МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:
Завідувач кафедри комп'ютерних технологій та економічної кібернетики
д. е. н., проф. П. М. Грицюк
_____
«____» _____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня «магістр»
за освітньо-професійною програмою
«Інформаційні технології в бізнесі»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Автоматизація обліку та управління виконанням замовлень закладу громадського харчування»

Виконав:
здобувач вищої освіти 2 курсу, групи ІТБ-61м
Градовий О.О.
(прізвище, ім'я, по-батькові)
Керівник:
к.т.н, доцент Барановський С.В.
(науковий ступінь, вчене звання прізвище та ініціали)
Рецензент:
к.т.н, доцент Василів В.Б.
(науковий ступінь, вчене звання прізвище та ініціали)

Рівне 2024

РЕФЕРАТ

Кваліфікаційна робота магіста: 51 с., 10 рис., 1 табл., 22 літературних джерел

Актуальність теми даної мгістерської роботи полягає в тому, що використовуються сучасних інформаційних технологіях, таких як Java, фраємоворк Spring і базу даних MySQL. Ця система спрямована на автоматизацію процесів управління завданнями, аналізу даних та інтеграції з іншими системами та сервісами.

Об'єкт дослідження – процеси автоматизації управління закладом громадського харчування, а предметом – технології та методи розробки програмного забезпечення для цих процесів.

Предмет дослідження – методи автоматизації, моніторингу, аналізу й управління завданням за допогою веб-сайту.

Мета роботи – розробка інформаційної системи управління кафе, яка забезпечуватиме автоматизацію ключових бізнес-процесів, включаючи адміністрування меню, обробку замовлень, управління користувачами та аналіз статистичних даних

У магістерській роботі було: розроблено інформаційну систему управління кафе, яка забезпечує автоматизацію ключових бізнес-процесів, включаючи адміністрування меню, обробку замовлень, управління користувачами та аналіз статистичних даних. Досягнуто мети дослідження шляхом створення системи, що відповідає сучасним вимогам надійності, безпеки та продуктивності. Використання сучасних інструментів і технологій, таких як Java, Spring та MySQL, дозволило реалізувати ефективне, масштабоване та зручне рішення.

КЛЮЧОВІ СЛОВА: JAVA, SPRING, MYSQL, ВЕБ-САЙТ, ORM, JPA, HIBERNATE, JVM, SOLID, SPA, ВЕБ-ДОДАТОК

Зміст

ВСТУП

Актуальність теми визначається сучасними тенденціями цифровізації бізнесу, зокрема у сфері громадського харчування, де автоматизація є ключовим фактором підвищення ефективності управління та якості обслуговування клієнтів. Успішне функціонування кафе значною мірою залежить від швидкості обробки замовлень, точності управління меню та доступності аналітичних даних для прийняття управлінських рішень. Запровадження програмного забезпечення для автоматизації цих процесів дозволяє мінімізувати ручну працю, зменшити ризики помилок та покращити користувацький досвід клієнтів.

Метою роботи є розробка інформаційної системи управління кафе, яка забезпечуватиме автоматизацію ключових бізнес-процесів, включаючи адміністрування меню, обробку замовлень, управління користувачами та аналіз статистичних даних. Система має бути побудована на основі сучасних підходів до програмної інженерії та відповідати високим стандартам безпеки, продуктивності та масштабованості.

Об’єктом дослідження є процеси автоматизації управління закладом громадського харчування, а предметом – технології та методи розробки програмного забезпечення для цих процесів. Особливу увагу приділено використанню сучасних фреймворків і баз даних для створення інтегрованого рішення, яке дозволить оптимізувати діяльність кафе та підвищити його конкурентоспроможність.

Практичне значення роботи полягає у створенні ефективного програмного продукту, який надаватиме адміністраторам кафе інструменти для управління меню, контролю замовлень, перегляду інформації про користувачів та аналізу фінансових показників. Система стане надійним інструментом для зменшення витрат часу на виконання рутинних завдань і покращення якості обслуговування клієнтів.

Очікуваними результатами є розробка та впровадження стабільної інформаційної системи, що відповідатиме функціональним вимогам закладу громадського харчування. Це забезпечить підвищення продуктивності праці, оптимізацію операційних процесів та покращення взаємодії з клієнтами, що сприятиме досягненню стратегічних цілей закладу.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ

1.1. Діяльність закладів громадського харчування

Сфера громадського харчування включає підприємства, установи та компанії, які готують їжу поза домом.[1] Він включає ресторани, продуктові магазини, шкільні та лікарняні їдальні, підприємства громадського харчування та багато інших форматів.[1]

Постачальники для операторів громадського харчування — це розповсюджувачі громадського харчування, які постачають дрібні вироби (кухонне начиння) та продукти харчування. Деякі компанії виробляють продукцію як для споживачів, так і для громадського харчування. Споживча версія зазвичай поставляється в упаковках індивідуального розміру з продуманим дизайном етикетки для роздрібного продажу. Версія для громадського харчування упакована в набагато більший промисловий розмір і часто не має барвистого дизайну етикеток споживчої версії.

У 2010 році система харчування, включаючи послуги громадського харчування та роздрібну торгівлю продуктами харчування, поставила продовольства в США на суму 1,24 трильйона доларів США, 594 мільярди доларів з яких було поставлено підприємствами громадського харчування, визначеними Міністерством сільського господарства США як будь-які місця, де готують їжу для негайного споживання на місці, включаючи місцях, які в основному не займаються роздачею їжі, як-от рекреаційні заклади та роздрібні магазини.[2] У 2010 році на ресторани з повним набором послуг і ресторани швидкого харчування припадає 77% усіх продажів громадського харчування, при цьому на ресторани з повним набором послуг припадає трохи більше, ніж на ресторани швидкого харчування в 2010 році.[2] Зміщення ринкових часток між фаст-фудом і ресторанами з повним набором послуг відповідно до ринкового попиту змінює пропозиції їжі та послуг обох типів ресторанів.[2]

За даними Національної асоціації ресторанів, зростаюча тенденція серед споживачів у сфері громадського харчування серед споживачів США – це глобальна кухня: у 2015 році 66% споживачів у США споживали більше їжі, ніж у 2010 році, 80% споживачів споживали страви «етнічної» кухні принаймні раз на місяць, а 29% пробували нову «етнічну» кухню протягом останнього року.[3][4]

Станом на 2015 рік обсяг ринку дистриб'юторів Foodservice у США становив 231 мільярд доларів; національний ринок широкого зв'язку контролюється компаніями US Foods і Sysco, які разом мають 60-70% частки ринку, і FTC заблокувала їх об'єднання з міркувань ринкової влади.[5]

Їжа громадського харчування, як правило, має в середньому більше калорій і менше основних поживних речовин, ніж їжа, приготована вдома.[6] Багато ресторанів, у тому числі закладів швидкого харчування, додали більше салатів і фруктових пропозицій і або за бажанням, або у відповідь на місцеве законодавство забезпечили маркування харчової цінності.[6]

У США FDA працює над створенням єдиних інструкцій щодо маркування фаст-фуду та ресторанів, його запропоновані правила були опубліковані в 2011 році, а остаточні правила опубліковані 1 грудня 2014 року, які замінюють державні та місцеві положення щодо маркування меню та набувають чинності 1 грудня 2015 року. [6][7] Дослідження показали, що нові етикетки можуть вплинути на вибір споживачів, але насамперед якщо вони надають неочікувану інформацію та що споживачі, які піклуються про своє здоров'я, стійкі до зміни поведінки на основі маркування меню [7]. ERS очікує, що ресторани швидкого харчування будуть працювати краще за умов нове маркування меню, ніж ресторани з повним набором послуг, оскільки ресторани з повним набором послуг, як правило, пропонують набагато калорійнішу їжу, причому 50% страв швидкого харчування становить від 400 до 800 калорій і менше 20% понад 1000 калорій, на відміну від ресторанів із повним набором послуг, 20% страв містить

понад 1400 калорій.[7] Коли споживачі знають про кількість калорій у ресторанах із повним набором послуг, 20% вибирають менш калорійні варіанти, і споживачі також зменшують споживання калорій протягом решти дня.[7]

Їжа один раз поза домом щотижня означає 2 зайві кілограми щороку або щоденне збільшення на 134 калорії та зниження якості дієти на 2 бали за Індексом здорового харчування.[8]

Крім того; ймовірність зараження харчовими захворюваннями (такими як черевний тиф і гепатит В або захворювання, спричинені кишковою паличкою, H. pylori, лістерією, сальмонелою та норовірусом) значно підвищується через те, що їжа не зберігається при температурі нижче 40 °F (4 °C) або готувати до температури вище 160 °F (71 °C), не мити руки протягом принаймні 20 секунд для людей, які займаються обробкою їжі, або ні миття забруднених обробних дощок та інших кухонних інструментів у гарячій воді.

1.2. Автоматизиція обліку та управління виконанням замовлень

Планування ресурсів підприємства (ERP) — це інтегроване управління основними бізнес-процесами, часто в режимі реального часу за допомогою програмного забезпечення та технологій. ERP зазвичай називають категорією програмного забезпечення для управління бізнесом, як правило, набору інтегрованих програм, які організація може використовувати для збору, зберігання, керування та інтерпретації даних багатьох видів діяльності. Системи ERP можуть бути локальними або хмарними. Хмарні додатки зросли за останні роки завдяки підвищенню ефективності, пов’язаному з доступністю інформації з будь-якого місця з доступом до Інтернету.

