НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
_____комп'ютерних наук_____
(назва кафедри)

_____ _Голуб Б.Л.__
(підпис) (ПІБ)

"___" _____ 20 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Програмне забезпечення системи підтримки прийняття рішень керівництвом у мережі магазинів»

Спеціальність 121 – «Програмне Забезпечення Інформаційних Систем»

Гарант освітньої програми
___доцент к.т.н._____ _____ ___Голуб Б.Л.___
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи
____доцент к.т.н. професор_____ _____ ____Бушма О.В__
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ _____Здобнов Д.Д_____
(підпис) (ПІБ студента)

КИЇВ – 2024

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) _____інформаційних технологій_____

ЗАТВЕРДЖУЮ
Завідувач кафедри___комп'ютерних наук__
_____
_____доцент к.т.н._____ _____ Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"____" _____20_____року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

_____Здобнова Дмитра Дмитровича_____
(прізвище, ім'я, по батькові)

Спеціальність ___121 – «Програмне Забезпечення Інформаційних Систем»_____
(код і назва)

Освітня програма _____
(назва)

Орієнтація освітньої програми_____
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи___Програмне забезпечення системи підтримки__
___прийняття рішень керівництвом у мережі магазинів_____
_____

затверджена наказом ректора НУБіП України від "____"_____20___р. №_____

Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи _____
_____
_____
_____
_____
_____

Перелік питань, що підлягають дослідженню:

1. __Предметна область дослідження. Системи підтримки прийняття рішень_____
2. __Аналіз існуючих рішень_____
3. __Моделювання, розробка та тестування системи підтримки прийняття рішень_____

Перелік графічного матеріалу (за потреби) _____
_____

Дата видачі завдання "_____"_____20___ р.

Керівник магістерської кваліфікаційної роботи _____ ___Бушма О.В___
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ _____Здобнов Д.Д_____
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АСУП – автоматизована система управління підприємством;

БД – база даних;

СКБД – система керування базами даних;

СППР – система підтримки прийняття рішень;

ПЗ – програмне забезпечення;

DSS – Decision Support System;

UML – Unified Modeling Language;

ER – Entity-relationship;

LAMP – Linux, Apache, MySQL, PHP;

HTML – HyperText Markup Language;

PHP – Hypertext Preprocessor;

JS – JavaScript;

CSS – Cascading Style Sheets;

SQL – Structured Query Language.

ВСТУП

Протягом останніх років у світі спостерігається стрімкий розвиток інформаційних технологій. Разом із цим відбувається зростання конкуренції на ринку і ефективне управління бізнесом стає невід'ємною складовою успіху для будь-якого підприємства. Особливо це стосується мереж магазинів, для яких ефективна взаємодія з клієнтами, швидка обробка замовлень та оптимізація внутрішніх процесів необхідні для збереження конкурентних переваг. У таких умовах важливою складовою управління підприємством стає запровадження систем підтримки прийняття рішень, які дозволяють обробляти великі обсяги інформації, проводити аналіз даних і приймати обґрунтовані рішення. Сучасні системи підтримки прийняття рішень можуть поєднувати в собі різноманітні методи обробки даних, моделювання та прогнозування. Розробка та впровадження таких систем часто вимагає врахування специфічних потреб мереж магазинів, особливості ринку, вимоги до обробки інформації тощо.

В даній роботі розглядається розробка програмного забезпечення системи підтримки прийняття рішень керівництвом у мережі магазинів. Актуальність теми дослідження обумовлена стрімким розвитком інформаційних технологій та їхньою інтеграцією в різні сфери бізнесу, зокрема в галузі торгівлі. Сучасні мережі магазинів потребують ефективного управління запасами, обробки замовлень та прийняття рішень на основі аналізу наявних даних. Системи підтримки прийняття рішень стають необхідними, оскільки вони дозволяють оптимізувати бізнес-процеси та знижувати ризики, пов'язані з прийняттям рішень. Важливим є впровадження автоматизованих механізмів, які дозволяють швидко реагувати на зміни у попиті та запасах товарів.

Метою дослідження є створення програмного рішення, що дозволить автоматизувати процеси збору, обробки та аналізу даних для прийняття оптимальних рішень в управлінні магазином. Враховуючи можливі зміни у споживчих вподобаннях, економічних умовах та технологічному розвитку, розроблювана система має бути гнучкою, масштабованою та адаптивною, тобто

бути придатною для використання у різних магазинах, незалежно від сфери їхньої діяльності.

Об’єктом дослідження є мережа магазинів, торговельні процеси якої потребують управлінських рішень. Предметом дослідження виступає програмне забезпечення для підтримки прийняття рішень, яке забезпечує автоматизацію процесів управління товарами та замовленнями в цій мережі.

Завданнями дослідження є: системний аналіз предметної області, визначення основних проблем управління; формулювання вимог до системи підтримки прийняття рішень; моделювання архітектури та розробка системи, аналіз отриманих результатів.

Методи дослідження включають використання сучасних веб-технологій для створення інтерактивного інтерфейсу. Для аналізу даних і підтримки прийняття рішень будуть застосовані алгоритми автоматизації процесів обробки інформації, а також технології, що забезпечують адаптивність системи.

Наукова новизна роботи полягає в тому, що вперше буде розроблено веб-сайт як універсальну та просту у використанні систему підтримки прийняття рішень для мережі магазинів, що забезпечує автоматизацію процесів управління замовленнями та товарами. Запропоновано удосконалення архітектури системи з інтеграцією автоматизованих інтерфейсів для прийняття рішень.

У рамках роботи виконується дослідження, що включає аналіз існуючих рішень, вивчення методів обробки даних, а також визначення основних вимог до програмного забезпечення, які дозволять забезпечити максимальну ефективність та зручність використання для кінцевих користувачів. Робота має на меті проведення не лише теоретичних досліджень, але і практичну реалізацію, яка дозволить мережі магазинів підвищити ефективність управлінських процесів, скоротити час на прийняття рішень і, як наслідок, підвищити конкурентоспроможність на ринку. Також очікується, що результати роботи стануть корисними для подальших досліджень у галузі інформаційних технологій, відкриваючи нові можливості для впровадження інноваційних рішень у бізнес-практику.

Отже, дана робота спрямована на розробку комплексного підходу до створення програмного забезпечення системи підтримки прийняття рішень, що відповідає сучасним вимогам та потребам управління мережами магазинів, створюючи тим самим основи для подальшого вдосконалення процесів управління та прийняття рішень.

Робота складається з чотирьох розділів. Перший розділ присвячено системному аналізу предметної області, в ньому визначено мету, предмет та об’єкт дослідження, проаналізовано існуючі рішення та поставлено задачі дослідження. У другому розділі виконано моделювання системи, визначено її структуру, побудовано відповідні діаграми. Третій розділ включає розробку клієнтського та серверного програмного забезпечення, відповідно до визначених у другому розділі моделей. У четвертому розділі описано системні вимоги, наведено інструкцію користувачів, проведено тестування та описано отримані результати роботи.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Предметна область дослідження

В даній роботі розглядається розробка програмного забезпечення системи підтримки прийняття рішень керівництвом у мережі магазинів. Таке програмне забезпечення може використовуватись для оптимізації процесів управління підприємством, контролю наявності товарів, оцінки продажів, проведення акцій та рекламних кампаній тощо. Рішення орієнтовано на мережі магазинів, що мають опрацьовувати великі обсяги інформації про товари та замовлення. Впровадження такого програмного забезпечення обумовлено тим, що у зв’язку зі зростанням обсягів інформації та постійного розвитку інформаційних технологій управління мережами магазинів дедалі більше вимагає використання різноманітного програмного забезпечення для оптимізації різноманітних процесів, в тому числі, процесу прийняття рішень. Конкуренція на ринку роздрібної торгівлі зростає, і керівництво магазинів змушене шукати нові підходи для підвищення ефективності бізнес-процесів, зокрема, у сферах управління запасами товарів, обробки замовлень, взаємодії з клієнтами тощо.

Системи підтримки прийняття рішень у мережах магазинів дозволяють автоматизувати процес прийняття тих чи інших рішень персоналом, залежно від наявної інформації. Такі системи є спеціалізованою формою інформаційних систем, сконцентрованою на забезпеченні керівництва необхідною інформацією для прийняття обґрунтованих рішень. Вони можуть інтегрувати дані з різних джерел і використовувати аналітичні засоби для їх обробки, надаючи користувачам можливість аналізувати ситуації. Завдяки цьому, такі системи можуть значно підвищити якість прийняття рішень, зменшуючи ризики та невизначеність.

У системах підтримки прийняття рішень інформація відіграє ключову роль, оскільки саме на її основі формуються висновки та рекомендації. Збір, обробка та аналіз даних забезпечують створення цілісної картини, що дозволяє керівництву бачити не лише теперішній стан справ, але й потенційні сценарії

розвитку подій. Наприклад, у мережах магазинів інформація про продажі, запаси товарів, попит і поведінку споживачів може стати основою для подальшого прогнозування трендів і оптимізації бізнес-процесів. Таким чином, інформаційна система забезпечує не лише зручність управління, але й стратегічну перевагу в динамічному бізнес-середовищі.

Отже, враховуючи тісний зв’язок систем підтримки прийняття рішень з інформацією, доцільно розглянути детальніше роль інформації в економіці загалом та бізнес-процесах зокрема. Сьогодні інформація стала найважливішою економічною категорією та джерелом додаткового доходу і наявність релевантної системно організованої інформації стає значною економічною перевагою. Поняття інформації може бути визначено як зменшення невизначеності, що означає доцільність її використання під час аналізу прийняття ринкових рішень в умовах невизначеності. У підприємницькій діяльності, до якої зокрема відноситься діяльність мереж магазинів, доцільним є вживання терміну «економічна інформація». Під такою інформацією варто розуміти відомості про економічні явища та процеси, що знижують ступінь невизначеності економічних суб’єктів, перетворені у повідомлення, які можуть бути відтворені [1].

За способом використання, інформаційні ресурси можуть бути поділені на маркетингові та технологічні. Перші включають відомості про споживачів, постачальників, конкурентів, попит, методи і канали збуту тощо. Другі характеризують безпосередньо виробництво, науково-технічний прогрес тощо. Маркетингові ресурси, з якими працює система підтримки прийняття рішень, зазвичай менш стабільні, більш швидко змінюються і застарівають, формуються ззовні підприємства, але не залежать від виробничих процесів. Залежно від джерела, інформаційні ресурси можна поділити на внутрішні, що є більш точними, та менш точні зовнішні. При цьому важливо відзначити, що найважливішою характеристикою інформації є її достовірність, яка характеризується співвідношенням змістовної сторони інформації (об’єктивне відображення реальності) з реальністю [1].

