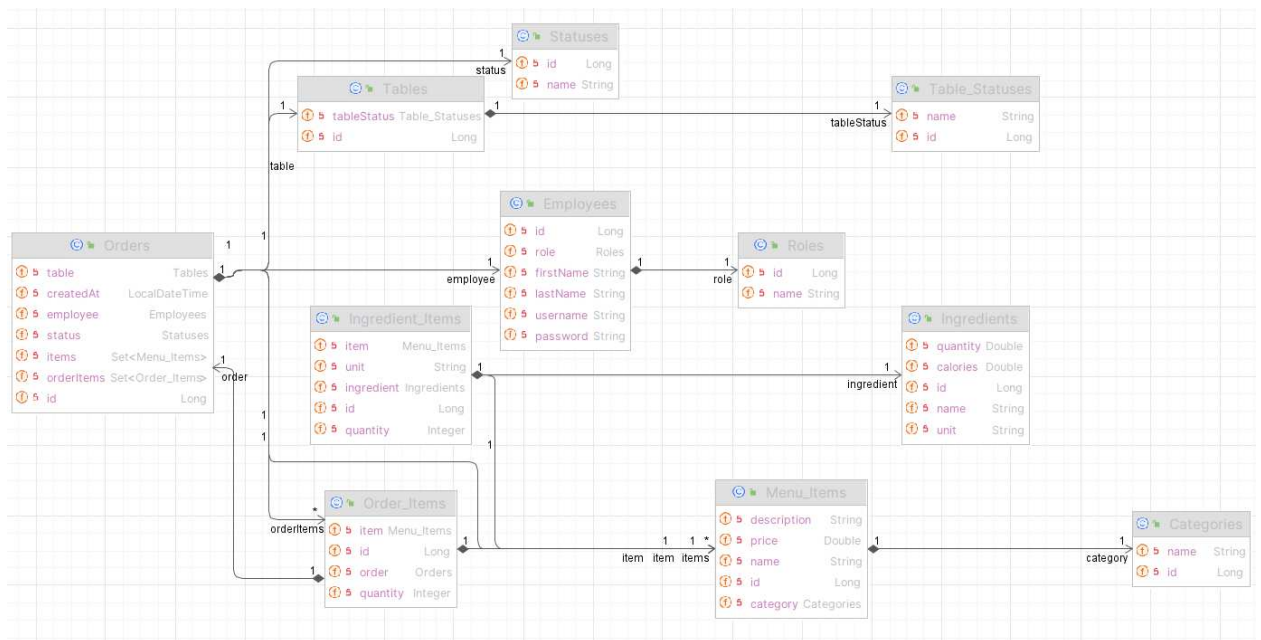
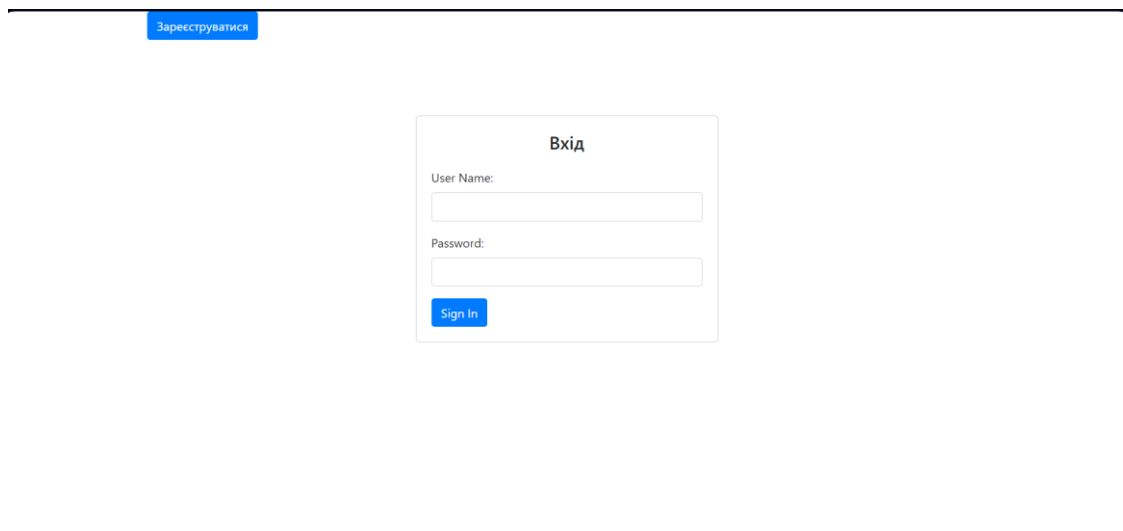


Рисунок 3.1 – Діаграма класів

3.3 Результат розробки програмного забезпечення

У результаті виконання кваліфікаційної роботи реалізовано заплановану мету проекту. Розроблено наступний функціонал: сторінка авторизації та реєстрації, головна сторінка, меню, створення замовлення, вільні столики, зарплата, статистика, всі замовлення, категорії.

При вході в додаток вимагається пройти авторизацію, сторінку авторизації зображено на рисунку 3.2.



Зареєструватися

Вхід

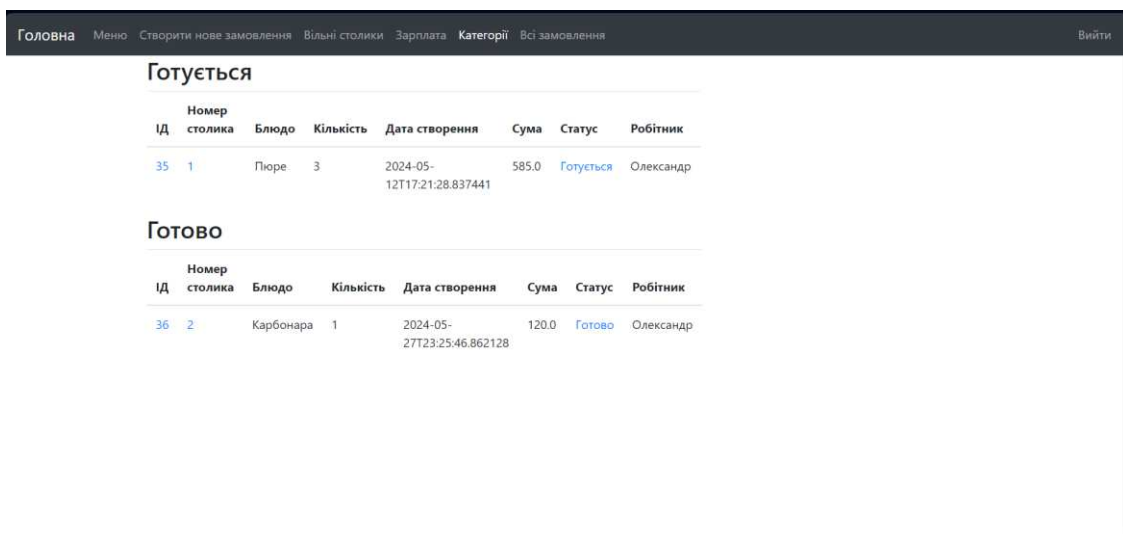
User Name:

Password:

Sign In

Рисунок 3.2 – Сторінка авторизації

Пройшовши авторизацію та аутентифікацію, користувач попадає на головну сторінку. Головну сторінку зображено на рисунку 3.3.