ERP надає інтегроване та постійно оновлюване уявлення про основні бізнес-процеси за допомогою загальних баз даних, які підтримуються системою керування базами даних. Системи ERP відстежують бізнес-ресурси — готівку, сировину, виробничі потужності — і статус бізнес-зобов’язань: замовлення, замовлення на купівлю та нарахування заробітної плати. Програми, які

складають систему, обмінюються даними між різними відділами (виробництво, закупівлі, продажі, бухгалтерія тощо), які надають дані.[1] ERP полегшує потік інформації між усіма бізнес-функціями та керує зв'язками із зовнішніми зацікавленими сторонами.[2]

За даними Gartner, розмір світового ринку ERP оцінюється в 35 мільярдів доларів у 2021 році.[3][4] Хоча перші системи ERP були зосереджені на великих підприємствах, менші підприємства все частіше використовують системи ERP.[5]

Система ERP об'єднує різноманітні організаційні системи та сприяє безпомилковим транзакціям і виробництву, тим самим підвищуючи ефективність організації. Однак розробка системи ERP відрізняється від розробки традиційної системи.[6] Системи ERP працюють на різноманітному комп'ютерному обладнанні та мережевих конфігураціях, зазвичай використовують базу даних як сховище інформації.[7]

Група Gartner вперше використала абревіатуру ERP у 1990-х роках [8] [9], щоб включити можливості планування потреб у матеріалах (MRP), а пізніше — планування виробничих ресурсів (MRP II), [10] [11], а також комп'ютер. - інтегроване виробництво. Не замінюючи ці терміни, ERP стала представляти більшу цілісність, яка відображає еволюцію інтеграції додатків за межі виробництва.[12]

Не всі пакети ERP розробляються на основі виробничого ядра; Постачальники ERP почали складати свої пакети з компонентами фінансів і бухгалтерського обліку, технічного обслуговування та людських ресурсів. До середини 1990-х років системи ERP охоплювали всі основні функції підприємства. Уряди та некомерційні організації також почали використовувати системи ERP.[13] «Методологія вибору системи ERP» — це офіційний процес вибору системи планування ресурсів підприємства (ERP). Існуючі методології

включають: метод воронки Койпера, тривимірний (3D) веб-інструмент підтримки прийняття рішень Добрина та методологію Кларкстона Потомака.[14]

Системи ERP пережили стрімке зростання в 1990-х роках. Через проблему 2000 року багато компаній скористалися можливістю замінити свої старі системи на ERP.[15]

Системи ERP спочатку були зосереджені на автоматизації функцій бек-офісу, які безпосередньо не впливали на клієнтів і громадськість. Фронт-офісні функції, такі як управління взаємовідносинами з клієнтами (CRM), мають справу безпосередньо з клієнтами, або системи електронного бізнесу, такі як електронна комерція та електронний уряд, або управління взаємовідносинами з постачальниками (SRM) стали інтегрованими пізніше, коли Інтернет спростив спілкування з зовнішні сторони.[16]

«ERP II» був створений у 2000 році в статті Gartner Publications під назвою ERP мертвий — хай живе ERP II.[17][18] Він описує веб-програмне забезпечення, яке надає співробітникам і партнерам (наприклад, постачальникам і клієнтам) доступ до систем ERP у реальному часі. Роль ERP II розширює традиційну оптимізацію ресурсів ERP і обробку транзакцій. Замість того, щоб просто керувати купівлею, продажем тощо, ERP II використовує інформацію в ресурсах, якими керує, щоб допомогти підприємству співпрацювати з іншими підприємствами.[19] ERP II є більш гнучким, ніж ERP першого покоління. Замість того, щоб обмежувати можливості системи ERP всередині організації, вона виходить за межі корпоративних стін, щоб взаємодіяти з іншими системами. Набір корпоративних програм є альтернативною назвою для таких систем. Системи ERP II зазвичай використовуються для забезпечення спільних ініціатив, таких як управління ланцюгом поставок (SCM), управління взаємовідносинами з клієнтами (CRM) і бізнес-аналітика (BI) між організаціями-діловими партнерами за допомогою різноманітних електронних бізнес-технологій.[20][21] Велика

частка компаній переслідує сильні управлінські цілі в системі ERP замість того, щоб придбати ERP-компанію.[22]

Тепер розробники докладають більше зусиль для інтеграції мобільних пристроїв із системою ERP. Постачальники ERP розширюють ERP на ці пристрої разом з іншими бізнес-додатками, щоб підприємствам не доводилося покладатися на сторонні програми.[23]

1.3. Огляд існуючих рішень

Сучасні інформаційні технології відіграють ключову роль в автоматизації бізнес-процесів підприємств. Впровадження інформаційних систем дозволяє оптимізувати роботу підприємств, підвищити ефективність управління та знизити витрати. Такі системи не лише забезпечують оперативне управління бізнесом, але й сприяють прийняттю рішень на основі аналізу даних.

Сучасний ринок інформаційних технологій пропонує різні типи програмного забезпечення для автоматизації облікових та управлінських процесів на підприємствах. Однією з основних категорій є комп'ютерні інформаційні системи (КІС), які забезпечують автоматизацію управління всіма етапами технологічного процесу, включаючи обробку інформації. Такі системи, як "1С: Підприємство" та ERP-системи (Enterprise Resource Planning), дозволяють підприємствам інтегрувати різні бізнес-процеси в єдину систему для ефективного управління ресурсами.

ERP-системи надають можливість гнучко налаштовувати робочі процеси, що дозволяє підвищити продуктивність та знизити адміністративні витрати. Наприклад, "1С: Підприємство" або зарубіжні SAP R/3 та MS AXAPTA дозволяють автоматизувати всі види господарської діяльності, що важливо для підприємств з великою кількістю підрозділів або філій.

Об'єктно-орієнтовані системи керування базами даних забезпечують багатокористувацький доступ до інформації та її надійне зберігання і обробку. Серед найбільш популярних рішень можна виділити продукти MySQL,

PostgreSQL, Oracle та Microsoft SQL Server. Вони спрощують обробку інформації та покращують аналітичні можливості системи.

Програмні продукти для бізнес-процесів дозволяють підприємствам ефективно управляти своїми процесами. Впровадження таких систем забезпечує інтеграцію процесів, таких як маркетинг, продажі, фінанси та управління персоналом. CRM-системи (Customer Relationship Management) дозволяють зосередитися на взаємодії з клієнтами та покращенні обслуговування.

Також існує програмне забезпечення класу DocFlow і WorkFlow для управління документами та потоками робіт. Ці системи дозволяють автоматизувати документообіг, що зменшує обсяги рутинної роботи та підвищує продуктивність [5].

Системи підтримки прийняття рішень надають можливість приймати зважені рішення на основі аналізу даних. Основними компонентами таких систем є база даних, база моделей та програмна підсистема. Ці компоненти забезпечують зберігання та обробку даних, створення моделей для моделювання процесів та оцінки результатів.

Системи підтримки прийняття рішень використовують системи керування базами даних, які забезпечують обробку даних з різних джерел. Важливою характеристикою систем підтримки прийняття рішень є здатність забезпечувати координацію рішень на різних рівнях управління, що корисно для великих підприємств. Інтерфейси для взаємодії з користувачем, такі як візуальні або голосові, роблять роботу з системою зручнішою та підвищують продуктивність.

Популярними рішеннями для автоматизації є “1С: Підприємство”, SAP та MS АХАРТА. Ці системи забезпечують автоматизацію не лише облікових, але й управлінських процесів, що дозволяє ефективно управляти ресурсами підприємства. Наприклад, “1С: Підприємство” дозволяє вести бухгалтерський та податковий облік, а також управляти виробничими процесами [5].

Проте використання таких систем має і певні недоліки. Впровадження ERP-систем часто супроводжується труднощами з адаптацією до потреб підприємства, а деякі системи потребують додаткового налаштування та навчання персоналу, що призводить до додаткових витрат.

Основним недоліком багатьох програмних продуктів є їх складність у використанні. Багато систем потребують додаткових модулів або налаштувань для відповідності специфічним вимогам підприємства. Крім того, документація багатьох систем часто доступна лише англійською мовою, що може ускладнити впровадження для підприємств без відповідних знань. Також вартість додаткових послуг та технічної підтримки часто є високою.

Автоматизація процесів залишається важливою складовою успішної роботи підприємства. Використання сучасних інформаційних систем дозволяє підвищити продуктивність, знизити витрати та покращити якість управління [5].

Аналіз сучасних інформаційних технологій показує, що автоматизація бізнес-процесів є важливою складовою ефективного управління підприємствами. Використання ERP та CRM дозволяє підвищити продуктивність та знизити витрати, хоча впровадження таких систем потребує додаткових ресурсів. Автоматизація процесів за допомогою сучасних інформаційних технологій відкриває нові можливості для підприємств, дозволяючи їм адаптуватися до змін на ринку та підвищувати ефективність управління.

1.4. Методи і засоби дослідження

Основним елементом для ефективного управління каталогом є структура бази даних, яка забезпечує зберігання і доступ до товарів. При розробці інформаційної системи для вебсайту необхідно побудувати реляційну базу даних, яка дозволяє чітко структурувати дані про продукти, категорії, підкатегорії та інші атрибути. Наприклад, кожен товар має бути пов’язаний із певною категорією чи підкатегорією, що забезпечує логічну організацію ієрархії товарів

на вебсайті. Це дозволяє користувачам зручно переміщатися по каталогу, а також використовувати різні фільтри та методи сортування.

Однією з основних задач при проєктуванні бази даних є моделювання зв’язків між таблицями, що зберігають інформацію про товари, їхні характеристики, ціни, знижки, доступність тощо. Наприклад, структура бази даних повинна дозволяти зберігання детальних атрибутів товару, таких як варіанти кольорів або розмірів, що необхідні для надання точних і релевантних даних користувачам.

Не менш важливим є питання забезпечення масштабованості бази даних. Оскільки каталог вебсайту може поступово розширюватися, структура бази даних повинна підтримувати додавання нових категорій, товарів та атрибутів без необхідності кардинальних змін у самій системі. Це дозволяє зберегти стабільність і працездатність вебсайту навіть при суттєвому збільшенні обсягу даних [6].

При проєктуванні баз даних необхідно забезпечити нормалізацію даних. Нормалізація схеми бази даних — покроковий процес розбиття одного відношення (на практиці: таблиці) відповідно до алгоритму нормалізації на декілька відношень на базі функціональних залежностей [7].

Нормалізація є процесом оптимізації структури бази даних з метою уникнення надлишкових даних і забезпечення максимальної ефективності управління інформацією. Вона є ключовою складовою проєктування бази даних для систем електронної комерції, оскільки дозволяє зробити дані більш організованими і зручними для обробки.

У процесі нормалізації бази даних використовуються три основні нормальні форми:

Перша нормальна форма (1NF): Усі значення в таблиці повинні бути атомарними, тобто неподільними. Це означає, що кожна клітинка таблиці повинна містити лише одне значення, а кожен стовпець повинен містити

однотипні дані. Наприклад, у таблиці з товарами кожен запис повинен включати лише одну ціну для одного товару, а не кілька цін для різних варіантів одного товару.

Друга нормальна форма (2NF): Вимога другої нормальної форми полягає в тому, що таблиця повинна бути у першій нормальній формі і всі неключові атрибути повинні залежати від усього первинного ключа, а не від його частини. Це забезпечує те, що дані не дублюються і належним чином зв'язані з ключовими атрибутами. Наприклад, якщо таблиця містить інформацію про замовлення, то всі деталі замовлення повинні залежати від номера замовлення (первинного ключа), а не від інших окремих елементів.

Третя нормальна форма (3NF): Таблиця повинна відповідати другій нормальній формі і не повинна містити транзитивних залежностей. Це означає, що неключові атрибути не повинні залежати один від одного, а лише від первинного ключа. Для каталогу товарів це означає, що, наприклад, ціна товару повинна залежати тільки від самого товару, а не від інших атрибутів, таких як категорія товару.