Таким чином, систему, що розглядається в роботі, можна охарактеризувати як інформаційну систему, що обробляє внутрішню маркетингову інформацію з метою автоматизованого формування рекомендацій щодо прийняття рішень.

Проте, окрім визначення важливості ролі інформації у підприємницькій діяльності, для подальшого обґрунтування завдань системи підтримки прийняття рішень доцільно розглянути більш детально власне поняття прийняття рішень. Це дозволить більш поглиблено дослідити предметну область та поставити чіткі завдання до розроблюваного програмного забезпечення.

Робота систем підтримки прийняття рішень пов’язана з процесами прийняття рішень. Ці процеси притаманні всьому оточуючому світу і спостерігаються як у природних та біологічних структурах, так і в соціально-економічних системах. Діяльність осіб, що забезпечують управління підприємствами, тісно пов’язана з процесами прийняття рішень. Рішеннями називають обґрунтовану послідовність дій зі сторони особи, що приймає рішення, спрямовану на об’єкт управління і яка надає можливість досягнути поставленої мети або привести об’єкт до необхідного стану. При цьому управлінським рішенням є результат аналізу, прогнозування, економічного обґрунтування та вибору альтернативи з певної множини варіантів, спрямованих на досягнення цілей системи управління. Прийняття рішення є одним з видів розумової діяльності та проявом волі людини. Для рішення характерні наступні ознаки: наявність мети, можливість вибору з альтернатив та необхідність вольового акту особи, що приймає рішення. При цьому наявність альтернатив є обов’язковою, оскільки її відсутність означатиме відсутність вибору і, відповідно, самого процесу прийняття рішення [2].

Як було зазначено вище, на сьогодні інформаційні ресурси стали найважливішими складовими процесів управління. Це означає, що для прийняття рішень необхідною є обробка інформації, а також наявність засобів комунікації.

Процес прийняття рішення в загальному випадку складається з трьох основних етапів: аналіз, побудова альтернатив, вибір кращої альтернативи.

Аналіз передбачає діагностику проблеми, виявлення та опис відповідної ситуації, збір інформації. Проектування або побудова альтернатив передбачає визначення обмежень, що відокремлюють прийнятні варіанти від неприйнятних та визначення критеріїв вибору кращих варіантів шляхом дослідження можливості їх реалізації. Обрання кращої альтернативи здійснюється з використанням критеріїв, визначених на попередньому етапі, після чого виконується остаточне прийняття оптимального рішення [2].

Отже, прийняття рішень стає можливим тільки на підставі інформації про об’єкт управління, його процеси, що відбуваються в даний момент або з перебігом часу, та за наявності показників, за якими можна визначити якість та ефективність прийнятого рішення [2].

Під час прийняття рішень важливою характеристикою, пов’язаною зі ступенем структуризації проблеми і, відповідно, ефективністю рішень, є умови прийняття рішень. Сучасна теорія прийняття рішень передбачає наступну класифікацію умов: визначеність, ризик та невизначеність. Умови визначеності означають, що результат кожного з альтернативних виборів точно відомий заздалегідь. Рішення, що приймаються в умовах ризику, передбачають, що результати альтернатив невизначені, проте визначені їх імовірності. В умовах невизначеності рішення приймається тоді, коли оцінити імовірність потенційних результатів для альтернатив неможливо. Умови невизначеності є характерними для багатьох рішень, що приймаються в обставинах, що швидко змінюються. Такі рішення, як правило, приймає група експертів, а сам процес прийняття рішення складається з декількох ітераційних стадій [2].

Також варто зазначити, що процес прийняття рішень людиною пов’язаний з певними обмеженнями, що стосуються можливостей аналізу, обробки даних, належної якості, обґрунтованості та швидкості прийняття рішень. Робота людини обмежується відносинами між особами, фізіологічними та внутрішньо-психологічними факторами. Крім цього, люди нерідко проявляють невпевненість, деяку нелогічність, мінливість, спроби спрощувати задачі, особливо коли йдеться про задачі прийняття рішень з багатьма критеріями. За

умови, що людина не використовуватиме жодних допоміжних засобів під час прийняття рішень, можливо виділити наступні обмеження [2]:

1. Використання для обробки інформації робочої пам’яті, яка знаходиться між короткостроковою та довгостроковою зонами запам’ятовування. Тобто, під час прийняття рішень людина здатна обробляти тільки інформацію, що знаходиться в робочій пам’яті. Така інформація швидко забувається, якщо її стає багато.

2. Швидкість виконання осмислених операцій, що виконуються над елементами з робочої пам’яті людини, є обмеженою. Кожна операція мислення потребує деякого фіксованого проміжку часу. Чим складніший процес мислення, тим більше часу потребує обробка інформації.

3. Одержання інформації людиною можливе з двох джерел – органів почуттів і довгострокової пам’яті. При цьому, інформація з довгострокової пам’яті не завжди є достатньо надійною.

4. Обробка числових даних може супроводжуватись помилками в обчисленнях. В цьому випадку кожна елементарна операція потребує ще більше часу, потребуючи повторного виконання обчислень. Крім цього, розуміючи це обмеження, людина може свідомо уникати операцій, пов’язаних зі складними обчисленнями.

5. Зв’язок виконання операцій з часом і простором. Візуальне спостереження не завжди дозволяє отримати потрібні результати. Процес прогнозування на основі візуальної інформації потребує значно більше часу, як і у випадку зі складними обчисленнями.

Наведені вище обмеження є загальними і стосуються будь-яких випадків прийняття рішень без використання додаткових засобів. Це призводить до виникнення проблем, які необхідно враховувати під час прийняття рішень. Всі ці проблеми можна узагальнено назвати людським фактором. Застосування автоматизованих систем підтримки прийняття рішень дозволяє мінімізувати або навіть виключити негативний вплив розглянутих вище факторів на якість прийняття рішень.

Оскільки предметом даного дослідження виступає програмне забезпечення, призначене для підтримки прийняття рішень мережі магазинів, що забезпечує автоматизацію процесів управління товарами та замовленнями в цій мережі, наступним етапом розгляду предметної області є дослідження безпосередньо систем підтримки прийняття рішень.

На сьогоднішній день не існує єдиного, загальноприйнятого визначення системи підтримки прийняття рішень (СППР). Проте, існує кілька загальновживаних визначень, які відображають основні характеристики структури, функціонування та ефективності цих систем. Ці визначення акцентують увагу на ролі СППР у полегшенні процесу прийняття рішень, а також на їх здатності інтегрувати дані з різних джерел, аналізувати їх і надавати рекомендації, що дозволяють керівникам ухвалювати обґрунтовані рішення в умовах невизначеності [3, 4].

Розглянемо деякі визначення систем підтримки прийняття рішень [4]:

– це системи, що ґрунтуються на використанні моделей та процедур з обробки даних і думок, що допомагають керівникові приймати рішення;

– це інтерактивні автоматизовані системи, що допомагають особам, що приймають рішення, використовувати дані й моделі, щоб вирішувати неструктуровані та слабоструктуровані проблеми;

– це комп’ютерна інформаційна система, що використовується для підтримки різних видів діяльності під час прийняття рішень у ситуаціях, коли неможливо або небажано мати автоматичну систему, яка повністю виконує весь процес рішень;

– це специфічний та добре описуваний клас систем на базі персональних комп'ютерів.

Отже, узагалнюючи наведене вище, під системою підтримки прийняття рішень (СППР) пропонується розуміти інтерактивну комп’ютерну автоматизовану систему (або програмний комплекс), що призначена для допомоги та підтримки різних видів діяльності людини в процесі ухвалення рішень, зокрема в ситуаціях, що пов’язані зі слабкоструктурованими або

неструктурованими проблемами. Ці системи забезпечують користувачів необхідною інформацією та аналітичними інструментами для оптимізації процесів прийняття рішень, що робить їх незамінними в умовах сучасного управління [3].

Термін «система підтримки прийняття рішень» (або DSS, від англ. Decision Support System) вперше з'явився в 70-х роках XX століття, і його автори – Горрі та Мортон. Перші версії СППР фактично не відрізнялися від класичних управлінських інформаційних систем, тому в багатьох випадках використовувався термін «системи управлінських рішень». Однак із розвитком технологій та потреб бізнесу ці системи стали більш спеціалізованими, надаючи користувачам нові можливості для аналізу даних і підтримки ухвалення рішень у складних ситуаціях [4].

Сучасні інтелектуальні системи підтримки прийняття рішень (ІСППР) відрізняються активним застосуванням інтелектуальних методів для обробки даних і ухвалення рішень. Вони також спроектовані з урахуванням принципів, які покращують взаємодію між користувачем і системою. Інтелектуалізація цих систем полягає в адаптації до реакцій користувача, його вподобань у відображенні результатів аналізу, а також у виборі найбільш зручних методів введення, редагування та оновлення бази знань і даних. Це дозволяє зробити процес роботи з системою більш інтуїтивним і ефективним [3].

Інтелектуальні системи підтримки прийняття рішень утворюють різноманітний і надзвичайно корисний клас інформаційних обчислювальних систем, що спеціалізуються на обробці даних. Вони забезпечують можливість інтеграції широкого спектра інформації з різних джерел, що дозволяє користувачам отримувати комплексні аналітичні дані. Завдяки цій інтеграції, ІСППР можуть значно полегшити процес ухвалення рішень, надаючи необхідну інформацію у зручному для користувача форматі. Це робить їх незамінними у багатьох сферах, де важливо швидко та ефективно аналізувати дані для прийняття рішень [3].

Різноманітність визначень систем підтримки прийняття рішень відображає широкий спектр форм, розмірів і типів цих систем. Незважаючи на це, більшість комп'ютерних систем СППР має чітко окреслену структуру, що включає три основні компоненти: інтерфейс користувача, база даних, база моделей. Підсистема інтерфейсу користувача забезпечує взаємодію між користувачем і системою, дозволяючи зручно вводити дані та отримувати результати. Підсистема управління базою даних відповідає за зберігання, організацію та доступ до інформації, що використовується в процесі прийняття рішень. Підсистема управління базою моделей містить аналітичні моделі та алгоритми, які використовуються для обробки даних і генерування рекомендацій [4].