Головна Меню Створити нове замовлення Вільні столики Зарплата Категорії Всі замовлення Вийти

Готується

ІД	Номер столика	Блюдо	Кількість	Дата створення	Сума	Статус	Робітник
35	1	Пюре	3	2024-05-12T17:21:28.837441	585.0	Готується	Олександр

Готово

ІД	Номер столика	Блюдо	Кількість	Дата створення	Сума	Статус	Робітник
36	2	Карбонара	1	2024-05-12T23:25:46.862128	120.0	Готово	Олександр

Рисунок 3.3 – Головна сторінка

Натиснувши кнопку меню, користувач перенаправляється на сторінку за списком всіх страв. Сторінку меню зображено на рисунку 3.4.

Головна Меню Створити нове замовлення Вільні столики Зарплата Категорії Всі замовлення Вийти

Меню

Айди	Назва	Опис	Ціна	Категорія	Калорійність
3	Сковорідка зі свининою	(картопля, гриби, свинина, морква, фасоль, томатна паста, цибуля)	148.0	Гарячі страви	285.6
4	Пюре з ковбасками гриль	(картопля, ковбаски-гриль)	168.0	Гарячі страви	151.5
5	Мохіто	(спрайт, лайм, лід, м'ята)	35.0	Напої	58.3
6	Зелений чай	Чай "Ліптон"	15.0	Напої	0.06
7	Філадельфія	(лосось, рис, сир, огірок)	160.0	Роли	136.7
8	Каліфорнія	(лосось, сир, авокадо, огірок)	200.0	Роли	112.8
9	Цезарь	(курка, листя салату, яйця, соус, помідори черрі, пармезан)	150.0	Салати та закуски	155.48
10	Салат з тунцем	(тунець, листя салату, яйця, картопля, соус)	135.0	Салати та закуски	62.78
11	Пюре	(картопля)	65.0	Гарніри	22.5

Рисунок 3.4 – Меню

Наступна в нас сторінка «Створити замовлення». Ця сторінка доступна всім авторизованим користувачам. Сторінку створити замовлення зображено на рисунку 3.5.

Головна Меню Створити нове замовлення Вільні столики Зарплата Категорії Всі замовлення Вийти

Створити нове замовлення

Номер столика:

1

Офіціант:

Козлов

Статус:

Готується

Блюда:

Сковорідка зі свининою

2

Блюда:

Мохіто

1

Блюда:

Зелений чай

2

Добавити страву

Зберегти

Рисунок 3.5 – Створення замовлення

Наступна сторінка «Вільні столики», на рисунку 3.5 ми вказали, що замовлення готується для першого столику. Перейшовши на сторінку, відображено, що статус столика автоматично змінився після створення замовлення. Сторінку «Вільні столики» зображено на рисунку 3.6.

Головна	Меню	Створити нове замовлення	Вільні столики	Зарплата	Категорії	Всі замовлення	Вийти
Столики							
Ід	Статус						
1	Занято						
2	Вільно						
3	Вільно						

Рисунок 3.6 – Сторінка «Вільні столики»

Сторінка «Зарплата» включає в себе перегляд таблиці із зарплатнею кожного співробітника. Кожен користувач має різну зарплатню відповідно своїх показників. Сторінку «Зарплата» зображено на рисунку 3.7.

Головна	Меню	Створити нове замовлення	Вільні столики	Зарплата	Категорії	Всі замовлення	Вийти
Зарплати							
Ім'я	Фамілія	Зарплата					
Олександр	Козлов	12360.0					
Іван	Іванов	12810.0					

Рисунок 3.7 – Сторінка «Зарплата»

Текст програми наведено в додатку Б.

Інструкції користувачів представлені в додатку В.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ

4.1 Загальні питання охорони праці

Охорона праці – це комплекс заходів та засобів, спрямованих на забезпечення безпеки, збереження здоров'я та працездатності працівників у процесі їхньої трудової діяльності. Основною метою охорони праці є створення безпечних і здорових умов праці на кожному робочому місці, запобігання виробничому травматизму та професійним захворюванням.

В Україні охорона праці регулюється Законом України "Про охорону праці", який визначає основні принципи державної політики у сфері охорони праці, зокрема, пріоритет життя та здоров'я працівників перед виробничими інтересами, повну відповідальність роботодавця за створення безпечних умов праці та соціальний захист працівників, які постраждали від нещасних випадків на виробництві. Крім того, важливу роль відіграють нормативно-правові акти, санітарні норми та правила, стандарти безпеки праці та гігієнічні норми [18].

Законодавство України вимагає, щоб всі працівники були ознайомлені з правилами та інструкціями з охорони праці, а роботодавець зобов'язаний організовувати навчання з питань охорони праці та проводити інструктажі на робочих місцях. Це включає первинний, повторний, позаплановий та цільовий інструктажі, які повинні проводитися у відповідні терміни і з урахуванням специфіки виконуваних робіт.

4.2 Структура управління питаннями охорони праці на підприємстві

На підприємстві система управління охороною праці включає кілька ключових елементів, що забезпечують реалізацію державної політики у цій сфері та дотримання відповідних норм і правил.

Керівник підприємства несе загальну відповідальність за стан охорони праці, розробляє і впроваджує політику в цій сфері, забезпечує фінансування та ресурсне забезпечення заходів з охорони праці, а також контролює виконання запланованих заходів.

Служба охорони праці – це спеціалізований підрозділ або уповноважені особи, які відповідають за організацію, координацію та контроль за виконанням заходів з охорони праці. Основними функціями служби охорони праці є: аналіз стану охорони праці на підприємстві, розробка і впровадження програм і планів заходів з охорони праці, контроль за виконанням нормативних вимог, проведення інструктажів та навчання працівників.

Комісія з охорони праці – робочий орган, який складається з представників адміністрації та трудового колективу. Основні завдання комісії включають: розгляд і затвердження планів заходів з охорони праці, аналіз стану охорони праці, участь у розслідуванні нещасних випадків та професійних захворювань, підготовка пропозицій щодо покращення умов праці.

Уповноважений з охорони праці від трудового колективу обирається працівниками для представлення їх інтересів у питаннях охорони праці. Він бере участь у роботі комісії з охорони праці, контролює дотримання нормативних вимог та захищає права працівників на безпечні умови праці.

Інші посадові особи на підприємстві також несуть відповідальність за охорону праці в межах своїх посадових обов'язків. Це можуть бути керівники структурних підрозділів, інженери з техніки безпеки, начальники цехів та відділів. Вони зобов'язані забезпечувати виконання заходів з охорони праці, проведення інструктажів та контролювати дотримання правил безпеки [19].

4.3 Шкідливі та небезпечні фактори, що супроводжують працю програміста

Робота програміста, як і будь-яка інша професійна діяльність, пов'язана з певними шкідливими та небезпечними факторами, які можуть негативно впливати на здоров'я працівників. Основні з них включають:

Тривале перебування в сидячій позі. Постійне сидіння за комп'ютером може призвести до проблем з опорно-руховим апаратом, таких як остеохондроз, сколіоз, болі в спині та ший. Відсутність фізичної активності сприяє розвитку застійних явищ у м'язах та суглобах.