Четверта нормальна форма (4NF): Таблиця повинна відповідати третій нормальній формі і не повинна містити багатозначних залежностей. Це означає, що кожен неключовий атрибут має бути функціонально залежним від первинного ключа і не бути пов'язаним з іншими неключовими атрибутами через множинні зв'язки. Наприклад, якщо в таблиці зберігаються дані про постачальників товарів і місцезнаходження складів, необхідно створити окремі таблиці для постачальників і складів, щоб уникнути дублювання інформації в кожному записі.

П'ята нормальна форма (5NF): Таблиця повинна бути в четвертій нормальній формі і не повинна містити будь-яких залежностей між атрибутами, які можна розділити на менші таблиці без втрати інформації. Це необхідно для складних систем, де один набір даних може бути пов'язаний з іншим через кілька

проміжних атрибутів. Наприклад, якщо таблиця містить дані про товари, постачальників і умови доставки, слід розділити ці дані на окремі таблиці, щоб зберігати кожен атрибут у найменшій необхідній формі.

Нормалізація забезпечує цілісність даних, покращує продуктивність і зменшує обсяг надлишкової інформації. Використання нормальних форм гарантує, що структура бази даних є логічною, масштабованою та здатною ефективно обробляти запити користувачів [6].

Контент вебсайту також включає динамічну інформацію, яка змінюється залежно від стану товарів, акційних пропозицій або інших змін. Наприклад, система повинна мати можливість своєчасно оновлювати наявність товару або відображати актуальні ціни. Для цього бази даних повинні інтегруватися із зовнішніми системами, які надають ці дані в реальному часі, що є важливим для підтримання актуальності контенту. Це забезпечує гнучкість управління контентом та дозволяє підприємству швидко реагувати на зміни ринкових умов.

Одним із ключових викликів при управлінні каталогом є оптимізація запитів до бази даних. Це особливо важливо для великих каталогів, де обсяги даних можуть суттєво впливати на продуктивність системи. Запити повинні бути оптимізовані таким чином, щоб вони мінімально навантажували сервери і забезпечували швидку обробку запитів користувачів. Це досягається через правильне проєктування індексів, нормалізацію структури даних та використання ефективних методів обробки запитів.

Також необхідно врахувати питання безпеки даних. Оскільки інформація про товари може включати конфіденційні дані або дані, що мають значну комерційну цінність, система повинна забезпечувати захист від несанкціонованого доступу та зловживань. Це може включати використання шифрування даних, контроль доступу та інші засоби кібербезпеки, що забезпечують цілісність і конфіденційність даних [6].

Отже, управління контентом і каталогом вебсайту потребує проєктування бази даних і структури взаємодії між її елементами. Ефективна організація даних, їхня доступність, масштабованість та захищеність є необхідними для успішної роботи інформаційної системи. Це дозволяє забезпечити не лише зручність використання для користувачів, але й стабільну та ефективну роботу системи навіть за умов інтенсивного використання.

1.5. Постановка задачі

У сучасних умовах ведення бізнесу онлайн-присутність є невід'ємною складовою успішного функціонування будь-якої організації. Кафе, як і багато інших закладів громадського харчування, активно використовують вебсайти для взаємодії з клієнтами, надання інформації про страви, прийому замовлень та їх обробки.

Адміністрування вебсайту є одним з ключових аспектів його успішного функціонування, особливо коли йдеться про сайти, пов'язані з бізнесом, такими як сайт кафе. Адміністратор виконує численні функції, які мають на меті забезпечити правильне відображення меню, безперервне функціонування системи замовлень, управління користувачами та їх замовленнями.

Перш ніж перейти до виконання завдання з розробки сторінки замовлень, необхідно визначити, які є головні функції адміністратора сайту. Його основні обов'язки включають:

1. Обслуговування вебсайту: оновлення програмного забезпечення, плагінів, моніторинг помилок та виправлення проблем.
2. Управління контентом: організація, керування та публікація вмісту, забезпечення актуальності та відповідності стилю вебсайту.
3. Управління користувачами: керування обліковими записами, автентифікація, призначення ролей.
4. Безпека та резервне копіювання: встановлення заходів безпеки, моніторинг загроз, регулярне резервне копіювання даних.

5. Оптимізація продуктивності: поліпшення швидкості завантаження та продуктивності вебсайту.
6. Аналітика та звітність: аналіз трафіку та поведінки користувачів, створення звітів для покращення ефективності сайту.
7. Комунікація та підтримка: взаємодія з користувачами та надання технічної підтримки [13].

Конкретні обов’язки адміністратора вебсайту можуть відрізнятися залежно від розміру та складності вебсайту, а також вимог організації. У деяких випадках адміністратор веб-сайту може керувати всіма аспектами керування вебсайтом, тоді як у великих організаціях різні особи чи групи можуть відповідати за певні сфери, такі як створення вмісту, технічне обслуговування чи безпека [13].

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ

2.1. Проектування архітектури системи

Архітектура Model-View-Template (MVT) залишається фундаментальним підходом до розробки програмного забезпечення незалежно від обраного технологічного стеку, включаючи Spring Framework у Java. Цей підхід базується на ідеї чіткого розподілу відповідальності між різними компонентами системи, що дозволяє зберігати код структурованим, забезпечувати його легкість у підтримці та підвищувати гнучкість у подальшій розробці. У контексті Java Spring архітектура MVT успішно інтегрується у веб-додатки, спираючись на можливості цього фреймворку для роботи з модельними класами, поданнями та шаблонами.

Основна мета використання архітектури MVT полягає у створенні прозорої взаємодії між логікою програми, обробкою даних і представленням інформації для користувача. Усі три компоненти — Model (модель), View (подання) і Template (шаблон) — виконують визначені ролі, що сприяє ефективному управлінню складними системами та їх масштабуванню.

Модель (Model) відіграє роль центрального елементу, відповідального за доступ до даних, їх зберігання та обробку. У Spring цей компонент зазвичай реалізується за допомогою POJO-класів (Plain Old Java Objects), які описують сутності програми, а також використовують API для взаємодії з базами даних, як-от JPA (Java Persistence API). Завдяки анотаціям, як-от @Entity, @Table, @Id тощо, розробники можуть описувати структуру даних і взаємозв’язки між ними на високому рівні, не занурюючись у низькорівневі деталі SQL. Використання ORM дозволяє спростити маніпуляції з даними, забезпечуючи безпечність і зручність роботи з базами даних.

Подання (View) у Spring відповідає за обробку HTTP-запитів, виконання бізнес-логіки та повернення результатів у вигляді HTTP-відповідей. Цей

компонент зазвичай реалізується у вигляді контролерів, які використовують анотації, як-от @Controller або @RestController, для обробки запитів. Подання взаємодіє з моделлю для отримання необхідних даних, виконує бізнес-логіку та передає результати у відповідний шаблон. У Spring MVC (Model-View-Controller) контролери також можуть обробляти вхідні дані, виконувати валідацію та реагувати на помилки, що виникають під час виконання запиту.

Шаблони (Template) у контексті Spring є інструментом для рендерингу динамічного HTML, який користувач отримує як частину відповіді від сервера. Зазвичай у Java-проектах для роботи з шаблонами використовуються технології, як-от Thymeleaf, JSP (JavaServer Pages) або Freemarker. Шаблони дозволяють динамічно інтегрувати дані, отримані з моделі, у структуру HTML-сторінки, забезпечуючи тим самим генерацію контенту, що відповідає конкретному запиту користувача. Окрім того, використання шаблонів забезпечує розділення бізнес-логіки та представлення, роблячи код більш організованим і зрозумілим.

Архітектура MVT у Spring забезпечує високий рівень модульності та гнучкості, дозволяючи змінювати або вдосконалювати окремі компоненти системи без необхідності кардинальних змін в інших частинах. Наприклад, додавання нових полів до моделі або оновлення структури бази даних може бути реалізоване без внесення змін у шаблони чи логіку контролерів. Це спрощує процес розробки, особливо у великих командах, де відповідальність за різні компоненти може бути розподілена між різними спеціалістами.

Ще однією перевагою підходу MVT у Spring є можливість використання потужних інструментів і бібліотек, які підтримують сучасні вимоги до розробки. Наприклад, Spring Data JPA забезпечує високорівневу інтеграцію з базами даних, а Spring Security — механізми аутентифікації та авторизації. Усі ці інструменти можуть органічно вписуватися в архітектуру MVT, допомагаючи створювати масштабовані, безпечні та ефективні веб-додатки.

Однією з головних переваг Spring є його здатність інтегруватися з різними типами фронтенд-технологій, що дозволяє реалізовувати MVT як у традиційних серверно-орієнтованих додатках, так і в сучасних SPA (Single Page Applications). У традиційній моделі Spring MVC використовуються серверні шаблони для генерації сторінок, тоді як у SPA клієнтський інтерфейс може бути розроблений на базі таких фреймворків, як Angular чи React, а серверна частина Spring відповідає лише за обробку запитів API.

Важливим аспектом архітектури MVT є її відповідність принципам SOLID і DRY (Don’t Repeat Yourself). Завдяки чіткому розподілу відповідальності між компонентами забезпечується висока якість коду, його повторне використання та легкість у підтримці. Наприклад, усі операції з базою даних інкапсулюються у модельному шарі, а валідація та бізнес-логіка реалізуються на рівні подання, що дозволяє уникнути дублювання коду та полегшує внесення змін у систему.

Архітектура Model-View-Template у Spring забезпечує не лише зручність розробки, але й підвищує якість програмного забезпечення. Вона дозволяє будувати чітко структуровані веб-додатки, які відповідають вимогам сучасних стандартів у сфері програмної інженерії. Це робить її ефективним інструментом для створення як невеликих проектів, так і складних корпоративних систем, забезпечуючи баланс між простотою реалізації, гнучкістю та продуктивністю.

2.2 Проектування варіантів використання

Діаграма варіантів використання є одним із ключових засобів моделювання функціональних можливостей системи. Вона відображає взаємодію між користувачами (акторами) та прецедентами, що відповідають окремим функціям або сценаріям використання. Основна мета створення таких діаграм полягає в забезпеченні зрозумілого опису поведінки системи з точки зору кінцевих користувачів, а також у визначенні їхніх основних завдань і функцій, які система має підтримувати. Діаграма дозволяє ідентифікувати основні потреби

користувачів, структурувати функціонал і спростити процес планування розробки системи.