Ця структурована архітектура дозволяє ефективно обробляти інформацію і підтримувати користувачів у складних ситуаціях ухвалення рішень. Описані вище три компоненти складають основну структуру класичної системи підтримки прийняття рішень, яка відрізняє її від інших типів інформаційних систем, зокрема тих, що використовують сховища даних і мають аналогічну англійську абревіатуру DSS. У сучасному світі, з розвитком Інтернет-технологій, до СППР часто додається новий компонент – система управління електронною поштою та повідомленнями, що значно розширює можливості цих систем і покращує комунікацію між користувачами [4].

Описану вище структуру можна зручно представити графічно у вигляді схеми. Загальна структура класичної системи прийняття рішень наведена на рис. 1.1, де відображено зв'язки між елементами класичної системи підтримки прийняття рішень.

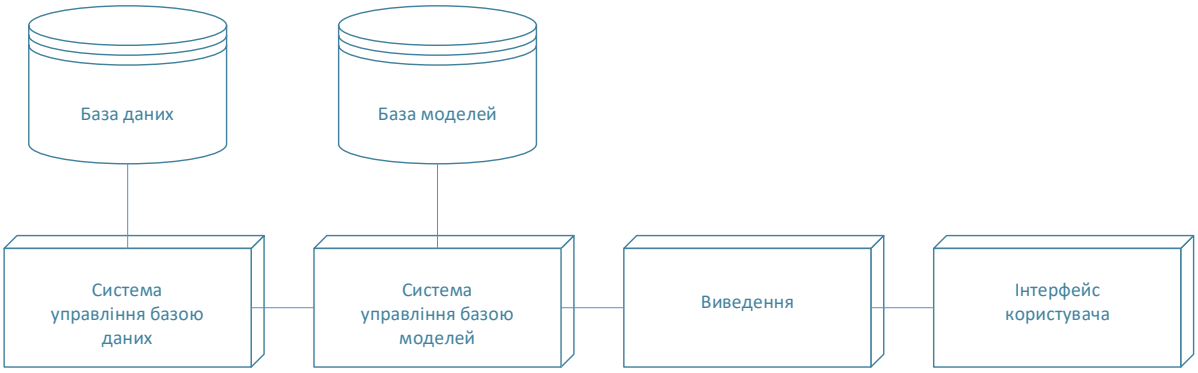

Рис. 1.1 Класична структура системи підтримки прийняття рішень

Наведену вище структуру можна деталізувати, розділивши узагальнений інтерфейс користувача на дві підсистеми: підсистему введення та аналізу запитів і підсистему оброблення запитів користувача та генерації результатів. При цьому перша підсистема має включати опис всіх можливих (допустимих) запитів користувача, а також їх формальне подання. Важливо правильно визначити тип введення, що використовується в системі та обґрунтувати його вибір. Друга підсистема виконує обробку коректних запитів користувача, звертаючись до баз моделей та знань для отримання необхідних даних та критеріїв (правил) прийняття рішень [3].

Виходячи з цього, можна також зробити висновок про можливість об’єднання бази даних із базою моделей на практиці. Іншим можливим варіантом є реалізація бази моделей у жорсткому програмному вигляді. Всі ці рішення змінюють структуру програмного забезпечення, проте зберігають загальну структуру системи підтримки прийняття рішень. Детальніше обґрунтування структури системи буде розглянуто у наступних розділах роботи.

Серед напрямів розробки людино-машинних систем сьогодні можна виокремити системи автоматичного управління, експертні системи та системи підтримки прийняття рішень. СППР виявляються найбільш ефективними для вирішення різноманітних завдань, зокрема в області розподілу ресурсів. Завдяки СППР організації можуть безпосередньо використовувати обчислювальні засоби для проектування, порівняння та вибору альтернативних варіантів рішень, застосовуючи різні методи та підходи для оптимізації процесу ухвалення рішень. Це дозволяє досягати кращих результатів і підвищувати ефективність управлінських процесів [3].

У багатьох ситуаціях автоматизація процесу прийняття рішень стає більш ефективною завдяки системам підтримки прийняття рішень, що здатні допомагати в ухваленні важливих рішень, орієнтуючись на події, які ще не відбулися. Завдяки такій функціональності, ці СППР дозволяють створювати різноманітні сценарії, які відповідають на запитання типу «що було б, якби», а також допомагають у визначенні оптимальних дій у конкретних умовах [3].

Проте, у випадку з мережею магазинів, першочерговою задачею стає обробка наявної інформації, а не прогнозування майбутнього. Це пов’язано зі специфікою проблем, які можуть виникати у мережах магазинів. Тому доцільно провести стислий огляд особливостей, характерних для магазинів як об’єктів управління та проблем, пов’язаних з ними.

Мережі магазинів стикаються з багатьма проблемами, які можуть суттєво вплинути на їхню прибутковість. Однією з найактуальніших проблем є необхідність своєчасних перевірок наявності товарів на складі. Часті помилки в управлінні запасами можуть призводити до нестачі популярних товарів або, навпаки, до їх перевантаження, що негативно відображається на фінансових показниках. Системи підтримки прийняття рішень здатні автоматизувати процес моніторингу запасів, аналізуючи дані про продажі, сезонність та тренди споживання. Це дозволяє магазинам своєчасно поповнювати запаси, уникати втрат через нестачу товарів та оптимізувати обсяги замовлень.

Крім того, проведення акцій і встановлення знижок є ще однією важливою проблемою, яку можуть вирішити СППР. Правильний підбір стратегій знижок і акційних пропозицій може суттєво збільшити продажі, однак для цього необхідно провести аналіз споживчих уподобань, конкурентних цін та сезонних коливань. СППР можуть допомогти в моделюванні різних сценаріїв, прогнозуючи, як зміна цін вплине на обсяги продажів, що дозволяє менеджерам приймати обґрунтовані рішення про акції, які принесуть максимальний прибуток.

Ще однією проблемою, яку можна вирішити за допомогою СППР, є оптимізація асортименту товарів. В умовах великої конкуренції важливо пропонувати споживачам актуальний асортимент, який відповідає їхнім потребам і смакам. СППР можуть аналізувати дані про продажі, поведінку споживачів та ринкові тренди, що дозволяє приймати рішення про необхідність додавання нових товарів до асортименту або зняття з продажу ті, що не користуються попитом. Таким чином, системи підтримки прийняття рішень

допомагають мережам магазинів зберігати конкурентоспроможність і адаптуватися до умов ринку.

Таким чином, інтеграція СППР в управлінські процеси мереж магазинів дозволяє значно підвищити їхню ефективність, знизити ризики і покращити обслуговування клієнтів, що, в свою чергу, позитивно вплине на загальну прибутковість бізнесу.

Отже, підсумовуючи проведений аналіз предметної області дослідження, можна зробити висновок, що система підтримки прийняття рішень у мережі магазинів дозволяє вирішити велику кількість проблем. Проте найбільшою з них є саме людський фактор, оскільки людина – особа, що приймає рішення, має значні обмеження в аналізі великих обсягів інформації. Тобто застосування таких систем є більш ефективним за умови роботи з великими обсягами даних. В контексті мережі магазинів це означає, що прийняття рішень щодо великої кількості товарів або замовлень потребує обов’язкового застосування спеціалізованих систем.

Крім цього, системи підтримки прийняття рішень надають зручний користувацький інтерфейс, а їхня структура передбачає певну гнучкість, важливу в умовах можливих змін та масштабування.

Враховуючи потреби мереж магазинів, стає очевидно, що застосування систем підтримки прийняття рішень є необхідним, особливо в умовах великої кількості товарів та замовлень. Потреба в автоматизації процесу прийняття рішень обумовлена тим, що цей процес потребуватиме меншого часу, порівняно з ручною обробкою даних, буде враховувати всю наявну інформацію, відповідно до розробленої моделі прийняття рішень та пропонувати користувачу, відповідальному за прийняття рішень свої альтернативи, визначені за допомогою моделі прийняття рішень.

Отже, на основі аналізу предметної області дослідження можна зробити висновок про необхідність розробки ефективних програмних рішень для підтримки прийняття рішень керівництвом у мережах магазинів. Відповідно до розглянутих потреб можуть бути сформульовані задачі розроблюваної системи.

1.2 Аналіз наявних рішень

Оскільки системи підтримки прийняття рішень отримали широке застосування у багатьох сферах діяльності, на сьогодні відома значна кількість таких систем. Проте не всі вони придатні для застосування в мережах магазинів, але і спеціалізовані системи, призначені для магазинів, можуть мати як переваги, так і недоліки. Крім того, існує програмне забезпечення, яке використовується для роботи спільно з системами підтримки прийняття рішень, їхньої розробки та моделювання.

В рамках даної роботи все наявне програмне забезпечення можна умовно розділити на спеціалізоване, призначене для застосування в магазинах, та універсальне, призначене для використання будь-якими підприємствами. Це необхідно тому, що універсальне програмне забезпечення може не враховувати специфіку магазинів, а також мати надлишковий функціонал, що ускладнюватиме використання програмного забезпечення.

Також необхідно провести аналіз наявних наукових публікацій за темою роботи. Це дозволить побачити поточний стан ситуації та актуальні напрямки досліджень, а також проаналізувати результати, отримані іншими дослідниками. При цьому важливо приділяти увагу як системам, орієнтованим на інтернет-магазини, так і більш універсальним рішенням, а також спеціалізованим системам, що застосовуються в інших галузях.

Виконання аналізу наявних рішень пропонується почати з наукових літературних джерел, а саме статей та тез доповідей. Після цього доцільно перейти до огляду наявних програмних рішень, орієнтованих на магазини, а також універсальних систем підтримки прийняття рішень, які можуть бути застосовані для різних цілей. Остання категорія програмного забезпечення є найбільшою, проте в рамках даної роботи не потребує поглибленого аналізу, оскільки не орієнтована на мережі магазинів. В останню чергу доцільно оглянути програмне забезпечення, призначене для виконання моделювання у системах підтримки прийняття рішень, їх розробки або виконання додаткових функцій.