Високе навантаження на зір. Тривала робота з комп'ютером викликає зорову втому, сухість очей, зниження гостроти зору, а також може призвести до розвитку короткозорості або інших проблем з очима. Яскравість екрану, відблиски та неправильне освітлення додатково погіршують зорові функції.

Монотонність та статичність роботи. Постійна концентрація на одному виді діяльності, відсутність зміни положення тіла та монотонність можуть сприяти розвитку психоемоційного напруження, перевтоми, стресу, депресивних станів.

Електромагнітне випромінювання. Робота з комп'ютерним обладнанням супроводжується впливом електромагнітних полів, які можуть негативно впливати на нервову систему та загальний стан здоров'я. Хоча сучасні комп'ютери випромінюють менше, ніж раніше, ризик все одно існує.

Недостатнє освітлення або неправильне розташування джерел світла. Це може викликати додаткове навантаження на зір, сприяти розвитку головного болю, підвищувати ризик травматизму через погану видимість.

Психологічний стрес. Висока інтенсивність роботи, часті дедлайни, необхідність постійного навчання та освоєння нових технологій можуть призводити до психологічного стресу та вигорання.

Для мінімізації шкідливого впливу цих факторів необхідно застосовувати комплексний підхід, включаючи ергономічну організацію

робочого місця, регулярні перерви та фізичні вправи, використання спеціальних засобів для захисту зору, а також проведення профілактичних медичних оглядів [20].

4.4 Ергономічні вимоги до робочого місця при роботі за комп'ютером

Ергономічна організація робочого місця програміста є ключовим фактором для забезпечення комфортних і безпечних умов праці, що сприяє підвищенню продуктивності та зниженню ризику професійних захворювань. Основні ергономічні вимоги включають:

Розташування монітора. Верхній край екрану повинен знаходитися на рівні очей або трохи нижче, щоб зменшити напруження м'язів шиї. Відстань до екрана повинна становити 50-70 см, що дозволяє уникнути перенапруги зору.

Стіл та стілець. Стіл повинен бути достатньо просторим для розміщення всього необхідного обладнання та матеріалів. Стілець має бути регульованим по висоті, з можливістю нахилу спинки та підтримкою попереку. Підлокітники повинні забезпечувати підтримку рук, знижуючи навантаження на плечі та шиї.

Клавіатура та миша. Розташування клавіатури та миші повинно бути на рівні ліктів або трохи нижче. Руки мають залишатися в природному положенні, з мінімальним напруженням. Рекомендується використовувати спеціальні підставки для зап'ясть, щоб запобігти розвитку тунельного синдрому.

Освітлення. Робоче місце повинно бути добре освітлене. Бажано використовувати природне освітлення, а також настільні лампи з регульованою яскравістю. Джерела світла не повинні створювати відблисків на екрані монітора.

Перерви та гімнастика. Для запобігання втомі та збереження працездатності рекомендується робити регулярні короткі перерви кожні 60 хвилин роботи за комп'ютером. Під час перерв варто виконувати прості фізичні вправи для зняття напруги з м'язів ший, спини, рук та очей.

Температурний режим і вентиляція. Температура в робочому приміщенні повинна бути комфортною, приблизно 22-24°C. Приміщення має бути добре провітрюване, щоб забезпечити приток свіжого повітря.

Акустичні умови. Рівень шуму не повинен перевищувати 50 дБ, що дозволяє уникнути додаткового навантаження на нервову систему та підвищує концентрацію [21].

4.5 Нормативні вимоги до робочої площі на одного працюючого за комп'ютером

Нормативні вимоги щодо робочої площі на одного працюючого за комп'ютером регламентуються санітарними нормами та правилами, які забезпечують створення безпечних та комфортних умов праці. Основні положення включають:

Робоча площа. На одного працівника повинна бути передбачена площа не менше 6 м². Ця площа включає місце для розміщення столу, стільця, комп'ютерного обладнання та проходів. Дотримання цього нормативу дозволяє уникнути скупченості та забезпечити комфортні умови праці.

Висота приміщення. Висота приміщення повинна бути не менше 3 м. Це забезпечує достатній об'єм повітря для нормального самопочуття працівника та сприяє кращій вентиляції робочого простору.

Вентиляція та кондиціонування. Приміщення має бути обладнане системами вентиляції та кондиціонування для забезпечення комфортного мікроклімату. Вентиляція повинна забезпечувати свіжість повітря та підтримувати оптимальну температуру і вологість.

Акустика. Рівень шуму в робочому приміщенні не повинен перевищувати 50 дБ. Зниження рівня шуму дозволяє уникнути додаткового навантаження на нервову систему, підвищити концентрацію та продуктивність праці.

Організація простору. Робочі місця повинні бути розташовані таким чином, щоб забезпечити вільний доступ до кожного з них. Це включає забезпечення достатньої ширини проходів, вільний доступ до виходів та евакуаційних шляхів у разі надзвичайної ситуації.

Меблі та обладнання. Робочі місця повинні бути обладнані ергономічними меблями та сучасним комп'ютерним обладнанням, яке відповідає вимогам безпеки та комфорту. Меблі мають бути регульованими по висоті та забезпечувати підтримку всіх необхідних частин тіла.

Санітарні умови. Приміщення повинно регулярно прибиратися та провітрюватися для підтримання чистоти та свіжості. Санітарні умови включають наявність засобів для гігієни, доступ до питної води та обладнаних місць для прийому їжі [21].

Дотримання цих вимог сприяє збереженню здоров'я працівників, підвищенню їхньої працездатності та ефективності праці. Належна організація робочих місць, забезпечення комфортного мікроклімату та безпечних умов праці є важливими складовими успішної діяльності будь-якого підприємства.

ВИСНОВОК

У розробленому проекті реалізовано інформаційну систему для управління рестораном. Головною метою цього проекту було покращення та прискорення процесів комунікації за допомогою автоматизації процесів роботи в команді.

Процес розробки системи розпочався з формулювання концепції та аналізу вимог до неї. На основі цього була розроблена концепція, яка визначила функціонал інформаційної системи. Технічний проект був визначений, у тому числі інформаційне та програмне забезпечення, а також технічне забезпечення системи.

Впровадженням Spring MVC для реалізації клієнт-серверної архітектури забезпечено ефективне та зручне взаємодію з системою. Використання браузера Opera та доступу до Інтернету у сполученні з цими архітектурними рішеннями дозволило забезпечити високу якість та продуктивність розробки.

Проведена апробація проекту підтвердила коректну роботу всіх його компонентів та модулів. Завершальні документи включають правила охорони праці під час використання комп'ютера.