Рисунок 2.1. — Діаграма варіантів використання

Діаграма варіантів використання для системи Wasabi, наведена вище, представляє взаємодію адміністратора з основними функціями вебсайту кафе. У системі виділяється лише один актор — адміністратор, який відповідає за управління контентом, перегляд замовлень і керування інформацією про користувачів. Усі функції адміністратора чітко структуровані та відображають його ключові завдання в системі. Зокрема, адміністратор виконує такі прецеденти: перегляд списку страв, додавання нової страви, редагування страв,

перегляд замовлених страв, перегляд користувачів і перегляд здійснених замовлень.

Функція перегляду списку страв забезпечує адміністратора можливістю переглядати всі страви, доступні в меню. Це дозволяє оперативно контролювати актуальність інформації, включаючи назву, категорію та вартість страв. Сценарій використання передбачає відкриття списку страв через інтерфейс адміністратора, де відображаються всі доступні позиції. Завдяки цьому адміністратор може оцінити наявність помилок або неточностей у меню та швидко їх виправити за допомогою інших функцій.

Додавання нової страви є важливою функцією для підтримки актуальності меню. Сценарій передбачає введення адміністратором усіх необхідних даних про нову страву, включаючи її назву, опис, категорію, вартість і завантаження зображення. Ця функція забезпечує швидке та зручне оновлення асортименту, що дозволяє оперативно реагувати на зміни в пропозиції закладу, такі як сезонні акції або нові рецепти.

Функція редагування страв дозволяє адміністратору змінювати вже існуючі позиції в меню. Сценарій використання передбачає відкриття списку страв, вибір необхідної позиції та внесення змін у її опис, вартість або категорію. Завдяки цьому забезпечується коректність відображення меню для клієнтів і мінімізується ймовірність виникнення непорозумінь між користувачами та закладом.

Перегляд замовлених страв є важливою функцією, що надає адміністратору інформацію про те, які позиції з меню були замовлені користувачами. Ця функція допомагає аналізувати попит на окремі страви, виявляти популярні позиції та приймати управлінські рішення щодо їхнього просування або оновлення асортименту. У межах сценарію використання адміністратор може переглядати список замовлених страв із деталями, такими як кількість замовлень і середній рейтинг.

Функція перегляду користувачів дозволяє адміністратору отримувати інформацію про зареєстрованих клієнтів. Сценарій передбачає доступ до списку користувачів, де відображаються їхні імена, прізвища, логіни та інші дані. Адміністратор може аналізувати активність користувачів, відстежувати історію їхніх замовлень і вирішувати можливі проблеми, пов’язані з їхньою взаємодією із системою. Це сприяє покращенню сервісу та підвищенню рівня довіри клієнтів.

Перегляд замовлень є однією з найважливіших функцій для адміністратора, оскільки вона забезпечує контроль за всіма здійсненими замовленнями. Сценарій передбачає відкриття списку замовлень, який містить інформацію про ідентифікатор замовлення, ім’я користувача, назву страви та її вартість. Завдяки цій функції адміністратор може швидко реагувати на проблеми, що виникають під час обробки замовлень, забезпечуючи тим самим високу якість обслуговування.

2.3. Проектування бази даних

Проектування бази даних є важливим етапом розробки інформаційних систем, що визначає структуру, взаємозв’язки між сутностями та логіку зберігання даних. У процесі створення бази даних для системи управління кафе Wasabi були визначені основні сутності, які відображають ключові аспекти роботи системи: користувачі, ролі, категорії страв, страви, замовлення та відгуки. Кожна з цих сутностей була спроєктована з урахуванням бізнес-логіки системи, її функціональних вимог і принципів нормалізації даних.

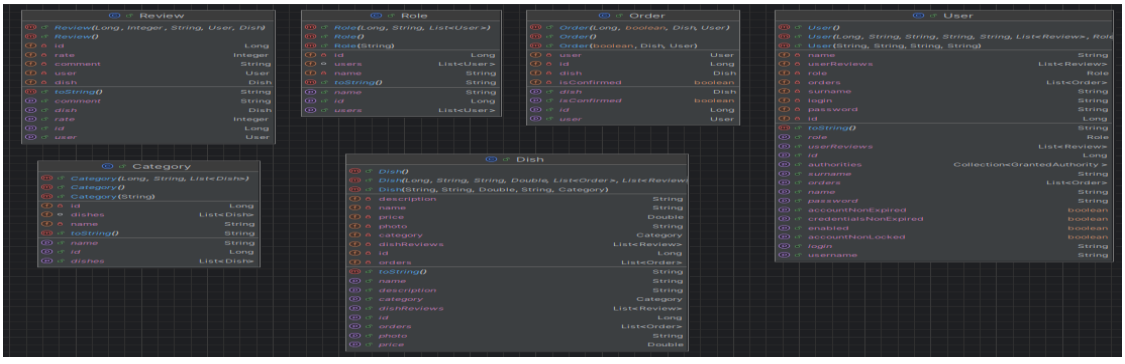

Рисунок 2.2. — Схема бази даних

Однією з базових сутностей у базі даних є сутність користувача, яка описує зареєстрованих на платформі осіб. Усі користувачі зберігаються у вигляді таблиці, де кожен рядок відповідає окремому користувачу. Таблиця має унікальний ідентифікатор користувача (id) як первинний ключ. Інші атрибути таблиці включають ім’я (name), прізвище (surname), логін (login), пароль (password) і пов’язані атрибути безпеки, такі як стан облікового запису (enabled, accountNonExpired, accountNonLocked, credentialsNonExpired). Для забезпечення ролей і прав доступу кожен користувач має зв’язок з сутністю Role, яка визначає доступні функції системи. Це реалізовано через зовнішній ключ, що пов’язує таблиці User та Role.

Сутність ролі (Role) описує доступні для користувачів типи доступу. Ця таблиця містить атрибути id як первинний ключ і name, який визначає назву ролі, наприклад, "адміністратор" або "користувач". Вона пов’язана з таблицею користувачів через відношення "один до багатьох", оскільки один тип ролі може бути застосований до декількох користувачів.

Для класифікації страв у меню було створено сутність категорії (Category), яка дозволяє групувати страви за певними тематиками або характеристиками. Таблиця категорій містить такі атрибути, як унікальний ідентифікатор (id) і назву категорії (name). Відношення між категоріями та стравами реалізоване через зв’язок "один до багатьох", оскільки одна категорія може містити кілька страв, але кожна страва належить лише до однієї категорії.

Сутність страви (Dish) відображає позиції меню, доступні для замовлення користувачами. Таблиця містить атрибути id, назву (name), опис (description), вартість (price), фото (photo) та зовнішній ключ до категорії (category). Для забезпечення гнучкості роботи із замовленнями та відгуками, страва пов’язана з таблицями замовлень і відгуків через відношення "один до багатьох". Завдяки цій

структурі можна отримувати інформацію про те, скільки разів певну страву було замовлено, а також її рейтинг, сформований користувачами.

Система замовлень, реалізована через сутність Order, є ключовим компонентом бази даних. Таблиця замовлень включає атрибути id, підтверджений статус замовлення (isConfirmed), а також зовнішні ключі до таблиць користувачів (user) і страв (dish). Це дозволяє визначити, який користувач замовив конкретну страву, а також статус виконання цього замовлення. Відношення між замовленнями, користувачами та стравами організоване таким чином, щоб забезпечити цілісність даних і відображати всі важливі аспекти взаємодії.

Окрім замовлень, у системі також важливу роль відіграють відгуки, реалізовані через сутність Review. Ця таблиця зберігає інформацію про оцінки та коментарі користувачів щодо певних страв. Відгуки мають унікальний ідентифікатор (id), рейтинг (rate), текст коментаря (comment), а також зовнішні ключі до користувача (user) і страви (dish). Така структура дозволяє аналізувати популярність та якість страв, а також зворотний зв’язок від клієнтів, що сприяє покращенню роботи кафе.

Усі сутності бази даних були спроєктовані відповідно до принципів нормалізації. Це дозволяє уникнути надлишковості даних і забезпечити їхню цілісність. Наприклад, дані про категорії зберігаються окремо від даних про страви, що дозволяє легко змінювати інформацію про категорії без необхідності оновлення всіх пов’язаних записів. Так само зв’язки між сутностями реалізовані через зовнішні ключі, що гарантує їхню взаємозалежність і підтримку актуальності.

Процес проектування бази даних також включав визначення методів оптимізації роботи системи. Наприклад, для швидкого пошуку страв за категоріями або аналізу замовлень були передбачені індекси на відповідних

атрибутах. Це значно підвищує продуктивність системи, особливо при великому обсязі даних.

Реалізація зв’язків між сутностями дозволяє забезпечити гнучкість роботи системи. Наприклад, якщо адміністратор додає нову страву до меню, ця інформація автоматично стає доступною для замовлення користувачами, а відгуки, залишені для цієї страви, коректно відображаються в пов’язаних звітах. Крім того, зміна даних у таблиці ролей дозволяє налаштувати права доступу для різних категорій користувачів без необхідності змін у базовій логіці системи.

Забезпечення безпеки даних є ще одним важливим аспектом проектування бази даних. Усі критичні дані, такі як паролі користувачів, зберігаються у зашифрованому вигляді, що мінімізує ризики несанкціонованого доступу. Крім того, базується система на принципах розмежування прав доступу, що дозволяє обмежувати доступ до певних таблиць або записів залежно від ролі користувача.

2.4. Проектування внутрішньої будови

Проектування внутрішньої будови програмного забезпечення є ключовим етапом у розробці інформаційних систем, що визначає організацію компонентів, їхню взаємодію та принципи функціонування. У системі управління кафе Wasabi внутрішня архітектура побудована за принципами багатошарової архітектури, що забезпечує чіткий розподіл відповідальності між компонентами, високу модульність та легкість у підтримці. Основні компоненти системи включають контролери, сервіси, репозиторії, моделі, а також класи для конфігурації безпеки та обробки винятків.

Рисунок 2.3. — Діаграма класів системи

Контролери є вхідною точкою для взаємодії з користувачами та відповідають за обробку HTTP-запитів. У проекті передбачено кілька спеціалізованих контролерів, таких як UserController, AdminController, CatalogController, ImageController, LoginController та RegistrationController. Кожен із них реалізує специфічний набір функцій, пов’язаних із певною сферою системи. Наприклад, UserController обробляє запити, що стосуються користувачів, зокрема отримання списків користувачів, перегляд деталей конкретного користувача та аналіз їхніх замовлень. AdminController забезпечує управління адміністративними функціями, такими як перегляд замовлень, редагування страв та управління категоріями. Контролери працюють із сервісами, які реалізують бізнес-логіку системи, і повертають клієнтам відповідні відповіді у форматі JSON.