Детальний аналіз літературних джерел показав, що в українській науковій літературі питання розробки систем підтримки прийняття рішень для мереж магазинів або взагалі для застосування саме в галузі торгівлі на сьогодні є недостатньо дослідженим. Існуючі наукові публікації присвячені у більшості випадків розробці більш широкопрофільних систем підтримки прийняття рішень, їхньому моделюванню та оптимізації. При цьому відомі також навчальні посібники, в яких детально описуються процеси проєктування комп'ютерних систем підтримки прийняття рішень. Такі літературні джерела не мають конкретики щодо застосування систем у сфері торгівлі, але надають багато інформації щодо розробки систем підтримки прийняття рішень в цілому.

Зважаючи на зазначене вище, огляд літературних джерел пропонується розпочати саме з таких посібників, оскільки вони відображають відомі на сьогодні методи та засоби розробки систем підтримки прийняття рішень. Автори надають детальний опис різноманітних підходів та загальні теоретичні відомості про такі системи.

Одними з прикладів найбільш ємнісних літературних джерел, що можуть використовуватись під час моделювання та розробки систем підтримки прийняття рішень, є навчальний посібник «Проектування комп'ютерних інформаційних систем підтримки прийняття рішень», виданий Інститутом прикладного системного аналізу Національного технічного університету України «Київський політехнічний інститут» [5] та підручник «Системи і методи підтримки прийняття рішень», виданий в цьому ж університеті [3]. Автори посібників детально описують методику проектування інформаційних систем підтримки прийняття рішень з точки зору системного аналізу. Підручник [3] є найновішим відомим на сьогодні українським літературним джерелом для вивчення та розробки систем підтримки прийняття рішень. Він є продовженням та доповненням виданого раніше навчального посібника [5]. Автори підручника розглядають функціональну структуру та альтернативні підходи до розробки систем підтримки прийняття рішень, використовуючи різні типи моделей проектування, а також використовуючи створення прототипу. Детально

розглядаються всі етапи проектування СППР, включаючи складання технічного завдання, створення проекту системи, розробку та тестування всіх модулів, супровід системи на етапі експлуатації. Також авторами розглядаються основні типи архітектур систем підтримки прийняття рішень у залежності від обмежень, пов’язаних з обробкою даних, типами інформації, процесами оцінювання альтернатив. Значна частина даних літературних джерел присвячена математичному забезпеченню систем підтримки прийняття рішень, зокрема, авторами розглядається прогнозування часових рядів за допомогою СППР, а також системи, побудовані на основі експертних оцінок та мереж Байєса.

Також відомий навчально-методичний посібник «Системи підтримки прийняття рішень», виданий Запорізькою державною академією [4] та однойменний навчальний посібник, виданий Національним гірничим університетом [6]. У цих посібниках автори надають загальну теоретичну інформацію про побудову та застосування систем підтримки прийняття рішень. Значна увага приділяється основам теорії прийняття рішень, що закладає теоретичне підґрунтя і стає основою для розробки автоматизованих систем підтримки прийняття рішень. Матеріал викладений на рівні, доступному для самостійного вивчення студентами. Посібники розглядають стислу історію розвитку та теоретико-методологічні основи теорії прийняття рішень, наводять класифікацію та основні базові компоненти (структуру) систем підтримки прийняття рішень, аналізують етапи розвитку та впровадження таких систем. Автори наводять типові структурні схеми систем підтримки прийняття рішень, описують взаємодію користувачів з системами та наводять приклади алгоритмів прийняття рішень. Навчальний посібник [6] також надає огляд програмного забезпечення, що використовується для моделювання в системах підтримки прийняття рішень. Ця особливість посібника має значну практичну користь, оскільки відображає наявні програмні рішення, що можуть бути використані під час самостійної розробки СППР.

Розглянуті вище літературні джерела доповнюють одне одного і надають достатній обсяг теоретичних відомостей для розробки власних систем. Можна

зробити висновок про те, що тема проектування та розробки систем підтримки прийняття рішень добре опрацьована в українській науковій літературі.

Проте, окрім теоретичної інформації, наведеної у підручниках та навчальних посібниках, доцільно розглянути наукові джерела, що відображають результати досліджень у сфері розробки систем підтримки прийняття рішень. Як зазначалось раніше, такі публікації відомі, проте більшість з них присвячені проектуванню систем підтримки прийняття рішень в цілому, без врахування специфіки тієї чи іншої галузі. В свою чергу більш спеціалізовані дослідження, спрямовані на застосування систем для вирішення конкретних задач у більшості випадків не пов’язані з роботою магазинів. Серед статей зустрічаються присвячені застосуванню СППР у різноманітних галузях: містобудівництво, рослинництво, процеси споживчого кредитування, медична діагностика, прогнозування нестаціонарних процесів тощо. Найбільш наближеними науковими публікаціями в даному випадку є статті, присвячені аналізу фінансових даних.

Огляд наукових публікацій пропонується розпочати зі статті «Система підтримки прийняття рішень для аналізу фінансових даних» [7]. Автори розглядають основні особливості проведення фінансового аналізу діяльнсоті підприємства і пропонують технологію аналізу фінансових даних, засновану на оригінальному інтегрованому підході. Основною метою даної статті є створення ймовірнісних моделей та інших типів моделей, щоб розробити загальну технологію для аналізу фінансово-економічних даних з використанням інформаційної системи підтримки прийняття рішень. Ця система базується на унікальному інтегрованому підході до проектування. За словами авторів, використання інформаційної СППР дозволить вирішувати завдання прогнозування розвитку фінансово-економічних процесів на підприємствах, а також здійснювати аналіз можливостей впровадження даної інформаційної технології для обробки даних. Це, в свою чергу, може дати змогу ефективно аналізувати і прогнозувати обсяги продажів компанії, а також оцінювати платоспроможність кредитних позичальників [7]. Запропонований авторами

підхід є доволі складним і потребує тривалої розробки математичного забезпечення для коректної роботи системи. У статті наведено алгоритм процедури інтегрованого підходу для прийняття рішень, для чого використовується мережа Байєса. Структура цієї мережі залежить від багатьох факторів, що враховуються під час прийняття рішень і у випадку, що розглядається у статті, має досить складну структуру. Архітектура інформаційної СППР у комерційному банку, наведена у статті, включає інтерфейс системи для менеджерів, операціоністів та аналітиків, базу даних клієнтів, підсистему звітів, підсистему моделювання та вихідні модулі, що забезпечують їхню взаємодію. В результаті проведення дослідження було отримано можливість системно підходити до побудови інформаційної СППР, призначеної для аналізу та прогнозування стану підприємства.

Інша наукова стаття «Система підтримки прийняття рішень у задачах фінансового аналізу» наводить багаторівневу структурну модель для фінансового аналізу з урахуванням динаміки зовнішнього та внутрішнього середовища [8]. Автори пропонують застосовувати для прийняття рішень математичний апарат нечітких множин. Також у статті визначено, що під час вирішення задач оцінювання фінансового стану підприємств, найбільшою проблемою є відсутність оптимальної системної моделі. Авторами розроблено розширену множину оцінювальних параметрів з метою підвищення точності оцінки. Стаття доводить, що використання систем підтримки прийняття рішень у сучасних підприємствах є однією з найважливіших умов їхньої ефективності.

Розглянуті вище статті демонструють важливість застосування систем підтримки прийняття рішень в економіці з метою підвищення ефективності діяльності підприємств, оцінки ризиків тощо. Запропоновані рішення використовують різноманітні підходи до моделювання та розробки систем підтримки прийняття рішень і дозволяють досягти більш ефективного управління підприємствами. Таким чином, можна зробити висновок про доцільність застосування систем підтримки прийняття рішень з метою автоматизації бізнес-процесів, а також про необхідність застосування

системного підходу до моделювання та розробки таких систем з урахуванням специфіки того чи іншого підприємства. Що стосується мереж магазинів, необхідно виокремити ті параметри, стосовно яких найважливіше застосовувати автоматизацію у прийнятті рішень. Прикладом можуть бути великі обсяги товарів та замовлень, коли відстежувати всі зміни вручну стає дедалі складніше зі зростанням обсягів продажу.

Також варто виокремити наукові публікації, що стосуються більш вузьких галузей застосування систем підтримки прийняття рішень. Таких статей опублікована досить велика кількість, як було зазначено раніше, проте галузь роздрібної торгівлі досліджена недостатньо. Розглянемо деякі з наявних статей, присвячених розробці спеціалізованих СППР. У статті «Система підтримки прийняття рішень (СППР) як інструмент управління рішеннями в галузі ІТ в умовах невизначеності» автором розглядається питання створення системи підтримки прийняття рішень та її застосування в діяльності ІТ-компаній [9]. Дослідження має на меті визначення особливостей розробки СППР для менеджера з управління проєктами з метою формалізації складних багатоальтернативних ситуацій. Іншим дослідженням, присвяченим вирішенню більш технічних завдань, є стаття «Автоматизація та системи підтримки прийняття рішень на поліграфічних підприємствах» [10]. Автори пропонують застосовувати системи підтримки прийняття рішень для автоматизації контролю й керування процесами друку. Крім цього, інші дослідники також пропонують застосовувати СППР в автоматизації технологічних процесів. Наприклад, відоме дослідження, присвячене розробці прецедентної СППР з метою діагностики мостових кранів [11]. Дослідники розробили структуру СППР, що використовує бібліотеку прецедентів, наповнення якої здійснюється експертом та на основі аналізу поточної інформації.

Отже, підсумовуючи проведений аналіз літературних джерел, можна зробити висновок про необхідність врахування особливостей сфери застосування розроблюваної СППР та досягнення найкращих результатів завдяки застосуванню спеціалізованих СППР для того чи іншого процесу.

Наступним етапом аналізу наявних рішень є розгляд програмного забезпечення, що використовується для підтримки прийняття рішень. Відоме програмне забезпечення також у більшості випадків є універсальним і не спрямованим на ту чи іншу галузь. Таке універсальне програмне забезпечення часто надає можливості для роботи з даними, їхнього зберігання, аналізу та спрощеного програмування правил прийняття рішень. В свою чергу спеціалізоване програмне забезпечення у більшості випадків розробляється індивідуально під потреби підприємства і може бути застосовано виключно для вирішення його індивідуальних завдань. Таке програмне забезпечення не отримує широкого розповсюдження, проте його застосування дозволяє досягти найкращих результатів, оскільки його моделі спрямовані на вирішення індивідуальних завдань того чи іншого підприємства.