Цей проект відзначається своєю актуальністю, оскільки значно покращує зручність управління рестораном та швидкість обслуговування клієнтів, що дозволяє отримувати більше прибутку закладу. Тепер користувачі можуть легко та зручно керувати своїми замовленнями з будь-якого пристрою, що підвищує конкурентоспроможність у цій галузі та забезпечує привабливість для нових клієнтів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Автоматизація їдальні [Електронний ресурс] – Режим доступу до ресурсу: <https://joinposter.com/ua/business/canteen>.
2. Kavapp - Особистий помічник для власника кав'ярні! [Електронний ресурс] – Режим доступу до ресурсу: https://kavapp.com/uk/about_me.
3. https://rkeeper.com.ua/?gad_source=1#about-1 [Електронний ресурс] – Режим доступу до ресурсу: R-Keeper. URL.
4. Діаграма варіантів використання UML: підручник із прикладом [Електронний ресурс] – Режим доступу до ресурсу: <https://www.guru99.com/uk/use-case-diagrams-example.html>
5. Zosym M. Моделювання даних (Data Modelling) [Електронний ресурс] / Махум Zosym – Режим доступу до ресурсу: <https://www.maxzosim.com/data-modelling/>.
6. Web MVC framework [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html>.
7. Що таке діаграма компонентів в UML? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.guru99.com/uk/component-diagram-uml-example.html>.
8. Навіщо потрібна діаграма станів: розуміння роботи системи у світі [Електронний ресурс] – Режим доступу до ресурсу: <https://life.woodstar.com.ua/navishho-potribna-diagrama-staniv-rozuminnya-roboti-sistemi-u-sviti/>.
9. Що таке діаграма розгортання? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.guru99.com/uk/deployment-diagram-uml-example.html>.
10. IntelliJ IDEA [Електронний ресурс] – Режим доступу до ресурсу: <https://www.jetbrains.com/idea/>

11. What is Java technology and why do I need it? [Електронний ресурс] – Режим доступу до ресурсу: https://www.java.com/en/download/help/whatis_java.html.
12. Коротко про MariaDB [Електронний ресурс] – Режим доступу до ресурсу: <https://mariadb.org/uk/>.
13. Bringing MySQL to the web [Електронний ресурс] – Режим доступу до ресурсу: <https://www.phpmyadmin.net>.
14. Why Spring? [Електронний ресурс] – Режим доступу до ресурсу: <https://spring.io/why-spring>.
15. Spring Security [Електронний ресурс] – Режим доступу до ресурсу: <https://spring.io/projects/spring-security>.
16. Spring Data JPA [Електронний ресурс] – Режим доступу до ресурсу: <https://spring.io/projects/spring-data-jpa>.
17. Introduction [Електронний ресурс] – Режим доступу до ресурсу: <https://getbootstrap.com/docs/4.3/getting-started/introduction/>.
18. Закон України \"Про охорону праці\" [Електронний ресурс] – Режим доступу до ресурсу: https://zakononline.com.ua/documents/show/151241_591220.
19. Кодекс законів про працю України [Електронний ресурс] – Режим доступу до ресурсу: <https://zakon.rada.gov.ua/laws/show/322-08#Text>.
20. Про порядок розробки, побудови, викладення, оформлення, затвердження державних санітарних правил і норм, гігієнічних нормативів та методичних документів [Електронний ресурс] – Режим доступу до ресурсу: <https://zakon.rada.gov.ua/rada/show/v0011488-98#Text>.
21. Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги [Електронний ресурс] – Режим доступу до ресурсу: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=71028.

ДОДАТОК А – Технічне завдання

Вступ

Завданням даної дипломної роботи є розробка інформаційної системи управління рестораном. В таблиці 1 представлено загальні відомості про систему.

Таблиця 1 - Загальні відомості про систему

Повна назва системи	Інформаційна система для управління рестораном
Умовне позначення системи	ІС УР
Розроблювач	Козлов Олександр Юрійович
Замовник	Кафедра інформаційних управляючих систем та технологій

1. Підстави для розробки

ІС УР розробляється на підставі завдання для дипломного проектування за спеціальністю 126 – «Інформаційні системи та технології».

2. Призначення розробки

Об'єкт автоматизації: процеси комунікації між персоналом ресторану

Основними завданнями персоналу є:

- обслуговування клієнтів;
- формування замовлень;
- приготування замовлень;
- управління персоналом;
- менеджмент ресторану.

3. Мета створення системи

Основною метою створення ІС для управління рестораном є удосконалення та пришвидшення комунікацій на основі автоматизації виконуваних функцій, формування замовлень без паперової технології при його веденні, оперативне формування у базі даних інформації про

замовлення, меню, співробітників, наповненість залу, яка використовуються персоналом для управління рестораном.

4. Вимоги до програми

4.1 Вимоги до функціональних характеристик

ІС, що буде спроектована, повинна виконувати наступні завдання:

- здійснювати вхід користувача в систему після коректного введення даних входу;
- виконувати операції додавання, корегування, видалення та перегляду даних;
- виконувати наступні запити:
 - Перегляд даних про меню (назви блюд, їх опис, до якої категорії вони відносяться та їх ціна);
 - Створення замовлення;
 - Перегляд наповненості залу відвідувачами;
 - Перегляд зарплат співробітників;
 - Перегляд замовлень за вказаний період часу.
 - Зміна статусів замовлення.
 - Зміна статусу столика.
 - Підрахунок калорійності блюда.
 - Перегляд залишків продуктів на складі.

Для введення (редагування) даних потрібно створити діалогові форми, які дозволять користувачеві вводити (редагувати) інформацію. Для зручності роботи користувачів необхідно розробити відповідні інтерфейси.

4.2 Вимоги до надійності

При проектуванні повинні бути використані типові проектні рішення там, де це можливо. На стадії технічного проектування повинні прийматися рішення, що зменшують трудомісткість експлуатації системи.

4.3 Умови експлуатації

ІС розроблена на мові програмування Java в якості СУБД використано MariaDB.

4.4 Вимоги до інформаційної та програмної сумісності

Клієнтський комп'ютер повинен використовувати операційні системи (ОС) Microsoft Windows /7/8/10/11 чи Linux. Клієнтський телефон має використовувати операційну систему Android чи iOS.

Вимоги для роботи веб-додатку:

Наявність стабільного підключення до глобальної мережі «Інтернет» на робочому місці клієнта / співробітника / тестового користувача зі швидкістю не менше 256 Кбіт / сек (рекомендується 1 Мбіт / сек і більше).

Вимоги до сервера:

- Процесор з частотою 2.4 ГГц або більш потужний;
- Оперативна пам'ять 8GB, рекомендується 16Gb.
- Операційна система Linux.

4.5 Вимоги до технічного забезпечення

Технічні вимоги до клієнтських комп'ютерів визначаються вимогами операційної систем.

5. Вимоги до програмної документації

Усі регламентуючі документи є елементами ІС УР, які безпосередньо входять до цього. Регламент системи описує правила її ефективного використання. В таблиці 2 наведено склад документів.

Таблиця 2 - Склад документів організаційно-методичного та експлуатаційного забезпечення

№	Назва	Стадія
1	Опис організаційної структури ІПГІА	Технічне проектування
2	Керівництво користувача	Реалізація системи

6. Стадії та етапи розробки

В таблиці 3 представлено стадії та етапи по створенню ІС УР.