Сервіси виступають ядром бізнес-логіки системи, забезпечуючи виконання функціональних вимог. У проекті реалізовані сервіси для роботи з користувачами (UserService), ролями (RoleService), стравами (DishService), категоріями (CategoryService), замовленнями (OrderService) та відгуками (ReviewService). Кожен із цих сервісів взаємодіє з відповідними репозиторіями для доступу до даних і реалізує необхідну логіку. Наприклад, DishService відповідає за додавання нових страв, оновлення їхніх даних, отримання списків страв за категоріями та обробку пов’язаних відгуків. OrderService виконує функції управління замовленнями, включаючи створення, підтвердження та перегляд деталей замовлень.

Репозиторії реалізують доступ до бази даних, використовуючи інтерфейси JPA (Java Persistence API). Кожен репозиторій відповідає за роботу з певною сутністю, наприклад, UserRepository, RoleRepository, DishRepository, CategoryRepository, OrderRepository та ReviewRepository. Ці компоненти забезпечують виконання запитів до бази даних, таких як пошук записів за

унікальними ідентифікаторами, фільтрація даних або отримання пов’язаних сутностей. Наприклад, DishRepository реалізує методи для пошуку страв за категоріями або назвами, а OrderRepository дозволяє відстежувати статуси замовлень користувачів.

Окрему роль у внутрішній будові системи відіграють класи DTO (Data Transfer Object), що забезпечують передачу даних між рівнями програми. У проекті використовується кілька спеціалізованих DTO-класів, таких як UserDTO, DishDTO, OrderDTO, ReviewDTO та DishOrderCountDTO. Вони допомагають формувати структуру даних, яка передається від сервісів до контролерів, і спрощують серіалізацію та десеріалізацію інформації. Наприклад, UserDTO включає лише ті поля, які необхідні для відображення інформації про користувачів, що дозволяє уникнути передачі зайвих даних і підвищує безпеку системи.

Важливим компонентом є конфігурація безпеки, реалізована у класі SecurityConfig. Вона забезпечує аутентифікацію та авторизацію користувачів, визначаючи доступ до певних частин системи на основі їхніх ролей. У проекті реалізовано поділ на ролі адміністратора та звичайного користувача, що дозволяє обмежувати функції, доступні для кожного типу облікового запису. Наприклад, лише адміністратор має доступ до редагування страв або перегляду замовлень усіх користувачів. Для зберігання паролів використовується шифрування, що підвищує безпеку системи.

Для обробки помилок та виняткових ситуацій у проекті передбачено глобальний обробник винятків (GlobalExceptionHandler). Цей компонент забезпечує централізоване управління помилками, що дозволяє обробляти некоректні запити або технічні збої у зрозумілій для користувачів формі. Наприклад, якщо користувач намагається отримати доступ до неіснуючої страви, система повертає відповідне повідомлення про помилку.

Окрім основних компонентів, у системі передбачені класи для фільтрації та пагінації даних. Наприклад, ListPaginationFilter та CatalogPaginationData забезпечують зручний механізм обробки великих обсягів інформації, дозволяючи користувачам переглядати лише необхідну частину даних. Це особливо актуально для роботи з меню, яке може містити велику кількість позицій.

Розробка системи здійснювалася з використанням принципів SOLID та інверсії залежностей. Це забезпечує гнучкість архітектури, що дозволяє легко змінювати або додавати нові компоненти без порушення роботи інших частин системи. Наприклад, можна впровадити новий тип користувачів або функціональність для управління акціями, мінімально впливаючи на існуючий код.

РОЗДІЛ 3
РОЗРОБКА СИСТЕМИ

3.1. Вибір засобів розробки

Вибір засобів розробки є одним із найважливіших етапів проектування програмного забезпечення, оскільки саме інструменти визначають функціональність, продуктивність і гнучкість системи. Для розробки системи управління кафе Wasabi було обрано мову програмування Java, фреймворк Spring і базу даних MySQL. Кожен із цих засобів забезпечує високий рівень надійності, продуктивності та масштабованості, що робить їх ідеальним вибором для створення веб-додатків подібного масштабу.

Java була обрана як основна мова програмування через її універсальність, потужність та популярність у розробці корпоративних систем. Java є мовою, яка використовує принципи об’єктно-орієнтованого програмування, що забезпечує високу структурованість коду та полегшує підтримку складних проектів. Її платформа-незалежність завдяки віртуальній машині Java (JVM) дозволяє запускати додатки на будь-якій операційній системі без необхідності внесення змін у код. Це забезпечує широку сумісність і легкість у розгортанні. Крім того, Java підтримує багатий набір бібліотек і фреймворків, які дозволяють розробникам зосередитися на бізнес-логіці, мінімізуючи час, витрачений на вирішення низькорівневих технічних задач. Її стабільність і довговічність доведені часом, оскільки вона залишається однією з провідних мов у галузі програмування протягом останніх двох десятиліть.

Важливим аспектом вибору Java є її активна спільнота розробників і широка документація, що забезпечує доступ до знань і ресурсів у разі виникнення складнощів. Крім того, Java є стандартом у багатьох компаніях завдяки її безпеці, можливості масштабування та підтримці багатопотоковості. Ці характеристики роблять її ідеальним вибором для розробки веб-додатків, які потребують

стабільності та високої продуктивності навіть при великій кількості користувачів.

Наступним важливим елементом у процесі розробки став вибір фреймворку Spring, який був обраний через його потужність і гнучкість. Spring є одним із найпопулярніших фреймворків для Java-розробки завдяки своїй модульній структурі, яка дозволяє використовувати тільки ті компоненти, які необхідні для конкретного проекту. Основою Spring є принцип інверсії контролю (IoC), що дозволяє зменшити залежність між компонентами системи та забезпечити їхню легку змінюваність. Це особливо важливо для великих систем, які потребують частого внесення змін або розширення функціональності.

Spring забезпечує повний набір інструментів для розробки веб-додатків, серед яких особливо корисними стали Spring Boot і Spring MVC. Spring Boot дозволяє швидко створювати готові до роботи додатки з мінімальною кількістю конфігурацій, що суттєво зменшує час розробки. Його автоматична конфігурація знімає з розробників необхідність вручну налаштовувати всі залежності, що особливо важливо для зменшення складності проекту на початкових етапах. Spring MVC, у свою чергу, забезпечує реалізацію архітектури Model-View-Controller, яка дозволяє чітко розділити бізнес-логіку, інтерфейс користувача та дані. Це забезпечує легкість у розробці, тестуванні та підтримці системи.

Ще однією причиною вибору Spring є його інтеграція з широким спектром технологій, включаючи бази даних, сервіси аутентифікації, механізми обробки запитів і багато іншого. Наприклад, у рамках проекту використовується Spring Data JPA, що забезпечує доступ до бази даних через об’єктно-реляційне відображення (ORM). Цей компонент дозволяє працювати з базою даних, використовуючи високорівневі методи, що значно спрощує роботу з даними та зменшує кількість написаного коду.

Spring Security є ще одним важливим модулем, який був інтегрований у проект. Він забезпечує безпеку системи, включаючи аутентифікацію,

авторизацію та захист від атак. Завдяки Spring Security у системі легко реалізувати розмежування доступу для різних категорій користувачів, таких як адміністратори та звичайні користувачі. Крім того, цей модуль підтримує інтеграцію зі стандартними протоколами безпеки, такими як OAuth2, що дозволяє впроваджувати сучасні методи аутентифікації.

Для зберігання даних у системі була обрана реляційна база даних MySQL, яка є однією з найпопулярніших баз даних у світі. MySQL була обрана через її високу продуктивність, надійність та легкість у використанні. Вона чудово підходить для зберігання структурованих даних завдяки підтримці складних запитів і механізмів транзакцій, що забезпечують цілісність даних. MySQL має відкритий код, що робить її доступною для широкого кола розробників і зменшує витрати на впровадження. Важливою перевагою є її сумісність із різними платформами та легкість інтеграції зі Spring через використання JPA та Hibernate.

Ще однією причиною вибору MySQL є її масштабованість, яка дозволяє ефективно працювати як із малими, так і з великими обсягами даних. Це особливо важливо для систем, які мають потенціал до зростання, таких як система управління кафе. MySQL підтримує створення індексів, реплікацію даних і горизонтальне масштабування, що робить її ідеальним рішенням для розробки продуктивних веб-додатків. Крім того, її активна спільнота користувачів і доступна документація сприяють швидкому вирішенню технічних питань.

Інтеграція між Spring та MySQL здійснюється за допомогою Hibernate, який є потужним інструментом для ORM. Hibernate дозволяє працювати з базою даних, використовуючи об’єктну модель, що значно зменшує складність написання SQL-запитів. Завдяки Hibernate розробники можуть зосередитися на бізнес-логіці, не витрачаючи час на технічні деталі роботи з базою даних. Він також забезпечує підтримку транзакцій, кешування та автоматичну генерацію схем бази даних, що прискорює процес розробки.

Вибір Java, Spring та MySQL як основних засобів розробки для системи Wasabi забезпечив створення стабільного, масштабованого та безпечного веб-додатка. Кожен із цих інструментів доповнює один одного, створюючи потужну екосистему для розробки програмного забезпечення. Завдяки цьому проект відповідає сучасним вимогам продуктивності, зручності в експлуатації та легкості у подальшому розширенні функціональності.

3.2. Розробка бази даних

Розробка бази даних є фундаментальним етапом створення інформаційної системи, адже вона визначає структуру зберігання даних, їх взаємозв'язки та механізми доступу. У рамках розробки системи управління кафе Wasabi було спроектовано реляційну базу даних, яка забезпечує збереження, обробку та інтеграцію даних, необхідних для функціонування платформи. Проект базувався на попередньо розробленій діаграмі класів, яка визначила ключові сутності системи, їх атрибути та взаємозв'язки. Основна увага приділялася нормалізації даних, оптимізації запитів та забезпеченню цілісності інформації.

Сутність користувача (User) є центральною в базі даних, адже вона відображає зареєстрованих осіб, які взаємодіють із системою. Таблиця користувачів містить унікальний ідентифікатор (id) як первинний ключ, а також атрибути, що описують основну інформацію про користувача: ім'я (name), прізвище (surname), логін (login) і пароль (password). Для підвищення безпеки всі паролі зберігаються у зашифрованому вигляді, що унеможливлює їхню компрометацію. Крім того, додаткові поля, такі як стан облікового запису (enabled) і його статуси (accountNonLocked, accountNonExpired, credentialsNonExpired), дозволяють забезпечити гнучке управління доступом до системи.