Мережами магазинів для підтримки прийняття рішень може використовуватись достатньо велика кількість універсального програмного забезпечення. Серед них модна виділити SAP BusinessObjects, Microsoft Power BI, IBM Planning Analytics та Tableau. Всі ці програмні рішення мають великий набір інструментів для збору та аналітики великих обсягів даних. При цьому прийняття рішення виконується користувачем на основі отриманих результатів аналізу. Дані відображаються за допомогою графічного інтерфейсу в зручному для сприйняття форматі, проте автоматизоване прийняття рішень у випадку з мережею магазинів може бути дещо ускладненим.

SAP BusinessObjects Business Intelligence – це централізований пакет, призначений для звітування, візуалізації та спільного використання даних. Він перетворює дані на корисну інформацію, доступну в будь-який час і в будь-якому місці. Забезпечуючи гнучку архітектуру, аналітична платформа може підтримувати від кількох користувачів до десятків тисяч користувачів і від одного інструменту до кількох інструментів та інтерфейсів. Основними перевагами інструменту є: можливість розгортання на місці, бізнес-аналітика в режимі реального часу, підвищена автономність для користувача, персоналізована та динамічна обробка інформації [12].

Microsoft Power BI – це набір програмного забезпечення, призначеного для перетворення непов'язаних джерел даних на узгоджену, візуальну та інтерактивну статистику. Дані можуть бути електронною таблицею Excel або колекцією хмарних і локальних гібридних сховищ даних. Power BI дозволяє легко підключатися до своїх джерел даних, візуалізувати та виявляти важливі елементи, а також ділитися ними. Це програмне забезпечення має три різних версії: для настільних комп'ютерів, онлайн-служба, що використовує хмарні можливості, та мобільна версія для пристроїв з операційними системами Windows, Android та iOS. При цьому основна програма містить також два елементи: Power BI Report Builder, для створення розбитих на сторінки звітів для спільного використання в службі Power BI, і Сервер звітів Power BI – локальний сервер звітів [13]. Даний програмний пакет має ряд переваг, однією з яких є зручна інтеграція з Microsoft Excel, що дозволяє обробляти дані, використовуючи звичні для більшості користувачів офісні програми. Проте із цим пов'язані певні складнощі у випадку підтримки прийняття рішень мережею магазинів. Наприклад, користувач має виконувати значну кількість операцій вручну, а процес автоматизації може потребувати додаткових налаштувань, що ускладнює процес прийняття рішення. Аналогічно до цього програмного продукту, можна розглядати в якості засобу підтримки прийняття рішень звичайний Microsoft Excel, використовуючи умовне форматування, групування та формули для відображення порад щодо необхідності прийняття рішень. Проте цей варіант є менш зручним за інші, а програмне забезпечення не відноситься до інструментів для автоматизованої аналітики великих обсягів інформації.

IBM Planning Analytics пропонує прогнозувати результати за допомогою гнучкого прогнозування на основі штучного інтелекту. Інструмент також надає можливість роботи зі сценаріями типу «що-якщо» в режимі реального часу. Він також підтримує роботу з даними, отриманими з таблиць Excel, або за допомогою спеціального веб-інтерфейсу [14]. У випадку підтримки прийняття рішень інструмент може бути корисним, проте не може повністю замінити повноцінну систему підтримки прийняття рішень, оскільки більше орієнтований

на прогноз, а не на обробку виключно поточних даних. Це означає, що використання даного програмного рішення потребуватиме більш складного налаштування та подання вхідних даних.

Tabelau пропонує приймати рішення за допомогою аналітики з використанням штучного інтелекту. Програмне забезпечення являє собою хмарну платформу. Tabelau надає можливості з візуалізації даних, автоматизації за рахунок використання простих описів та персоналізації, відповідно до потреб користувача. Цей інструмент може ефективно використовуватися для аналітики великих обсягів інформації і орієнтований на прийняття рішень [15].

Також, окрім розглянутого програмного забезпечення, в одному зі згаданих раніше навчальних посібників [6] наведено перелік програмного забезпечення, що може бути використане для моделювання в системах підтримки прийняття рішень. В рамках даного дослідження пропонується зупинитись на вільному програмному забезпеченні. Розглянемо деякі з наведених у [6] програм:

– ADMB – надає можливості нелінійних методів оптимізації, використовуючи методи автоматичного диференціювання;

– ALGENCAN – Fortran-прогами нелінійної оптимізації, що працює з багатьма іншими мовами програмування;

– APMonitor – надає можливості змішаного цілочислового програмування, використовуючи мови MATLAB та Python;

– ASCEND – призначений для математичного моделювання систем;

– COBYLA – знаходить мінімум нелінійної функції з нелінійними обмеженнями типу нерівностей.

Завершуючи аналітичний огляд наявних рішень, можна зробити висновок про відсутність програмних розробок та необхідність розробки програмного забезпечення для підтримки прийняття рішень в мережах магазинів з урахуванням специфіки цієї галузі та без надлишкового функціоналу. Оскільки подібного програмного забезпечення, спрямованого на застосування у галузі роздрібної торгівлі, необхідно визначити задачі, відповідно до мети розробки програмного забезпечення та з урахуванням проведеного аналізу.

1.3 Постановка завдання

Для подальшого виконання роботи необхідно, спираючись на проведений аналіз предметної області дослідження, визначити задачі, які мають бути вирішені. Для цього необхідно зробити висновки про виявлені у попередніх підрозділах потреби мереж магазинів, недоліки існуючих рішень, особливості побудови систем підтримки прийняття рішень тощо. Отже, спираючись на цю інформацію, стає можливою постановка завдання.

Перш ніж перейти безпосередньо до визначення завдань, зазначимо, що метою роботи є розробка програмного рішення, яке автоматизує процеси збору, обробки та аналізу даних для прийняття найефективніших рішень. З урахуванням можливих змін, система повинна бути достатньо гнучкою, масштабованою та адаптивною для роботи в таких умовах. Тобто, система має працювати у випадках продажів різних товарів. Отже, основна мета розроблюваного програмного забезпечення – забезпечити зручний інтерфейс для менеджерів магазину для управління замовленнями та товарами.

В якості користувацького інтерфейсу зручно використовувати веб-інтерфейс, що дозволить отримувати доступ до системи з будь-якого пристрою у будь-який час. Також перевагою такого рішення є можливість зручної інтеграції з вебсайтом магазину, де може знаходитись користувацький інтерфейс оформлення замовлень. Основна функціональність має включати створення, редагування, перегляд і видалення замовлень клієнтів, а також управління товарами. Додаток повинен мати інтерактивний інтерфейс для менеджерів та адміністраторів. При цьому дані на сторінках мають оновлюватись динамічно, без перезавантаження сторінок, а сайт в цілому має бути адаптивним і коректно відображатись як на персональних комп’ютерах, так і на мобільних пристроях.

Сформулюємо основні функції розроблюваної системи підтримки прийняття рішень для мереж магазинів. Ці функції засновуються на потребах магазинів та особливостях наявних програмних рішень. Основні функції можна представити у вигляді списку:

1. Управління замовленнями:

– створення замовлень: користувач може створювати нові замовлення;

– редагування замовлень: користувач може редагувати існуючі замовлення через спеціальну форму;

– перегляд замовлень: система повинна надати інтерфейс для перегляду усіх замовлень з можливістю фільтрації за статусом;

– видалення замовлень: користувач може видаляти непотрібні замовлення;

2. Управління товарами:

– створення товарів: адміністратори можуть створювати нові товари з такими полями, як назва товару, опис, кількість та ціна;

– редагування товарів: адміністратори можуть редагувати інформацію про існуючі товари;

– перегляд списку товарів: адміністратори можуть переглядати наявні товари з можливістю сортування та фільтрації;

– видалення товарів: адміністратори можуть видаляти товари.

Отже, виходячи із зазначеного вище можна виділити основні завдання, які необхідно вирішити під час розробки нового програмного продукту для підтримки прийняття рішень:

– реалізувати функціонал додавання, редагування, перегляду та видалення замовлень;

– забезпечити валідацію даних як на клієнтській, так і на серверній стороні;

– підтримувати інтерактивне оновлення даних;

– забезпечити коректну роботу із зовнішніми базами даних для збереження товарів і замовлень.

Виходячи з визначеного функціоналу можна визначити приблизну структуру інтерфейсів користувачів системи підтримки прийняття рішень у мережі магазинів. Вирішення перелічених завдань передбачає розробку відповідних інтерфейсів, які будуть розглянуті в наступних розділах. До таких

інтерфейсів відносяться форми додавання, редагування та перегляду товарів, замовлень, користувачів та правил автоматизованого прийняття рішень системою.

Всі дані, що вводяться в систему за допомогою графічних інтерфейсів обов'язково мають проходити валідацію для забезпечення надійного функціонування системи.

Рішення системи носять рекомендаційний характер, а їх прийняття здійснюється користувачами системи. При цьому користувачі повинні мати, щонайменше, дві ролі – Адміністратор, що має повний доступ до всіх функцій додатку, та Менеджер, що може працювати тільки з замовленнями. В процесі подальшого розвитку системи можливе додавання також інших ролей.

Головні вимоги до програмного забезпечення: функціональність, адаптивність, безпека, продуктивність. Для забезпечення характеристик, відповідним цим вимогам, програмне забезпечення має використовувати спеціальні засоби розробки, що дозволяють досягти поставлених задач.

Таким чином, в результаті виконання першого розділу роботи було проведено аналітичний огляд предметної області дослідження, розглянуто питання прийняття рішень та поняття системи підтримки прийняття рішень, проаналізовано основні характеристики та потреби сучасних магазинів, а також їхній зв'язок з потребою у застосуванні систем підтримки прийняття рішень. Виконано аналіз наявних літературних джерел за темою дослідження та наукових праць, опублікованих у фахових виданнях. Обґрунтовано потребу в розробці системи підтримки прийняття рішень для мереж магазинів. Сформульовано задачі, що потребують вирішення в рамках роботи та вимоги до розроблюваної системи. Результатом виконання роботи має бути програмне забезпечення системи підтримки прийняття рішень для мереж магазинів.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Архітектура системи підтримки прийняття рішень

Перед початком розробки програмного забезпечення важливим етапом є моделювання. На цьому етапі визначається архітектура системи, проектуються діаграми, що відображають взаємозв'язки між елементами системи та відображають її роботу і взаємодію з користувачами. Для цього використовуються методи моделювання, що спираються на дані, отримані в результаті аналізу предметної області та постановці задачі.