Таблиця 3 - Стадії та етапи робіт по створенню ІС УР

№	Стадія	Документи
1	Аналіз предметної області	Звіт з описом предметної області
2	Системне проектування	Технічне завдання
3	Ескізне проектування	Ескізний проект
4	Технічне проектування	Технічний проект
5	Робоче проектування	Робочий проект

ДОДАТОК Б – ПРОГРАМНИЙ КОД

Повний код: https://github.com/kozlov2777/Diplom_Spring

Order.java

```
package com.example.demo.models;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.time.LocalDateTime;
import java.util.Set;
import jakarta.persistence.*;

@Data
@Entity
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "orders")
public class Orders {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "table_id")
    private Tables table;
    private LocalDateTime createdAt;

    @ManyToMany
    @JoinTable(
        name = "order_items",
        joinColumns = @JoinColumn(name = "order_id"),
        inverseJoinColumns = @JoinColumn(name = "menu_item_id")
    )
    private Set<Menu_Items> items;

    @ManyToOne
    @JoinColumn(name = "status_id")
    private Statuses status;

    @ManyToOne
    @JoinColumn(name = "employee_id")
    private Employees employee;

    @OneToMany(mappedBy = "order", cascade = CascadeType.ALL)
    private Set<Order_Items> orderItems;
}
```

Menu_Items.java:

```
package com.example.demo.models;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
```

```

@Data
@Entity
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "menu_items")

public class Menu_Items {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String description;
    private Double price;

    @ManyToOne
    @JoinColumn(name = "category_id")
    private Categories category;
}

```

Order_Items.java:

```

package com.example.demo.models;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Entity
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "order_items")
public class Order_Items {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "order_id")
    private Orders order;

    @ManyToOne
    @JoinColumn(name = "menu_item_id")
    private Menu_Items item;
    private Integer quantity;
}

```

Ingredient_Items.java:

```

package com.example.demo.models;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Entity
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "ingredient_items")
public class Ingredient_Items {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)

```

```

        private Long id;

        @ManyToOne
        @JoinColumn(name = "menu_item_id")
        private Menu_Items item;

        @ManyToOne
        @JoinColumn(name = "ingredient_id")
        private Ingredients ingredient;
        private Integer quantity;
        private String unit;
    }

```

Employees.java:

```

package com.example.demo.models;

import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Entity
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "employees")
public class Employees {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String firstName;
    private String lastName;

    @ManyToOne
    @JoinColumn(name = "role_id")
    private Roles role;

    private String username;
    private String password;
}

```

MenuItemRepository.java:

```

package com.example.demo.repositories;

import com.example.demo.dto.ItemDetailDto;
import com.example.demo.dto.ItemsForCreatedTO;
import com.example.demo.dto.MenuItemDto;
import com.example.demo.dto.StatusesDto;
import com.example.demo.models.Menu_Items;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;
import java.util.List;

@Repository
public interface MenuItemRepository extends JpaRepository<Menu_Items, Long> {
    @Query("SELECT new com.example.demo.dto.MenuItemDto(m.id, m.name, m.description, m.price, m.category.name, " +
        "ROUND(SUM(i.calories * ii.quantity), 3)) " +
        "FROM Menu_Items m " +
        "JOIN Ingredient_Items ii ON ii.item.id = m.id " +
        "JOIN Ingredients i ON i.id = ii.ingredient.id " +
        "GROUP BY m.id, m.name, m.description, m.price, m.category.name")

```

```

List<MenuItemDto> getMenuItems();

@Query("SELECT new com.example.demo.dto.ItemDetailDto(mi.name,
ii.quantity, i.name, ii.unit, i.calories) " +
      "FROM Menu_Items mi " +
      "JOIN Ingredient_Items ii ON ii.item.id = mi.id " +
      "JOIN Ingredients i ON i.id = ii.ingredient.id " +
      "WHERE mi.id = ?1")
List<ItemDetailDto> getItemDetails(Long orderId);

@Query("SELECT new com.example.demo.dto.ItemsForCreatedTO(m.id, m.name,
m.description, m.price)FROM Menu_Items m ")
List<ItemsForCreatedTO> getMenuItemsForCreate();

@Query("SELECT new com.example.demo.dto.StatusesDto(s.id, s.name) FROM
Statuses s")
List<StatusesDto> getStatusesList();
}

```

OrderRepository.java:

```

package com.example.demo.repositories;

import com.example.demo.dto.OrderByStatusDto;
import com.example.demo.dto.OrderDetailDto;
import com.example.demo.dto.OrderDto;
import com.example.demo.dto.OrderDetailByTableDto;
import com.example.demo.models.Orders;
import jakarta.transaction.Transactional;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

import java.time.LocalDateTime;
import java.util.List;

@Repository
public interface OrderRepository extends JpaRepository<Orders, Long> {
    @Query("select new com.example.demo.dto.OrderDto(o.id, o.table.id,
o.createdAt, sum(oi.quantity*m.price)) from Orders o join Order_Items oi on
o.id = oi.order.id join Menu_Items m on oi.item.id=m.id group by o.id order
by o.createdAt desc ")
    List<OrderDto> getOrders();

    @Query("SELECT new com.example.demo.dto.OrderDetailDto(oi.id, m.name,
oi.quantity, m.price, o.createdAt, e.lastName, o.table.id, s.name) " +
      "FROM Order_Items oi " +
      "JOIN Menu_Items m ON oi.item.id = m.id " +
      "JOIN Orders o ON oi.order.id = o.id " +
      "JOIN Employees e ON o.employee.id = e.id " +
      "JOIN Statuses s ON o.status.id = s.id " +
      "WHERE oi.order.id = :orderId")
    List<OrderDetailDto> getOrderDetails(Long orderId);

    @Query("SELECT new com.example.demo.dto.OrderDetailByTableDto(o.id,
m.name, oi.quantity, o.createdAt, SUM(m.price * oi.quantity)) " +
      "FROM Orders o " +
      "JOIN Order_Items oi ON o.id = oi.order.id " +
      "JOIN Menu_Items m ON oi.item.id = m.id " +
      "JOIN Tables t ON t.id = o.table.id " +
      "WHERE t.id = :tableNumber " +
      "GROUP BY o.id, m.id " +