Для визначення ролей користувачів використовується сутність Role, яка відображає права доступу та функціональні можливості кожного типу облікового запису. Таблиця ролей містить два основних атрибути: унікальний ідентифікатор

(id) і назву ролі (name), наприклад "Адміністратор" або "Користувач". Зв’язок між таблицями User та Role реалізований через відношення "один до багатьох", що дозволяє одному типу ролі належати декільком користувачам. Це рішення забезпечує чітке розмежування прав доступу, підвищуючи безпеку системи та її масштабованість.

Структура меню кафе відображається через сутності Category та Dish, які забезпечують зручну організацію страв. Сутність Category описує групи страв, до яких можуть належати окремі позиції меню. Таблиця категорій включає унікальний ідентифікатор (id) та назву категорії (name), наприклад "Салати", "Десерти" або "Напої". Кожна страва у меню належить до однієї категорії, що реалізовано через зв’язок "один до багатьох".

Сутність Dish відповідає за збереження інформації про окремі позиції меню. Таблиця містить такі атрибути, як назва (name), опис (description), ціна (price), фото (photo) та зовнішній ключ до таблиці категорій (category_id). Ця структура дозволяє забезпечити високу деталізацію інформації про кожну страву, що сприяє покращенню користувацького досвіду. Крім того, зв’язок із сутністю Order дозволяє відслідковувати, які страви були замовлені користувачами, а зв’язок із Review забезпечує аналіз відгуків і рейтингів.

Система замовлень реалізована через сутність Order, яка забезпечує збереження інформації про всі транзакції, здійснені користувачами. Таблиця замовлень включає ідентифікатор замовлення (id), статус підтвердження (isConfirmed), а також зовнішні ключі до користувача (user_id) і страви (dish_id). Така структура забезпечує повний контроль за всіма аспектами замовлень, включаючи їхній статус і деталі. Зв’язок із таблицею користувачів дозволяє визначити, хто саме здійснив замовлення, тоді як зв’язок зі стравами дозволяє відстежувати популярність окремих позицій меню.

Для аналізу зворотного зв’язку від користувачів у системі передбачена сутність Review. Відгуки є важливим джерелом інформації для оцінки якості

страв і покращення обслуговування. Таблиця відгуків містить такі атрибути, як ідентифікатор відгуку (id), рейтинг (rate), коментар (comment), а також зовнішні ключі до таблиць користувачів (user_id) і страв (dish_id). Завдяки цьому можна визначати популярність страв, виявляти проблеми та розробляти стратегії для їх вирішення.

Загальна структура бази даних була спроектована з урахуванням нормалізації, що дозволило уникнути дублювання даних і забезпечити їхню цілісність. Наприклад, інформація про категорії та страви зберігається окремо, що дозволяє змінювати категорії без необхідності оновлення всіх пов’язаних записів. Зв’язки між таблицями реалізовані через зовнішні ключі, що гарантує підтримку референційної цілісності та забезпечує зручний доступ до даних.

Для оптимізації продуктивності системи були створені індекси на найбільш запитуваних атрибутах, таких як name у таблицях Dish та Category. Це дозволяє значно прискорити виконання пошукових запитів, особливо при великому обсязі даних. Крім того, транзакції використовуються для забезпечення цілісності даних під час виконання складних операцій, таких як оформлення замовлення або редагування страви.

База даних інтегрується із системою за допомогою об’єктно-реляційного відображення (ORM) через Hibernate. Це дозволяє працювати з базою даних на рівні об’єктів, використовуючи методи Java, що значно спрощує розробку та зменшує кількість необхідного коду. Автоматична генерація схем бази даних забезпечує відповідність між моделями даних і їхньою реалізацією, що полегшує впровадження змін і підтримку системи.

3.3. Розробка основних алгоритмів

Розробка основних алгоритмів є одним із ключових етапів у процесі створення інформаційної системи, оскільки саме вони забезпечують логіку роботи всіх функціональних модулів. Для системи управління кафе Wasabi були спроектовані та реалізовані алгоритми, які відповідають за найбільш важливі

операції, включаючи додавання нової страви, перегляд інформації про страву, авторизацію користувача та підтвердження замовлення. Для полегшення розуміння цих алгоритмів було створено відповідні блок-схеми, які відображають послідовність виконання дій і основні розгалуження логіки. Ці схеми дозволяють наочно представити складні процеси, полегшуючи аналіз і підтримку системи.

Рисунок 3.1. — Блок-схема додавання нової страви

Першим розглянуто алгоритм додавання нової страви до меню, який є ключовим для адміністратора системи. Блок-схема цього алгоритму демонструє, як отримані дані страви перевіряються на наявність помилок, і якщо такі знайдено, система повертає адміністратора на форму додавання із відповідними повідомленнями. У разі успішної перевірки виконується збереження фото страви, мапування об’єкта DTO у сутність бази даних та додавання її до бази. Завершується алгоритм перенаправленням адміністратора до форми додавання нової страви, що дозволяє продовжити редагування меню. Така послідовність

забезпечує структуровану обробку даних і гарантує, що меню завжди залишається актуальним.

Рисунок 3.2 — Блок-схема отримання даних про страву

Другий важливий алгоритм стосується перегляду інформації про конкретну страву. На блок-схемі цього процесу видно, що після отримання ідентифікатора страви система перевіряє її наявність у базі даних. Якщо страва існує, виконується обчислення середнього рейтингу на основі відгуків користувачів, а також формування детальної інформації, яка передається в модель для подальшого відображення на веб-сторінці. У випадку, якщо страва не знайдена, користувачу повертається відповідне повідомлення про помилку. Така логіка дозволяє забезпечити швидкий доступ до актуальної інформації та підвищити зручність користування системою для клієнтів.

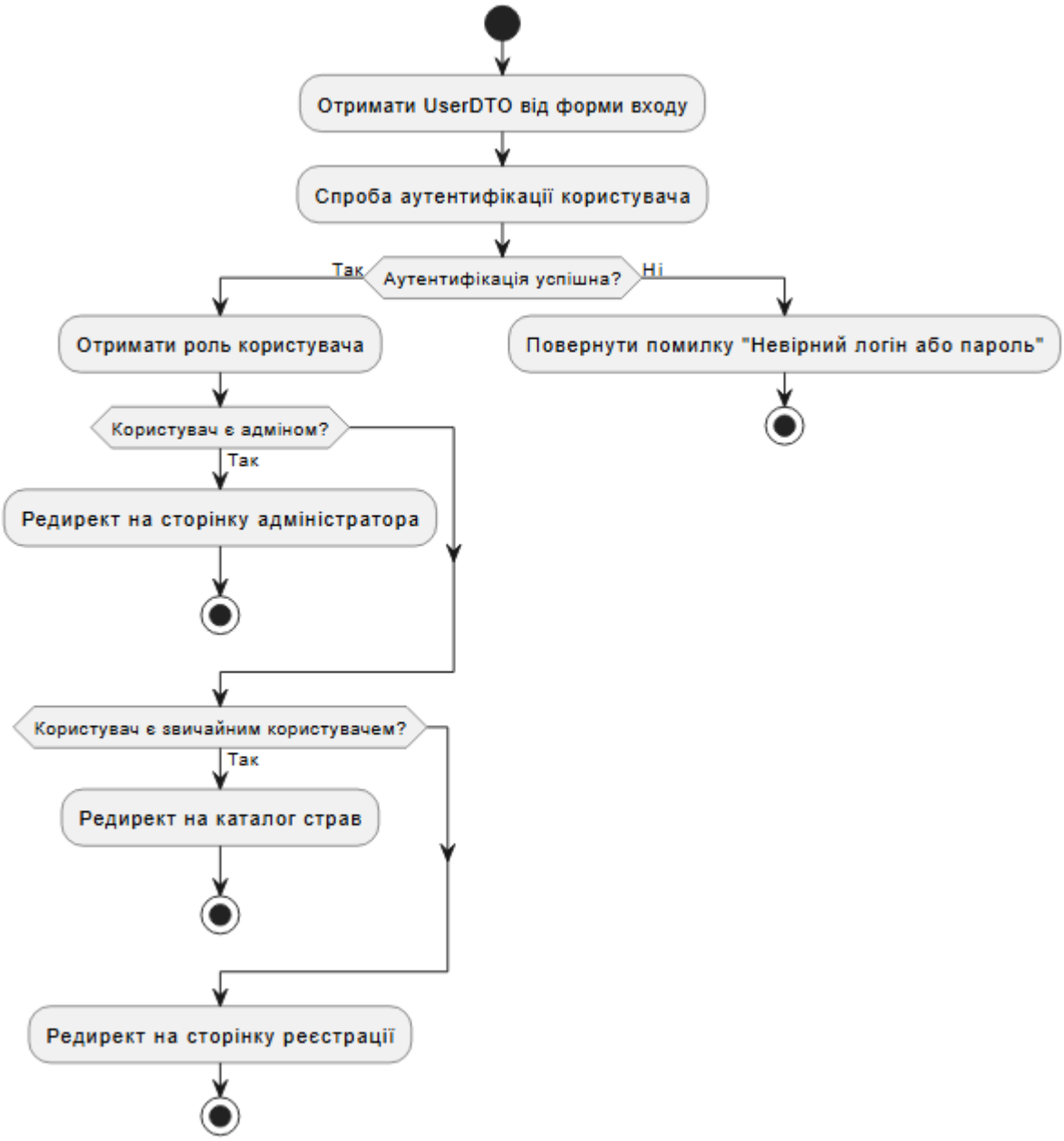

Рисунок 3.3 — Блок-схема авторизації користувача

Третім є алгоритм авторизації користувача, який забезпечує доступ до системи відповідно до ролі облікового запису. На блок-схемі видно, що після отримання даних із форми входу система проводить спробу аутентифікації. У разі успіху перевіряється роль користувача: якщо це адміністратор, його перенаправляють на сторінку управління системою, якщо звичайний користувач — на каталог страв. Якщо ж аутентифікація не вдалася, користувач отримує

повідомлення про невірний логін або пароль. Така структура алгоритму дозволяє забезпечити безпеку та точний розподіл доступу до функціоналу системи.

Рисунок 3.4. — Блок-схема підтвердження замовлення

Четвертий алгоритм стосується підтвердження замовлення, яке є важливою частиною роботи користувача із системою. Блок-схема показує, що після отримання ідентифікатора замовлення виконується перевірка, чи відповідає користувач власнику замовлення. У разі відповідності здійснюється оновлення статусу замовлення на "підтверджено" та його збереження в базі даних. Якщо перевірка не проходить, користувач отримує повідомлення про заборону доступу. Така логіка дозволяє уникнути несанкціонованих змін і підтримувати цілісність даних у системі.