Моделювання системи підтримки прийняття рішень пропонується виконати у три етапи. На першому буде визначено архітектуру системи, на другому побудовано діаграми, що детальніше відображатимуть структуру системи, її роботу та взаємодію з користувачами, а третій етап присвячено моделюванню бази даних, що використовуватиметься системою. Кожен з цих етапів може передбачати додаткові кроки. Наприклад, під час визначення архітектури системи або на етапі складання моделей може бути доцільно визначити її інформаційні процеси. Так само, на етапі моделювання бази даних необхідно визначити всі можливі елементи, що потребуватимуть зберігання.

Моделювання системи підтримки прийняття рішень може бути виконано як за функціональним, так і за об'єктно-орієнтованим підходом. В рамках виконання роботи пропонується використовувати поєднання цих підходів, залежно від кожної окремої частини системи. Такий підхід дозволяє раціонально розподілити час на виконання кожного окремого елементу системи та обрати для нього оптимальний підхід моделювання та розробки. Тому в роботі доцільно навести визначення для обох підходів, що можуть бути використані в тій чи іншій ситуації.

Моделювання – це процес створення абстрактного представлення реального об'єкта, системи або процесу з метою його аналізу, вивчення, прогнозування тощо. Завдяки моделюванню можна наочно представляти системи та процеси. Моделі можуть бути фізичними, математичними або комп'ютерними.

Архітектура системи підтримки прийняття рішень керівництвом мережі магазинів визначається в першу чергу тим, що це має бути веб-додаток. Отже, архітектура системи має відповідати типовій архітектурі для вебсайтів. Такою архітектурою програмного забезпечення є клієнт-серверна архітектура.

Клієнт-серверна архітектура – один з основних існуючих шаблонів архітектури програмного забезпечення. Ця концепція часто використовується і навіть є домінуючою під час розробки розподілених мережевих додатків, наприклад, таких як вебсайти. Така архітектура передбачає обов'язкову можливість взаємодії між різними компонентами системи завдяки обміну даними. Основними складовими клієнт-серверної архітектури виступають сервер, що надає послуги або ресурси; клієнт, що звертається до сервера для отримання цих послуг або ресурсів; та мережа, що забезпечує комунікацію та взаємодію між ними.

Клієнт – це звичайний персональний комп'ютер, що знаходиться на стороні користувача і передає запити на сервер з метою отримання інформації або виконання певних операцій. Сервер – це значно потужніший комп'ютер або спеціалізоване (серверне) обладнання, розроблене для вирішення певних задач, пов'язаних з виконанням програмного коду та/або наданням сервісних функцій за запитами, що надходять від клієнтів, забезпечення користувачів доступом до необхідних ресурсів та зберігання інформації в базах даних [16].

Розроблювана система підтримки прийняття рішень функціонує за принципом постійної взаємодії клієнта і сервера. Клієнт за певних умов ініціює запити на сервер, він обробляє запит та повертає результат обробки назад клієнту для відображення у користувацькому інтерфейсі. Сервер здатен обробляти багато запитів від кількох клієнтів паралельно. У випадку одночасного надходження більшої кількості запитів, ніж один, з них утворюється чергу, а їхня обробка здійснюється сервером послідовно. Деякі запити від клієнтів можуть мати вищий пріоритет, і в цьому випадку вони обробляються в першу чергу [16].

На стороні сервера виконуються наступні функції: надання доступу, зберігання, резервне копіювання та захист даних; виконання обробки

клієнтських запитів; передача результатів (відповідей) на сторону клієнта. Тим часом на стороні клієнта виконуються наступні функції: надання графічного користувацького інтерфейсу; формування запитів та їх відправлення на сервер; отримання результатів виконання запитів від сервера та відправлення додаткових команд (наприклад, запитів на оновлення, додавання або видалення даних) [16]. Додаткові команди необхідні для забезпечення умови динамічного оновлення сторінок без їх повного перезавантаження, таким чином зручно реалізовувати взаємодію з базою даних.

Архітектурою «клієнт-сервер» визначаються основні принципи взаємодії між комп'ютерами, в той час як правила та порядок їхньої взаємодії визначаються протоколом. Мережевий протокол – це деякий набір правил, згідно з якими здійснюється взаємодія між пристроями (комп'ютерами), що знаходяться в мережі. На сьогодні існують різні мережеві протоколи, кожен з яких має свої особливості та призначення. Найбільш поширеними і часто застосовуваними у програмному забезпеченні протоколами є TCP та IP (або TCP/IP, що включає поєднання цих протоколів), MAC, ICMP, UDP, HTTP, HTTPS, FTP, SSH, POP3, SMTP, IMAP та інші [16].

У літературі визначено дві основні концепції, що застосовуються для побудови клієнт-серверних систем [16]:

1. Слабкий клієнт – потужний сервер. У даній моделі обробка даних повністю здійснюється на стороні сервера, тоді як клієнт має обмеження у правах доступу. Сервер передає клієнту відповідь, що вже не потребує додаткової обробки. Клієнт через графічний інтерфейс здійснює взаємодію з користувачем, формує запити, передає їх серверу, отримує результат та відображає отриману інформацію на екрані.

2. Сильний клієнт – це концепція, яка передбачає, що деяка частина обробки інформації здійснюється на клієнтському пристрої. У цьому випадку сервер виступає в основному як сховище даних, тоді як вся робота з обробкою та відображенням інформації відбувається на комп'ютері або іншому пристрої клієнта.

Клієнт-серверну архітектуру можливо класифікувати за рівнями. Можна виділити дворівневу та трирівневу клієнт-серверну архітектуру.

Дворівнева клієнт-серверна архітектура має в своєму складі два вузли [16]:

– сервер, що здійснює отримання запитів від клієнта і відправлення йому відповідей, задіюючи при цьому виключно свої власні ресурси;

– клієнт, що здійснює представлення графічного інтерфейсу користувача.

Принцип роботи дворівневої клієнт-серверної архітектури заснований на тому, що сервер має отримати від клієнта запит, виконати його обробку та відповісти безпосередньо, не використовуючи при цьому сторонні ресурси.

Трирівнева клієнт-серверна архітектура є дещо складнішою і складається з таких трьох компонентів [16]:

– представлення даних – графічний інтерфейс, призначений для взаємодії клієнта з користувачем системи;

– прикладний компонент – основний сервер, призначений для взаємодії з клієнтом та сервером бази даних;

– керування ресурсами – сервер бази даних, призначеної для отримання та збереження інформації.

Принцип роботи трирівневої клієнт-серверної архітектури полягає в тому, що запит, отриманий від клієнта, можуть обробляти декілька серверів. При цьому розподіл операцій дозволяє знизити навантаження сервера.

Вирішення задач, поставлених в даній роботі, базується на клієнт-серверній архітектурі. На стороні клієнта відбувається відображення графічного інтерфейсу користувача, отримання команд від користувача (наприклад, шляхом натискання на елементи інтерфейсу), формування запитів на сервер, отримання відповідей та їхнє виведення в користувацький інтерфейс. На стороні сервера здійснюється обробка запитів, отриманих від клієнта, формування запитів до системи керування базами даних, отримання необхідної інформації, формування відповіді клієнту, та відправлення сформованого пакету на сторону клієнта для подальшої обробки. Такий підхід дозволяє обробляти запити від декількох

клієнтів одночасно, забезпечуючи для користувачів швидкий та зручний доступ до інформації про товари, замовлення, користувачів тощо. Система розроблятиметься за принципом слабкий клієнт – потужний сервер та матиме дворівневу архітектуру. Таку систему можна зобразити схематично (рис. 2.1), розділивши серверну частину і базу даних, як окремі складові.

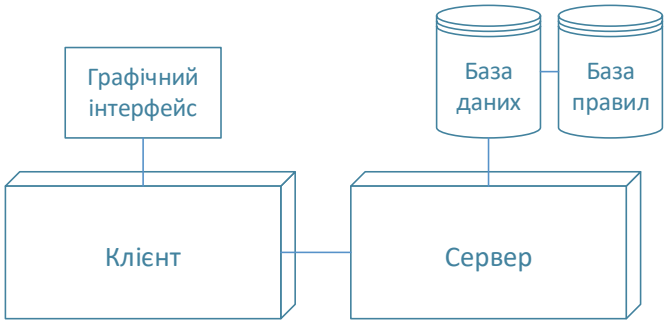

Рис. 2.1 Схема клієнт-серверної архітектури системи

Отже, розроблювана система підтримки прийняття рішень для мережі магазинів буде складатися з двох основних частин: клієнта і сервера, при чому сервер має також надавати можливості роботи з базою даних. Такий підхід є оптимальним, коли кількість запитів від клієнтів є порівняно невисокою і потреба у розподілі серверів відсутня. Також перевагою такого рішення є спрощення серверного програмного забезпечення.

Таким чином, система підтримки прийняття рішень керівництвом мережі магазинів надає користувачам графічний інтерфейс, що знаходиться на стороні клієнта (відображається у браузері), користувачі можуть взаємодіяти з елементом інтерфейсу, відправляючи при цьому запити на сторону сервера. Сервер обробляє отримані запити, звертається до бази даних, що знаходиться на цьому ж сервері, формує відповідь та відправляє її клієнту. Клієнт, отримавши відповідь від сервера, відображає її у графічному інтерфейсі.

Уточнюючи структуру системи, відносно класичної структури системи підтримки прийняття рішень, розглянутої в попередньому розділі, варто зауважити, що база моделей та база даних в даному випадку може бути представлена однією спільною базою даних (для інформації та умов прийняття рішень) та серверним програмним забезпеченням, що реалізує виконання умов прийняття рішень.

Після визначення архітектури системи, її необхідно доповнити аналізом інформаційних потреб та деталізованим оглядом функціоналу з метою переходу до наступних етапів моделювання. Важливо враховувати, що система має клієнт-серверну архітектуру, а значить функціонал має бути адаптований під цю архітектуру. На даному етапі нема сенсу детально оглядати інтерфейс користувача, але необхідно визначити, з якими даними відбувається робота, що передає клієнт, що повертає сервер тощо.