```

```

        "ORDER BY o.createdAt DESC")
    List<OrderDetailByTableDto> getOrderDetailsByTableNumber(Long
tableNumber);

    @Query("SELECT new com.example.demo.dto.OrderByStatusDto(o.id,
o.table.id, m.name, oi.quantity, o.createdAt, SUM(m.price * oi.quantity),
s.name, e.firstName) " +
        "FROM Orders o " +
        "JOIN Order_Items oi ON o.id = oi.order.id " +
        "JOIN Menu_Items m ON oi.item.id = m.id " +
        "JOIN Employees e ON o.employee.id = e.id " +
        "JOIN Statuses s ON o.status.id = s.id " +
        "WHERE o.status.id = :statusId " +
        "GROUP BY o.id, m.id " +
        "ORDER BY o.createdAt DESC")
    List<OrderByStatusDto> getOrdersByStatus(Long statusId);

    @Transactional
    @Modifying
    @Query("UPDATE Orders o SET o.status.id = ?1 WHERE o.id = ?2")
    void updateOrderStatus(Long status_id, Long orderId);

    @Transactional
    @Modifying
    @Query("UPDATE Tables t SET t.tableStatus.id = ?1 WHERE t.id = (SELECT
o.table.id FROM Orders o WHERE o.id = ?2)")
    void updateTableStatus(Long table_status_id, Long orderId);

    @Query("SELECT new com.example.demo.dto.OrderDto(o.id, o.table.id,
o.createdAt, sum(oi.quantity*m.price)) from Orders o join Order_Items oi on
o.id = oi.order.id join Menu_Items m on oi.item.id=m.id where o.createdAt
between :startDate and :endDate group by o.id order by o.createdAt desc ")
    List<OrderDto> getOrdersByDate(LocalDateTime startDate, LocalDateTime
endDate);

    @Transactional
    @Modifying
    @Query("UPDATE Ingredients i " +
        "SET i.quantity = i.quantity - ((SELECT MAX(ii.quantity) " +
        "FROM Menu_Items m " +
        "JOIN Ingredient_Items ii ON m.id
= ii.item.id " +
        "WHERE m.id = ?1) * ?2) " +
        "WHERE i.id IN (SELECT ii.ingredient.id " +
        "FROM Menu_Items m " +
        "JOIN Ingredient_Items ii ON m.id = ii.item.id " +
        "WHERE m.id = ?1)")
    void updateQuantityInIngredient(Long id, int count);
}

```

OrderByStatusDTO.java:

```

package com.example.demo.dto;

import lombok.AllArgsConstructor;
import lombok.Data;
import java.time.LocalDateTime;

@Data
@AllArgsConstructor
public class OrderByStatusDto {

```



```

        private Long id;
        private Long tableNumber;
        private String name;
        private int quantity;
        private LocalDateTime createdAt;
        private double total;
        private String status;
        private String employeeFirstName;
    }

```

ItemDetailDTO.java:

```

package com.example.demo.dto;

import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class ItemDetailDto {
    private String menuItemName;
    private int quantity;
    private String ingredientName;
    private String unit;
    private double calories;
}

```

CreateOrderDTO.java:

```

package com.example.demo.dto;

import lombok.AllArgsConstructor;
import lombok.Data;
import java.time.LocalDateTime;
import java.util.List;

@AllArgsConstructor
@Data
public class CreateOrderDTO {
    private Long tableId;
    private LocalDateTime createdAt;
    private Long employeeId;
    private Long statusId;
    private List<Long> itemIds;
    private List<Integer> quantities;
}

```

OrderService.java:

```

package com.example.demo.services;

import com.example.demo.dto.OrderByStatusDto;
import com.example.demo.dto.OrderDetailByTableDto;
import com.example.demo.dto.OrderDetailDto;
import com.example.demo.dto.OrderDto;

import java.time.LocalDateTime;
import java.util.List;
import com.example.demo.models.Orders;
import com.example.demo.repositories.OrderRepository;
import org.springframework.stereotype.Service;

```

```

@Service
public class OrderService {

    private final OrderRepository orderRepository;

    public OrderService(OrderRepository orderRepository) {
        this.orderRepository = orderRepository;
    }

    public List<OrderDto> getOrders() {
        return orderRepository.getOrders();
    }

    public List<OrderDetailDto> getOrderById(Long id){
        return orderRepository.getOrderDetails(id);
    }

    public List<OrderDetailByTableDto> getOrderDetailsByTableNumber(Long
tableNumber) {
        return orderRepository.getOrderDetailsByTableNumber(tableNumber);
    }

    public List<OrderByStatusDto> getOrdersByStatus(long l) {
        return orderRepository.getOrdersByStatus(l);
    }

    public void updateOrderStatus(Long status_id, Long orderId) {
        orderRepository.updateOrderStatus(status_id, orderId);
    }

    public void updateTableStatus(Long table_status_id, Long orderId) {
        orderRepository.updateTableStatus(table_status_id, orderId);
    }

    public void save(Orders orders){
        orderRepository.save(orders);
    }

    public List<OrderDto> getOrdersByDate(LocalDateTime startDate,
LocalDateTime endDate) {
        return orderRepository.getOrdersByDate(startDate, endDate);
    }

    public void updateIngredientsAfterOrderCreation(Long id, int count) {
        orderRepository.updateQuantityInIngredient(id, count);
    }
}

```

OrderController.java:

```

package com.example.demo.controllers;

import com.example.demo.dto.OrderByStatusDto;
import com.example.demo.dto.OrderDetailByTableDto;
import com.example.demo.dto.OrderDetailDto;
import com.example.demo.dto.OrderDto;
import com.example.demo.models.*;
import com.example.demo.repositories.*;
import com.example.demo.services.OrderService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import java.time.LocalDateTime;

```

```

import java.util.HashSet;
import java.util.List;
import java.util.Set;

@Controller
public class OrderController {

    private final OrderService orderService;
    private final TableRepository tableRepository;
    private final EmployeeRepository employeeRepository;
    private final StatusRepository statusRepository;
    private final MenuItemRepository menuItemRepository;
    private final OrderItemRepository orderItemRepository;
    @Autowired
    public OrderController(OrderService orderService, TableRepository
tableRepository, EmployeeRepository employeeRepository, StatusRepository
statusRepository, MenuItemRepository menuItemRepository, OrderItemRepository
orderItemRepository) {
        this.orderService = orderService;
        this.tableRepository = tableRepository;
        this.employeeRepository = employeeRepository;
        this.statusRepository = statusRepository;
        this.menuItemRepository = menuItemRepository;
        this.orderItemRepository = orderItemRepository;
    }

    @GetMapping("/orders_view")
    public String viewOrders(Model model) {
        List<OrderDto> orderList = orderService.getOrders();
        model.addAttribute("orderList", orderList);
        return "order_list";
    }

    @GetMapping("/order_detail/{order_id}")
    public String viewOrderDetail(@PathVariable("order_id") Long orderId,
Model model) {
        List<OrderDetailDto> orderDetailList =
orderService.getOrderById(orderId);
        model.addAttribute("orderDetailList", orderDetailList);
        return "order_detail";
    }

    @GetMapping("/order_detail_by_table_number/{tableNumber}")
    public String getOrderDetailByTableNumber(@PathVariable Long tableNumber,
Model model) {
        List<OrderDetailByTableDto> detailByTable =
orderService.getOrderDetailsByTableNumber(tableNumber);
        model.addAttribute("detailByTable", detailByTable);
        return "order_detail_by_table_number";
    }

    @GetMapping("/")
    public String getOrdersByStatus(Model model) {
        List<OrderByStatusDto> orderStatus1 =
orderService.getOrdersByStatus(1L);
        List<OrderByStatusDto> orderStatus2 =
orderService.getOrdersByStatus(2L);

        model.addAttribute("orderStatus1", orderStatus1);
        model.addAttribute("orderStatus2", orderStatus2);

        return "order_by_status";
    }
}