Кожен із цих алгоритмів розроблений з урахуванням принципів надійності, безпеки та ефективності. Вони забезпечують виконання ключових функцій системи, таких як управління меню, обробка замовлень і доступ користувачів до

відповідного функціоналу. Використання блок-схем як візуального інструмента для опису алгоритмів не лише полегшує їх розуміння, а й сприяє оптимізації та підвищенню якості коду. Завдяки цьому досягається більш структурований підхід до розробки, що дозволяє уникнути потенційних помилок і підвищити продуктивність системи.

3.4. Розробка інтерфейсу користувача

Розробка інтерфейсу користувача є одним із найважливіших етапів у створенні інформаційних систем, оскільки саме інтерфейс забезпечує взаємодію між користувачем і функціональними можливостями системи. У системі управління кафе Wasabi інтерфейс був спроектований з урахуванням зручності, зрозумілості та естетичної привабливості, що сприяє покращенню користувацького досвіду та підвищенню ефективності виконання завдань. Особливу увагу приділено забезпеченню доступу до основних функцій, таких як перегляд списку користувачів, аналіз замовлень та статистики, а також реалізації зручної навігації.

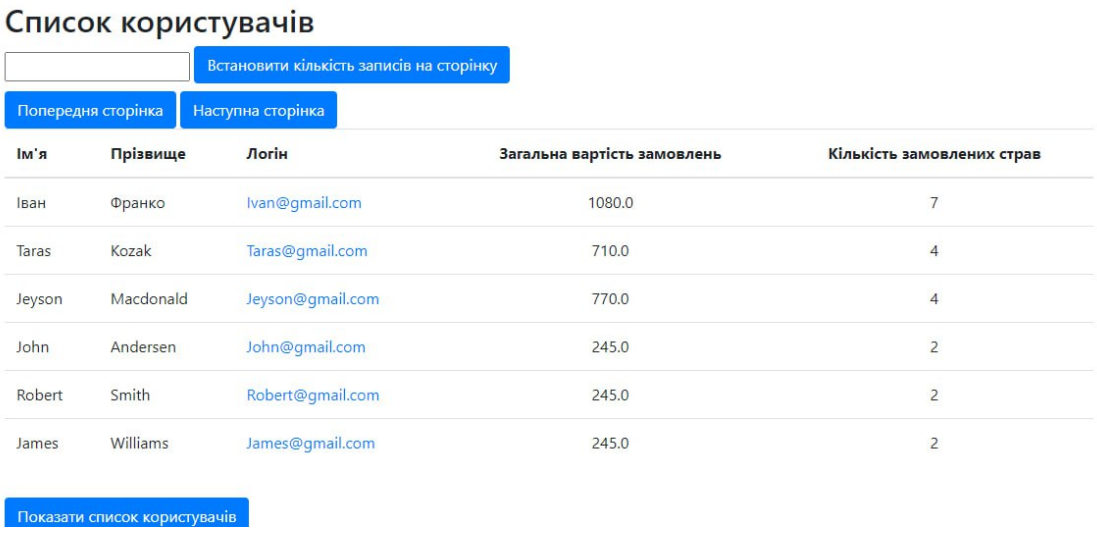

Рисунок 3.5. — Сторінка управління користувачами

Одним із ключових елементів інтерфейсу є сторінка перегляду списку користувачів, яка дозволяє адміністратору системи отримати доступ до основної інформації про клієнтів кафе. Ця сторінка містить таблицю, в якій

відображаються ім’я, прізвище, логін, загальна вартість замовлень та кількість замовлених страв кожного користувача. Для полегшення роботи з великим обсягом даних реалізовано функцію пагінації, яка дозволяє змінювати кількість записів на сторінці, а також перемикатися між сторінками. Такий підхід не лише покращує зручність користування, але й оптимізує швидкість завантаження сторінки, особливо при роботі з великою кількістю записів.

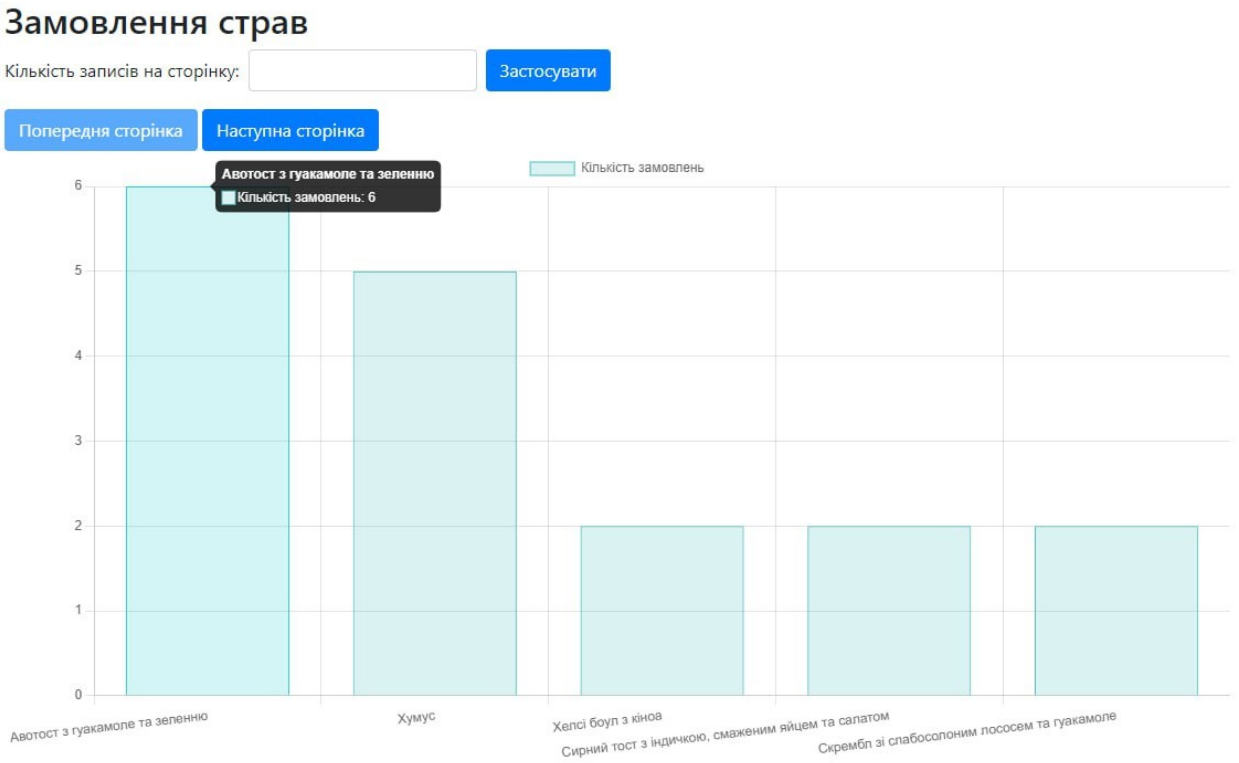

Рисунок 3.6. — Сторінка замовлень

Іншою важливою частиною інтерфейсу є сторінка замовлень страв, яка надає адміністраторам можливість аналізувати популярність різних позицій у меню. Інтерфейс цієї сторінки включає графік, який відображає кількість замовлень кожної страви. Завдяки цьому адміністратори можуть швидко отримати візуальну інформацію про те, які страви є найбільш популярними серед клієнтів, і на основі цих даних приймати управлінські рішення, наприклад, коригувати асортимент або розробляти акційні пропозиції. Графічне

представлення даних значно підвищує ефективність аналізу та спрощує сприйняття інформації.

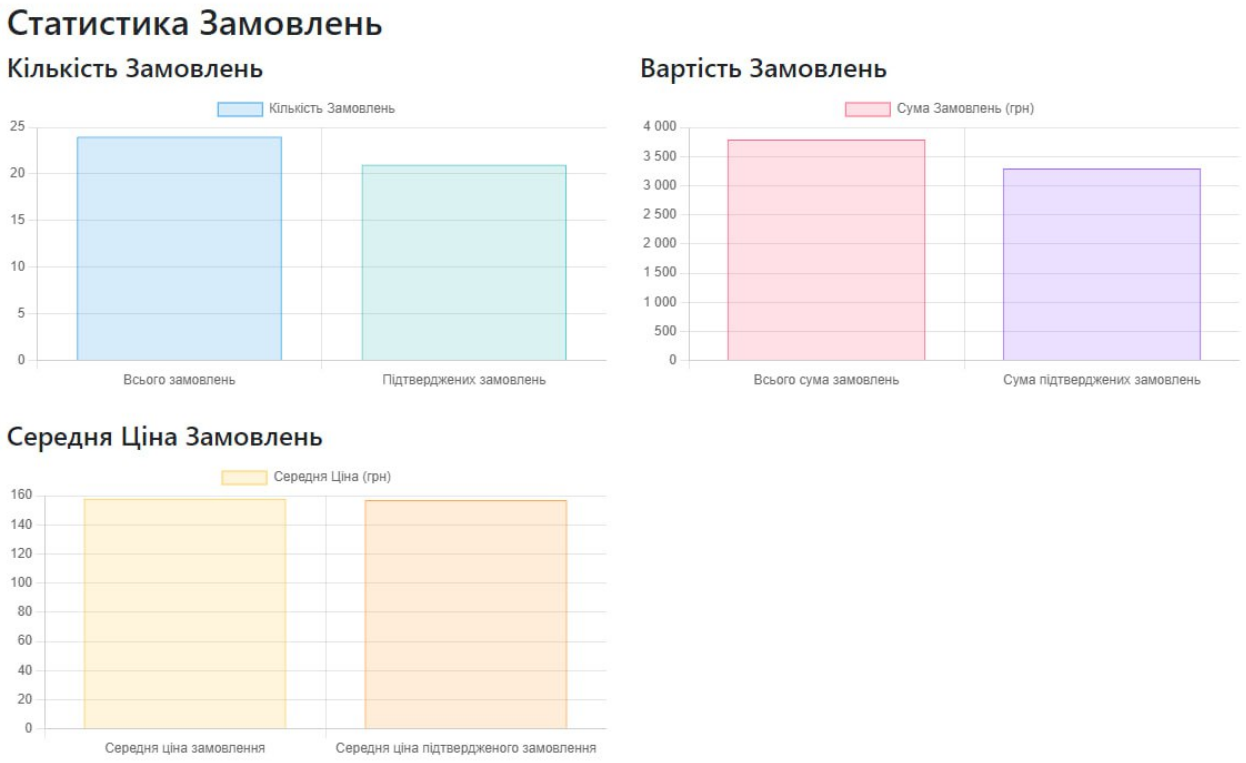

Рисунок 3.7. — Сторінка статистики замовлень

Окремою частиною інтерфейсу є модуль статистики замовлень, який дозволяє отримувати детальну інформацію про динаміку роботи кафе. На цій сторінці відображаються кількість замовлень (загальна і підтверджена), сума замовлень, а також середня вартість одного замовлення. Всі дані представлені у вигляді зрозумілих діаграм, що дозволяє адміністраторам легко оцінювати фінансові показники та ефективність роботи кафе. Графіки створені у зрозумілому форматі, деякі з них мають порівняльний характер, що полегшує оцінку результатів роботи у різні періоди.