Одним з найперших етапів у моделюванні є проведення аналізу інформаційних потреб. На основі цих потреб визначається точний функціонал системи та формуються моделі для подальшої розробки програмного забезпечення. Як було зазначено під час постановки задачі, система повинна забезпечити автоматичний вибір оптимальних рішень щодо наявних товарів, виходячи з наявних даних, та пропонувати користувачу прийняти рішення. При цьому менеджер може працювати тільки з замовленнями (додавати, редагувати, видаляти), а адміністратор може виконувати ці операції щодо користувачів, товарів, замовлень, правил прийняття рішень тощо. Це означає, що важливою функцією системи є розподіл ролей користувачів, а значить необхідна реалізація алгоритмів авторизації. Для зручності користувачів доцільно забезпечити можливості пошуку та фільтрації товарів та замовлень. Користувачі також повинні мати доступ до інструкції щодо використання системи, а головне меню має надавати можливості навігації основними розділами системи.

Інформаційні потреби системи тісно пов’язані з її функціоналом, який визначає, які дані мають передаватись між клієнтом, сервером та базою даних і в якому напрямку. Аналіз інформаційних потреб дозволяє деталізувати роботу системи підтримки прийняття рішень.

Функціонування системи підтримки прийняття рішень керівництвом мережі магазинів передбачає виконання певних операцій, пов’язаних з обліком товарів та обробкою замовлень. Кожна з цих операцій являє собою окрему складову загального функціоналу автоматизованої системи підтримки прийняття рішень. Розглянемо детальніше основні операції, що можуть виконуватись у системі.

1. Авторизація користувача.
2. Відображення списку товарів.
3. Відображення списку замовлень.
4. Відображення списку користувачів.
5. Відображення списку правил прийняття рішень.
6. Додавання нового користувача.
7. Додавання нового товару.
8. Додавання нового замовлення.
9. Додавання нового правила прийняття рішень.
10. Редагування існуючого користувача.
11. Редагування існуючого товару.
12. Редагування існуючого замовлення.
13. Редагування існуючого правила прийняття рішень.
14. Видалення існуючого користувача.
15. Видалення існуючого товару.
16. Видалення існуючого замовлення.
17. Видалення існуючого правила прийняття рішень.
18. Пошук товару за назвою.
19. Пошук замовлення за назвою товару, іменем або E-mail замовника.
20. Пошук тільки замовлень в обробці.
21. Пошук завершених та відхилених замовлень.
22. Прийняття рішення щодо товару.
23. Скасування рішення щодо товару.

Всі операції з наведеного вище переліку можна умовно розділити на п’ять основних категорій: відображення, додавання, редагування, видалення та пошук. Значна кількість функцій є подібною. При цьому варто відзначити, що наведений список не є вичерпним і може бути розширений в процесі вдосконалення системи. Наприклад, в процесі подальшої розробки до системи може бути додано можливості додавання, редагування та видалення ролей користувачів, статусів замовлень, рішень щодо товарів тощо.

Розглянемо детальніше функціонал системи в рамках перелічених 22 пунктів списку. Їхня реалізація є достатньою умовою для досягнення поставлених задач. Можливі доповнення в процесі подальшого розвитку проєкту можна буде спрямувати на розширення можливостей користувачів, додавання нових ролей тощо. Проте в рамках даної роботи розглядатимуться лише перелічені можливості системи.

Авторизація користувача є першим, що необхідно виконати для отримання доступу до повного функціоналу системи. Для цього користувач має ввести логін та пароль. Сервер приймає аутентифікаційні дані, перевіряє існування користувача з таким логіном, перевіряє правильність введення паролю і створює сесію для користувача з відповідною роллю. Логіном може бути будь-яка послідовність символів латиниці та цифр, а пароль може приймати будь-які символи і є чутливим до регістру. Це дозволяє підвищити безпеку.

Для відображення списку товарів, замовлень, користувачів або правил прийняття рішень, користувачу необхідно пройти авторизацію та перейти у відповідний розділ. При цьому сервер за сесією користувача визначає наявність обмежень прав.

Для додавання нового користувача необхідно ввести його логін, пароль, E-mail, ПІБ та обрати роль. Це означає, що ролі мають бути занесені до бази даних заздалегідь, оскільки відсутність ролей не дозволить додавати нових користувачів. За замовчуванням система має дві доступні для користувачів ролі: Адміністратор та Менеджер. Редагування даних користувача здійснюється шляхом редагування тих самих параметрів. Для видалення користувача система використовує його ідентифікатор (прихований від користувачів), введення додаткових даних не потрібне.

Для додавання нового товару адміністратор має ввести назву товару, його опис, ціну, кількість, вставити посилання на зображення, обрати дату додавання і дату останнього продажу товару (не обов’язково, може бути заповнено автоматично), а також вибрати рішення. Це означає, що додавання товару потребує попереднього додавання рішень заздалегідь. Редагування товару

здійснюється шляхом зміни перелічених даних та збереження оновлень. Видалення товару, як і у випадку з користувачем, здійснюється за ідентифікатором і не потребує введення додаткових даних.

Додавання нового правила прийняття рішень здійснюється адміністратором. Для цього необхідно ввести умову (логічний вираз в форматі системи керування базами даних), опис та рішення. При цьому рішення мають бути внесені до бази даних заздалегідь. Видалення правила прийняття рішення здійснюється за ідентифікатором і не потребує введення додаткових даних.

Додавання нового замовлення може бути здійснено як адміністратором, так і менеджером. Для цього необхідно обрати товар, ввести кількість, дату замовлення, ПІБ замовника, E-mail замовника, адресу доставки та обрати статус замовлення. Отже, як і в попередніх випадках, статус замовлення має бути внесений до бази даних заздалегідь. Це означає потребу в попередньому налаштуванні системи одразу після завершення її розробки та введення в експлуатацію. Редагування замовлення передбачає зміну перелічених вище відомостей про замовлення та збереження внесених змін. Видалення замовлення потребує тільки ідентифікатора замовлення, що призначається системою автоматично, введення додаткових даних для цього не потрібно.

Для виконання пошуку товарів виконується запит таблиці товарів, але з параметром пошуку. Пошук замовлень здійснюється аналогічно. Для товарів пошук здійснюється виключно за назвою товару, а для замовлень – за назвою товару, за ПІБ або E-mail замовника. Пошукові дані не обов’язково вводити дослівно, система знаходить всі схожі записи, в яких зустрічається пошукова послідовність. Відображення замовлень за критеріями (наприклад, тільки завершених) здійснюється аналогічно до пошуку, тільки замість пошукового запиту на сервер передається ідентифікатор критерію. Ця операція не потребує введення запитів або попереднього створення додаткових таблиць у базі даних.

Отже, враховуючи необхідність автоматизації процесу підтримки прийняття рішень, необхідно передбачити організоване зберігання даних і простий, інтуїтивно зрозумілий графічний інтерфейс користувача.

2.2 Розробка UML-діаграм клієнтської та серверної частини системи

Важливим етапом моделювання будь-якої системи є розробка діаграм, що відображають структуру системи, взаємодію її елементів між собою та з користувачем, її архітектуру тощо. Для цього використовуються графічні засоби моделювання, у яких кресляться відповідні схеми. В результаті такого моделювання можна отримати діаграми, на основі яких буде виконуватись розробка програмного забезпечення.

На даному етапі пропонується розглянути діаграму прецедентів використання, діаграму послідовності, діаграму впровадження (що відображає структуру системи). Також необхідно визначити структуру бази даних та представити її графічно у вигляді моделі. В даному випадку база даних виконує одну з найважливіших функцій – зберігає всі дані та правила прийняття рішень.

Завершення аналізу інформаційних процесів, що відбуваються в автоматизованій системі підтримки прийняття рішень дозволяє деталізувати функціонал клієнтської та серверної частини. Для цього зручно використовувати UML-діаграми, які дозволяють наочно відобразити взаємодію системи і користувача та елементів системи між собою.

UML-діаграми (Unified Modeling Language Diagrams) представляють собою засіб візуального моделювання, який сприяє розробникам програмного забезпечення у концептуалізації, проектуванні та втіленні складних систем. Вони включають декілька типів, що широко використовуються: діаграма класів, діаграма відношень, діаграма варіантів використання, діаграма діяльності, діаграма станів, діаграма послідовності, діаграма компонентів та діаграма впровадження.

В рамках виконання даної роботи зроблено висновок про доцільність виконання діаграми прецедентів використання, діаграми впровадження та діаграми послідовності. Це дозволить наочно та у зручній формі представити відношення між користувачем та елементами системи.

Діаграма прецедентів використання (Use-Case Diagram) використовується для моделювання функціональності системи з точки зору взаємодії користувачів з нею. Основна мета цієї діаграми – показати, як актори, тобто користувачі або інші системи, спілкуються з системою, які функції вона виконує для них та як вони взаємодіють між собою.

Діаграма впровадження (Deployment Diagram) в UML використовується для моделювання фізичної архітектури системи. Основна її мета – відобразити розміщення компонентів програмного забезпечення та апаратних засобів у різних фізичних середовищах, таких як сервери, вузли мережі та пристрої.

Діаграма послідовності (Sequence Diagram) ілюструє взаємодії між об’єктами в хронологічному порядку. Вона надає інформацію про участь об’єктів та порядок обміну повідомленнями.

За допомогою цих трьох діаграм можна зробити висновки про структуру системи, її функціонал та взаємодію елементів між собою.

Крім цього, необхідною також є побудова діаграм, що відображають взаємозв’язки між елементами системи, їхню взаємодію. Оскільки було прийняте рішення про реалізацію бази даних та бази моделей спільно, важливо також розглянути структуру системи підтримки прийняття рішень для мереж магазинів. Така структурна схема наведена на рис. 2.2.