```

```

@RequestMapping("/update_order_status/{order_id}")
public String updateOrderStatus(@PathVariable Long order_id) {
    orderService.updateOrderStatus(2L, order_id);
    return "redirect:/";
}

@RequestMapping("/update_order_status_and_table/{order_id}")
public String updateOrderStatusAndTable(@PathVariable Long order_id) {
    orderService.updateOrderStatus(3L, order_id);
    orderService.updateTableStatus(1L, order_id);
    return "redirect:/";
}

@RequestMapping("/new_order")
public String createNewOrder(@RequestParam("table_id_id") Long tableId,
                             @RequestParam("employee_id") Long
employeeId,
                             @RequestParam("status_id") Long statusId,
                             @RequestParam("item[]") List<Long> itemIds,
                             @RequestParam("quantity[]") List<Integer>
quantities) {
    Tables table = tableRepository.findById(tableId).orElse(null);
    Employees employee =
employeeRepository.findById(employeeId).orElse(null);
    Statuses status = statusRepository.findById(statusId).orElse(null);

    if (table != null && employee != null && status != null) {
        Orders order = new Orders();
        order.setTable(table);
        order.setEmployee(employee);
        order.setStatus(status);
        order.setCreatedAt(LocalDate.now());

        orderService.save(order);
        orderService.updateTableStatus(2L, order.getId());

        for (int i = 0; i < itemIds.size(); i++) {
            Long itemId = itemIds.get(i);
            int quantity = quantities.get(i);
            Menu_Items menuItem =
menuItemRepository.findById(itemId).orElse(null);

            if (menuItem != null) {
                Order_Items orderItems = new Order_Items();
                orderItems.setOrder(order);
                orderItems.setItem(menuItem);
                orderItems.setQuantity(quantity);
                orderItemRepository.save(orderItems);

                orderService.updateIngredientsAfterOrderCreation(menuItem.getId(), quantity);
            }
        }

        return "redirect:/";
    } else {
        return "redirect:/new_order";
    }
}

@GetMapping("/orders_by_date")
public String getOrdersByDate(@RequestParam(name = "start_date", required
= false) String startDateStr,

```

```

                                @RequestParam(name = "end_date", required =
false) String endDateStr,
                                Model model) {
    if (startDateStr != null && endDateStr != null) {
        LocalDateTime startDate = LocalDateTime.parse(startDateStr +
"T00:00:00");
        LocalDateTime endDate = LocalDateTime.parse(endDateStr +
"T23:59:59");

        List<OrderDto> ordersByDate =
orderService.getOrdersByDate(startDate, endDate);
        model.addAttribute("ordersByDate", ordersByDate);
    } else {
        return "orders_by_date";
    }
    return "orders_by_date";
}
}

```

TableController.java:

```

package com.example.demo.controllers;

import com.example.demo.dto.TableStatusDto;
import com.example.demo.services.TableService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import java.util.List;

@Controller
public class TableController {
    private final TableService tableService;
    @Autowired
    public TableController(TableService tableService) {
        this.tableService = tableService;
    }

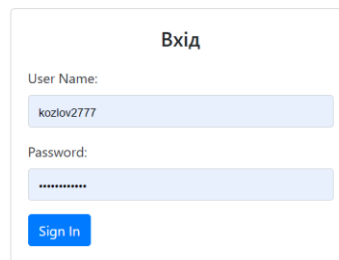
    @GetMapping("/tables_status")
    public String getTablesStatus(Model model) {
        List<TableStatusDto> tables = tableService.getTableStatusDTO();
        model.addAttribute("tables", tables);
        return "table_status";
    }
}

```

ДОДАТОК В – ІНСТРУКЦІЯ КОРИСТУВАЧА

Необхідно перейти на сторінку додатку, в першу чергу користувач побачить вікно для авторизації.

1. Потрібно ввести ім'я користувача
2. Потрібно ввести пароль



Вхід

User Name:

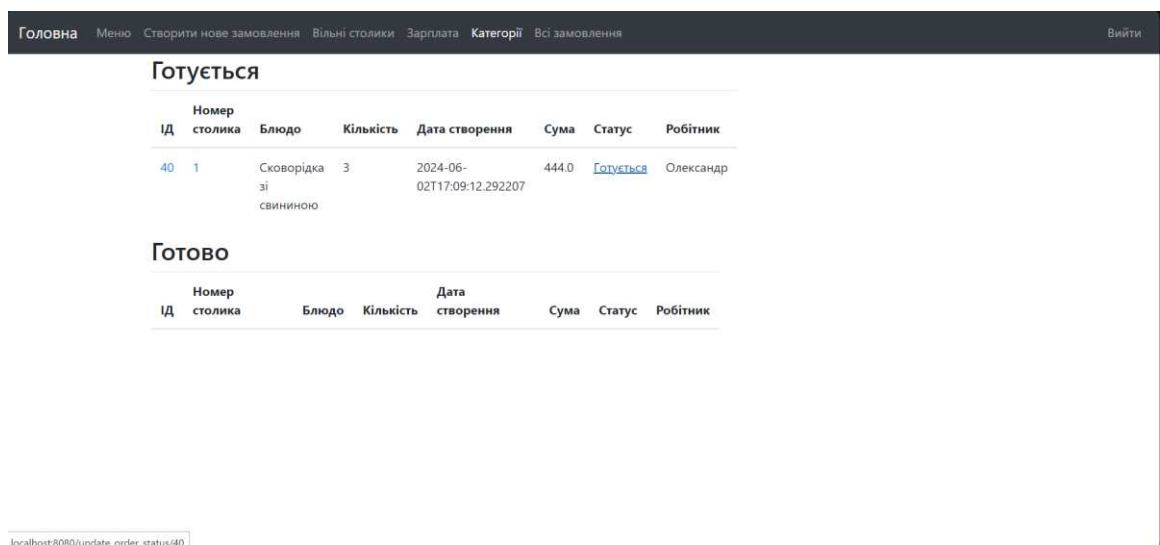
kozlov2777

Password:

Sign In

Рисунок А1 – Форма для авторизації

Для зміни статусу замовлення, на головній сторінці, потрібно натиснути на статус цього замовлення.



Головна	Меню	Створити нове замовлення	Вільні столики	Зарплата	Категорії	Всі замовлення	Вийти
---------	------	--------------------------	----------------	----------	-----------	----------------	-------

Готується

Номер ІД	столик	Блюдо	Кількість	Дата створення	Сума	Статус	Робітник
40	1	Сковорідка зі свининою	3	2024-06-02T17:09:12.292207	444.0	Готується	Олександр

Готово

Номер ІД	столик	Блюдо	Кількість	Дата створення	Сума	Статус	Робітник
----------	--------	-------	-----------	----------------	------	--------	----------

localhost8080/update_order_status/40

Рисунок А2 – Замовлення в статусі «Готується»

Головна	Меню	Створити нове замовлення	Вільні столики	Зарплата	Категорії	Всі замовлення	Вийти
Готується							
Номер ІД	Номер столика	Блюдо	Кількість	Дата створення	Сума	Статус	Робітник
Готово							
40	1	Сковорідка зі свининою	3	2024-06- 02T17:09:12.292207	444.0	Готово	Олександр

Рисунок А3 – Замовлення перейшло в статус «Готово»

Для перегляду меню, потрібно натиснути на кнопку «Меню» .