Реалізація зручного та інтуїтивно зрозумілого інтерфейсу забезпечує швидкий доступ до функціональності системи. Використання сучасних технологій і принципів адаптивного дизайну дозволяє системі однаково добре функціонувати як на стаціонарних комп’ютерах, так і на мобільних пристроях.

Завдяки цьому система залишається доступною для користувачів у будь-який час і з будь-якого пристрою. Інтерфейс спроектований таким чином, щоб забезпечити логічну послідовність дій, уникнути зайвих елементів і сфокусуватися на ключових функціях.

Інтерфейс також забезпечує високу ступінь інтерактивності, дозволяючи користувачам виконувати дії без необхідності повторного завантаження сторінки. Наприклад, функція встановлення кількості записів на сторінці або перемикання між сторінками виконується за допомогою асинхронних запитів, що значно підвищує швидкість роботи системи. Це особливо важливо для великих обсягів даних, де зручність і швидкість взаємодії є ключовими факторами.

3.5. Тестування системи

Тестування системи є невід'ємною частиною розробки програмного забезпечення, що дозволяє переконатися в її коректній роботі та відповідності функціональним вимогам. У процесі тестування системи управління кафе Wasabi було проведено ретельну перевірку всіх її компонентів, включаючи функціональність, інтерфейс користувача, інтеграцію модулів, а також обробку даних у базі. Тестування проводилося за допомогою підготовлених тест-кейсів, кожен із яких був спрямований на перевірку конкретного аспекту роботи системи. Усі тестові сценарії завершилися успішно, що підтверджує стабільність та надійність системи.

Нижче наведено підсумкову таблицю тест-кейсів, яка демонструє виконані перевірки та їх результати.

Таблиця 3.1.

Тестування системи

№	Назва тест-кейсу	Опис тесту	Очікуваний результат	Фактичний результат	Статус

1	Авторизація користувача	Перевірка входу користувача з валідними та невалідними даними	Успішний вхід із валідними даними, помилка для невалідних	Успішний вхід із валідними даними, помилка для невалідних	Пройдено
2	Додавання нової страви	Перевірка функції додавання страви до меню	Страва успішно додається до бази даних	Страва успішно додається до бази даних	Пройдено
3	Перегляд інформації про страву	Відображення детальної інформації про обрану страву	Відображення повної інформації (назва, опис, ціна, рейтинг, фото)	Відображення повної інформації (назва, опис, ціна, рейтинг, фото)	Пройдено
4	Підтвердження замовлення	Перевірка можливості підтвердження замовлення лише авторизованим користувачем	Статус замовлення змінюється на "підтверджено" лише для авторизованих користувачів	Статус замовлення змінюється на "підтверджено" лише для авторизованих користувачів	Пройдено

5	Перегляд списку користувачів	Відображення списку зареєстрованих користувачів із можливістю пагінації	Відображення списку користувачів з коректною роботою пагінації	Відображення списку користувачів з коректною роботою пагінації	Пройдено
6	Аналіз замовлень страв	Перевірка відображення популярності страв у графічному вигляді	Коректне відображення кількості замовлень для кожної страви у вигляді діаграми	Коректне відображення кількості замовлень для кожної страви у вигляді діаграми	Пройдено
7	Перегляд статистики замовлень	Аналіз загальної кількості замовлень, сум замовлень і середньої ціни	Відображення коректних графіків для всіх показників	Відображення коректних графіків для всіх показників	Пройдено
8	Обробка неіснуючих ресурсів	Спроба доступу до неіснуючої	Повертається повідомлення "Ресурс не знайдено"	Повертається повідомлення "Ресурс не знайдено"	Пройдено

		страви або користувача			
9	Захист від несанкціонованого доступу	Перевірка обмеження доступу до функцій, доступних лише адміністраторам	Відображення помилки "Доступ заборонено" для неавторизованих користувачів	Відображення помилки "Доступ заборонено" для неавторизованих користувачів	Пройдено
10	Редагування даних про страву	Перевірка можливості змінювати дані про страву	Дані оновлюються в базі даних і відображаються коректно	Дані оновлюються в базі даних і відображаються коректно	Пройдено

Результати тестування демонструють, що система працює відповідно до специфікації, забезпечуючи коректну роботу всіх ключових функцій. Тестування охопило як основну функціональність, так і критичні сценарії, що дозволяє зробити висновок про стабільність системи. Всі виявлені помилки були виправлені ще на етапі розробки, що підтверджує якість коду та відповідність проекту високим стандартам розробки програмного забезпечення.

ВИСНОВКИ

У результаті виконаної роботи було розроблено інформаційну систему управління кафе, яка забезпечує автоматизацію ключових бізнес-процесів, включаючи адміністрування меню, обробку замовлень, управління користувачами та аналіз статистичних даних. Досягнуто мети дослідження шляхом створення системи, що відповідає сучасним вимогам надійності, безпеки та продуктивності. Використання сучасних інструментів і технологій, таких як Java, Spring та MySQL, дозволило реалізувати ефективне, масштабоване та зручне рішення.

Розроблена система успішно пройшла тестування, що підтвердило її функціональну відповідність технічним вимогам. Усі критичні сценарії роботи були перевірені, і система продемонструвала високу стабільність та коректність виконання основних завдань. Це забезпечує надійну основу для її впровадження у реальну діяльність кафе.

Практичне значення розробки полягає у зменшенні витрат часу на виконання рутинних завдань, підвищенні точності обробки замовлень та поліпшенні якості обслуговування клієнтів. Система надає адміністраторам зручний інструмент для управління меню, контролю замовлень і аналізу фінансових показників, що сприяє оптимізації операційних процесів та підвищенню конкурентоспроможності закладу.

Таким чином, результати роботи підтверджують доцільність автоматизації бізнес-процесів у сфері громадського харчування та демонструють можливості ефективного використання сучасних технологій для досягнення стратегічних цілей. Розроблена система є готовим до впровадження продуктом, який може бути використаний у кафе для оптимізації його діяльності та підвищення ефективності управління.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Almajali, Dmaithan (2016). "Antecedents of ERP systems implementation success: a study on Jordanian healthcare sector". Journal of Enterprise Information Management. 29 (4): 549–565. doi:10.1108/JEIM-03-2015-0024.
2. ^ Radovilsky, Zinovy (2004). Bidgoli, Hossein (ed.). The Internet Encyclopedia, Volume 1. John Wiley & Sons, Inc. p. 707. ISBN 9780471222026.
3. ^ Wilson, Deborah (19 April 2019). ""The ERP Software Market: $35 billion+, 40 years in the making, but still growing nicely!" by Chris Pang". Blogs.gartner.com. Retrieved 24 July 2022.
4. ^ Louis Columbus. "Predicting The Future Of Services-Centric ERP". Forbes. Retrieved 24 July 2022.
5. ^ Rubina Adam, Paula Kotze, Alta van der Merwe. 2011. Acceptance of enterprise resource planning systems by small manufacturing Enterprises. In: Proceedings of the 13th International Conference on Enterprise Information Systems, edited by Runtong Zhang, José Cordeiro, Xuewei Li, Zhenji Zhang and Juliang Zhang, SciTePress, p. 229 - 238
6. Designing an E-Commerce Database: Best Practices | AppMaster. AppMaster - The No-Code platform for building web & mobile apps. URL: https://appmaster.io/blog/designing-an-e-commerce-database (дата звернення: 13.09.2024).
7. Нормалізація баз даних. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Нормалізація_баз_даних (дата звернення: 13.09.2024).
8. Шаура А. Основні етапи створення дизайну вебсайтів. Збірник матеріалів всеукраїнської науково-практичної конференції. 2023. URL: https://fomd.kubg.edu.ua/images/2023/2023_Ф/Збірник_Всеукраїнська_конференція_ФОМД.pdf#page=152 (дата звернення: 13.09.2024).

9. Безпека інтернет-магазину: забезпечення кібербезпеки в eCommerce Wezom. WEZOM – Київ, Україна. URL: https://wezom.com.ua/ua/blog/kiberbezpeka-v-proektah-ecommerce-kompleksniy-gayd (дата звернення: 14.09.2024).
10. Що таке Firewall?. URL: https://2ip.ua/ua/blog/firewall (дата звернення: 13.09.2024).
11. Khan S. W. Cyber Security Issues and Challenges in E-Commerce. SSRN Electronic Journal. 2019. URL: https://doi.org/10.2139/ssrn.3323741 (дата звернення: 13.09.2024).
12. Основи Кібербезпеки для бізнесу. Інтернет провайдер WESTELECOM. URL: https://westelecom.ua/blog/osnovy-kiberbezopasnosti-dla-biznesa (дата звернення: 13.09.2024).
13. Що таке веб-адміністратор і чим він займається?. Big Red Jelly. URL: https://bigredjelly.com/uk/blog/what-is-a-web-administrator-and-what-do-they-do/ (дата звернення: 13.09.2024).
14. Restaurant admin dashboard - Shopurfood. Online food ordering system. URL: https://www.shopurfood.com/knowledge-base/admin (дата звернення: 13.09.2024).
15. Ontabee. URL: https://www.ontabee.com/ (дата звернення: 13.09.2024).
16. Shaul, L.; Tauber, D. (2012). "CSFs along ERP life-cycle in SMEs: a field study". Industrial Management & Data Systems. 112 (3): 360–384. doi:10.1108/02635571211210031.
17. ^ Khosrow–Puor, Mehdi. (2006). Emerging Trends and Challenges in Information Technology Management. Idea Group, Inc. p. 865.
18. ^ InfoWorld, Heather Harreld (27 August 2001). "Extended ERP technology reborn in B2B". Retrieved 20 July 2016.
19. ^ "A Vision of Next Generation MRP II", Scenario S-300-339, Gartner Group, April 12, 1990[third-party source needed]

20.^ Anderegg, Travis. "MRP/MRPII/ERP/ERM — Confusing Terms and Definitions for a Murkey Alphabet Soup". Retrieved 23 September 2013.
21.^ "ERP". Archived from the original on 10 July 2011. Retrieved 7 October 2009.
22.^ Sheilds, Mureell G. (2005). E-Business and ERP: Rapid Implementation and Project Planning. John Wiley and Sons, Inc. p. 9.