Головна	Меню	Створити нове замовлення	Вільні столики	Зарплата	Категорії	Всі замовлення	Вийти
Меню							
Айди	Назва	Опис	Ціна	Категорія	Калорійність		
3	Сковорідка зі свининою	(картопля, гриби, свинина, морква, фасоль, томатна паста, цибуля)	148.0	Гарячі страви	285.6		
4	Пюре з ковбасками гриль	(картопля, ковбаски-гриль)	168.0	Гарячі страви	151.5		
5	Мохіто	(спрайт, лайм, лід, м'ята)	35.0	Напої	58.3		
6	Зелений чай	Чай "Ліптон"	15.0	Напої	0.06		
7	Філадельфія	(лосось, рис, сир, огірок)	160.0	Роли	136.7		
8	Каліфорнія	(лосось, сир, авокадо, огірок)	200.0	Роли	112.8		
9	Цезарь	(курка, листя салату, яйця, соус, помідори черрі, пармезан)	150.0	Салати та закуски	155.48		
10	Салат з тунцем	(тунець, листя салату, яйця, картопля, соус)	135.0	Салати та закуски	62.78		
11	Пюре	(картопля)	65.0	Гарніри	22.5		

Рисунок А4 – Сторінка меню

Для перегляду інформації про склад та калорійність блюда потрібно натиснути на калорійність цього блюда.

Головна	Меню	Створити нове замовлення	Вільні столики	Зарплата	Категорії	Всі замовлення	Вийти
---------	------	--------------------------	----------------	----------	-----------	----------------	-------

Деталі

Блюдо	Кількість в гр або мл	Інгредієнт	О.в.	Калорій в 1 гр
Сковорідка зі свининою	300	Свинина	грам	0.87
Сковорідка зі свининою	80	Картопля	грам	0.09
Сковорідка зі свининою	15	Морква	грам	0.06
Сковорідка зі свининою	50	Гриби	грам	0.11
Сковорідка зі свининою	50	Томатна заправка	мл	0.22

Рисунок А5 – Перегляд інформації про блюдо

Для створення замовлення, потрібно виконати наступні кроки:

- 1) Натиснути на кнопку «Створити нове замовлення»
- 2) Обрати номер столика
- 3) Обрати офіціанта хто прийняв замовлення
- 4) Обрати статус замовлення
- 5) Обрати блюда та їх кількість
- 6) Натиснути на кнопку «Зберегти»

Головна	Меню	Створити нове замовлення	Вільні столики	Зарплата	Категорії	Всі замовлення	Вийти
---------	------	--------------------------	----------------	----------	-----------	----------------	-------

Створити нове замовлення

Номер столика:

2

Офіціант:

Козлов

Статус:

Готується

Блюда:

Сковорідка зі свининою 1

Блюда:

Мохіто 1

Добавити страву

Зберегти

Рисунок А6 – Створення замовлення

Для перегляду наповненості ресторану потрібно натиснути на кнопку «вільні столики», яка знаходиться в навігаційному меню.

Головна	Меню	Створити нове замовлення	Вільні столики	Зарплата	Категорії	Всі замовлення	Вийти
Столики							
Ід	Статус						
1	Занято						
2	Вільно						
3	Вільно						

Рисунок А7 – Сторінка наповненості ресторану

Для перегляду зарплат співробітників потрібно перейти на сторінку «Зарплата» використовуючи кнопку в навігаційному меню.

Головна	Меню	Створити нове замовлення	Вільні столики	Зарплата	Категорії	Всі замовлення	Вийти
Зарплати							
Ім'я	Фамілія	Зарплата					
Олександр	Козлов	12390.0					
Іван	Іванов	12810.0					

Рисунок А8 – Сторінка перегляду зарплат персоналу

Для перегляду замовлень за вказаний період часу потрібно:

1. Перейти на сторінку Всі замовлення
2. Вказати початкову дату

3. Вказати кінцеву дату
4. Натиснути кнопку «Переглянути»

Головна

Меню

Створити нове замовлення

Вільні столики

Зарплата

Категорії

Всі замовлення

Вийти

Список заказів

Початкова дата

2023-11-12

Кінцева дата

2023-09-18

Переглянути

Айді	Номер столика	Дата створення	Загальна сума
2	1	2023-11-12 16:33:23	750.0
1	1	2023-09-18 16:56:01	740.0

Рисунок А9 – Вибір замовлень у вказаному часовому проміжку

Для реєстрації нового користувача системи адміністратор має заповнити наступну форму.

Register Employee

First Name:

Last Name:

Username:

Password:

Role ID:

Register

Рисунок А10 – реєстрація нового користувача

Якщо користувач не має прав до деякого функціоналу і отримує сторінку попередження він має натиснути на «Головну сторінку».

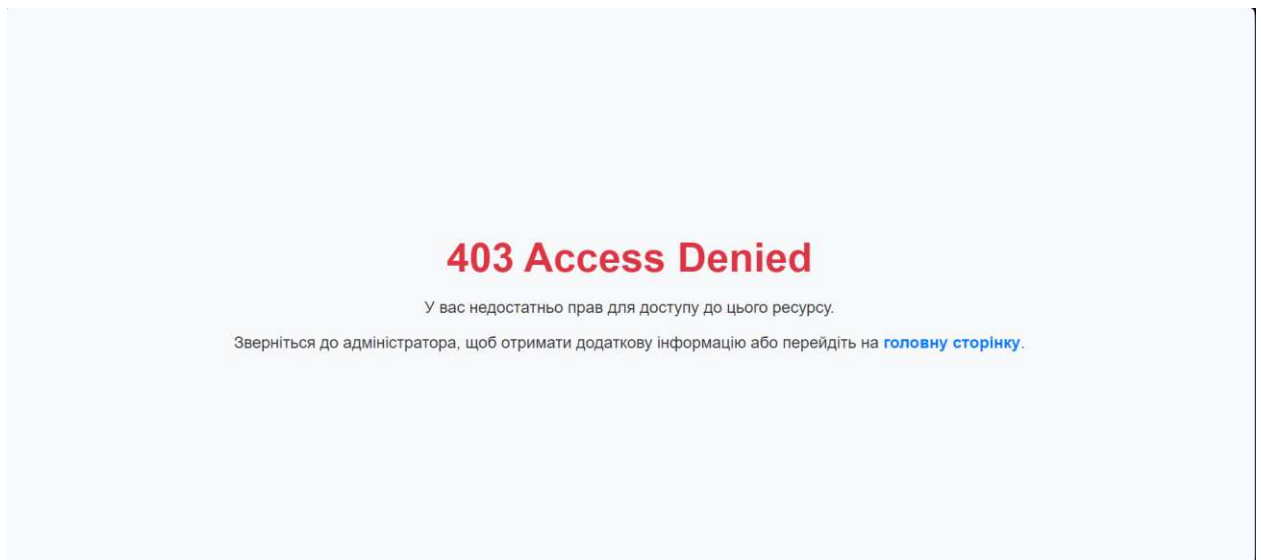


Рисунок А11 – Вихід зі сторінки попередження

Для виходу із системи потрібно натиснути на кнопку «Вийти», після чого користувача буде направлено на сторінку авторизації.