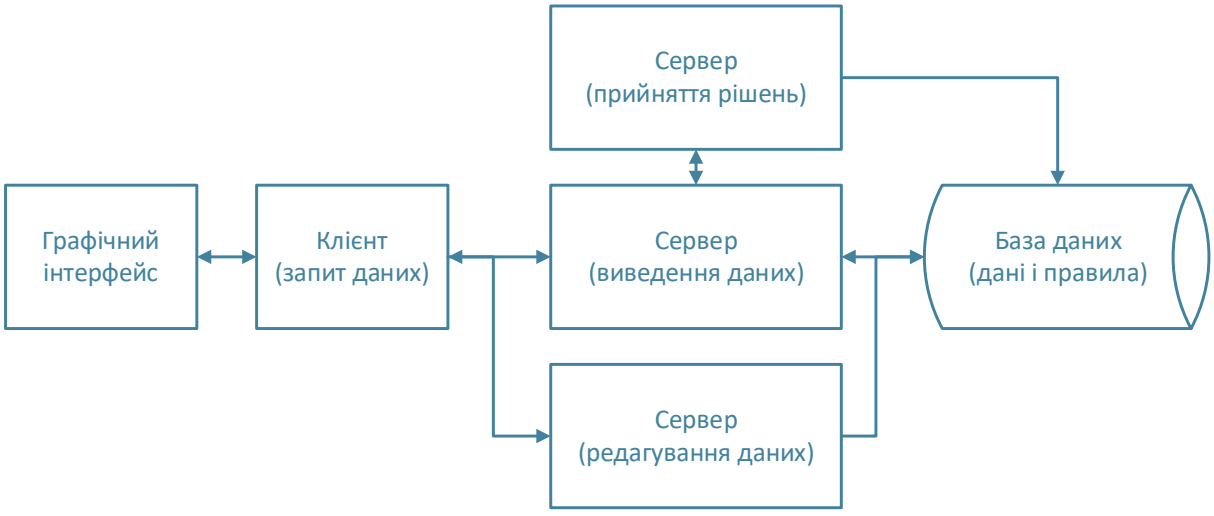

Рис. 2.2 Структурна схема системи

Згідно з наведеною схемою в системі відбувається взаємодія елементів зі сторони клієнта та сервера (включаючи базу даних).

Результати аналізу інформаційних процесів дозволяють зробити висновок про потреби користувача та створити діаграму прецедентів використання (рис. 2.3). Вона відображає основні функціональні блоки програмного забезпечення, що повинні бути виконаними.

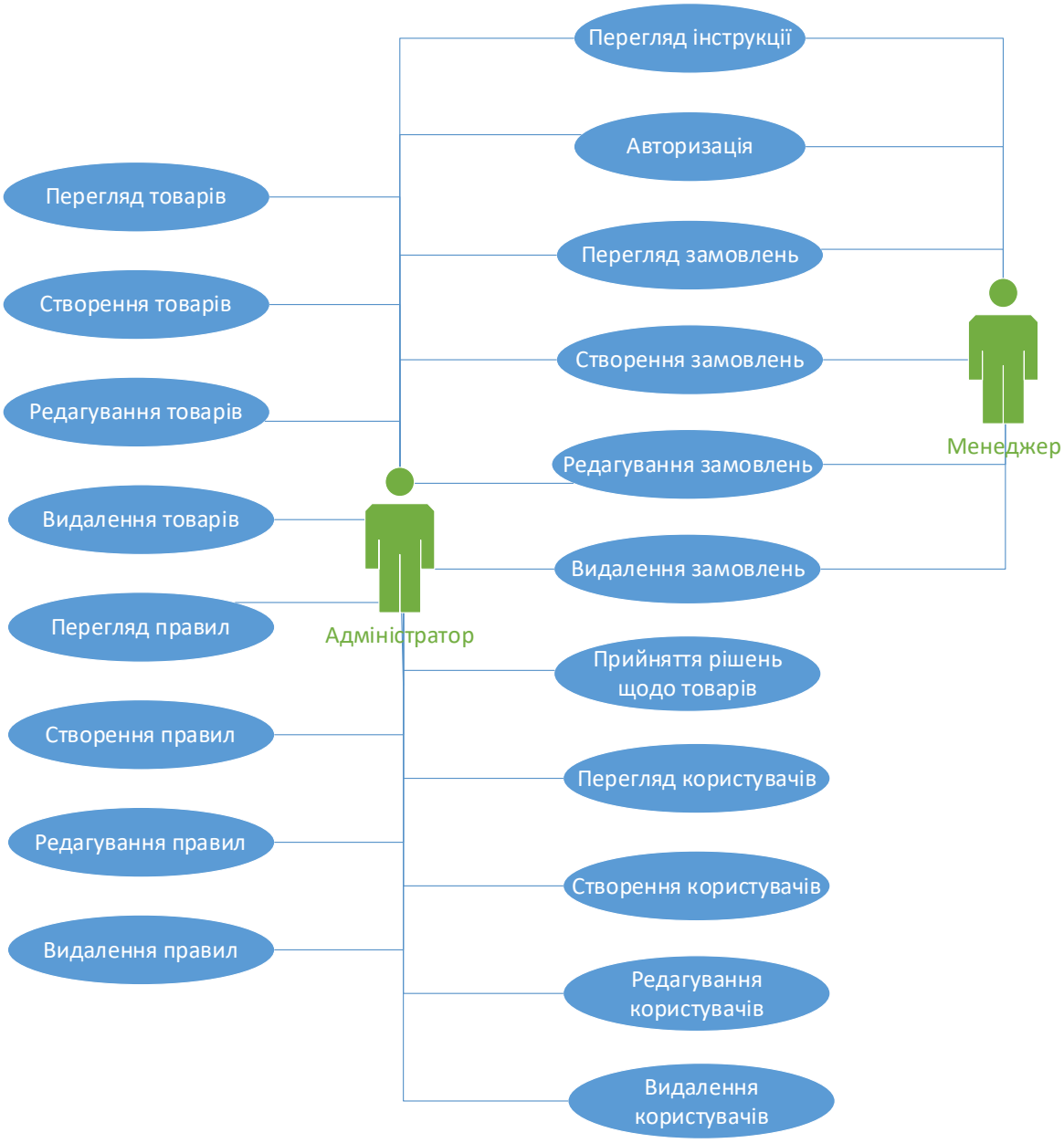

Рис. 2.3 Діаграма прецедентів використання

Згідно з побудованою діаграмою можна зробити висновок про функціонал системи для різних користувачів та побачити наочно відмінності між ролями. Всі користувачі системи можуть переглядати інструкцію, проходити авторизацію,

працювати із замовленнями. А весь інший функціонал системи доступний тільки Адміністратору.

На діаграмі впровадження (рис. 2.4) зображуються програмно-апаратні компоненти системи, що розробляється. В клієнт-серверній архітектурі виділяється частина клієнта, частина сервера та частина сервера бази даних, що відповідним чином відображається на структурі програмного забезпечення.

Рис. 2.4 – Діаграма впровадження

Як видно з діаграми впровадження, дані обробляються клієнтом, який взаємодіє з сервером, що здійснює роботу з базою даних. Така структура системи дозволяє виконувати прийняття рішень на основі аналізу даних, не використовуючи надлишкові зовнішні ресурси.

Крім цього, в рамках роботи доцільно скласти діаграму послідовності. Розглянемо діаграму послідовності для процесу відображення даних в інтерфейсі користувача. Розроблена діаграма послідовності наведена на рис. 2.5.

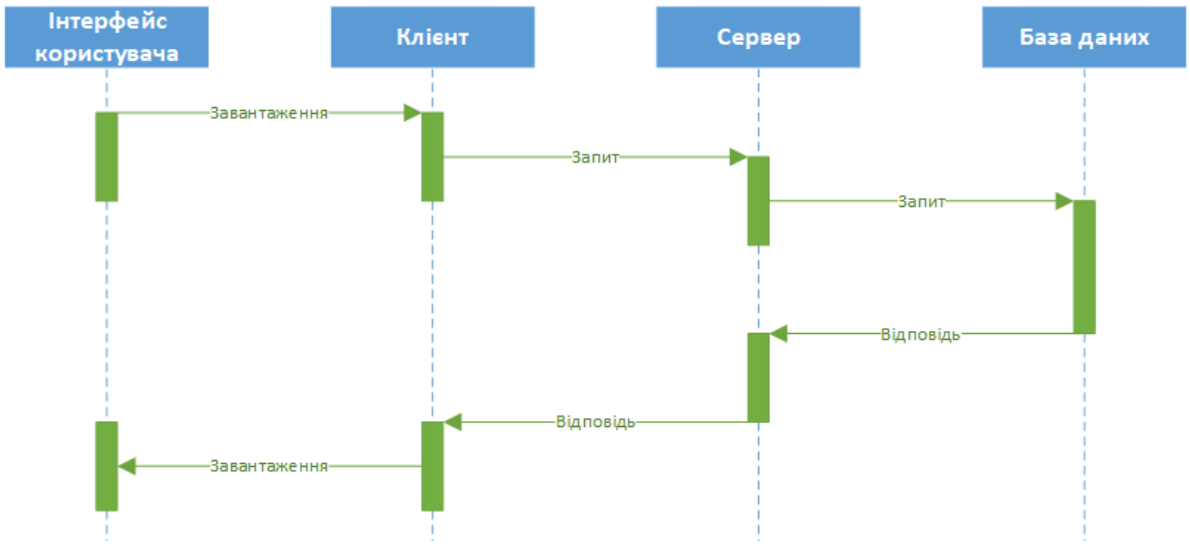

Рис. 2.5 Діаграма послідовності

З діаграми видно, що завантаження ініціюють елементи графічного інтерфейсу користувача, після чого програмне забезпечення клієнта формує та відправляє запит на сервер, який в свою чергу формує запит до бази даних. Від

бази даних сервер отримує відповідь, що містить запитані користувачем дані, формує відповідь клієнту, а клієнт виводить дані у графічний інтерфейс.

2.3 Моделювання бази даних

Моделювання бази даних системи підтримки прийняття рішень доцільно розпочати з формулювання концептуальних вимог користувачів до бази даних. Це можливо зробити за допомогою концептуальних запитів. Вивчення потреб користувачів та аналіз доступної інформації дозволяє сформувати такі концептуальні запити:

1. Вивести інформацію про користувачів (логін, E-mail, ПІБ, роль).
2. Вивести інформацію про товари (назва, опис, ціна, зображення, кількість, дата додавання, дата останнього замовлення, рішення).
3. Вивести інформацію про замовлення (ID, ID товару, назва товару, кількість, дата замовлення, ПІБ замовника, E-mail, адреса доставки, статус).
4. Вивести інформацію про правила автоматизованого прийняття рішень (умова, опис, рішення).
5. Вивести інформацію про товари (назва, опис, ціна, зображення, кількість, дата додавання, дата останнього замовлення, рішення) за заданою назвою товару.
6. Вивести інформацію про замовлення (ID, ID товару, назва товару, кількість, дата замовлення, ПІБ замовника, E-mail, адреса доставки, статус) за заданою назвою товару, іменем або E-mail замовника.
7. Вивести інформацію про замовлення (ID, ID товару, назва товару, кількість, дата замовлення, ПІБ замовника, E-mail, адреса доставки, статус), якщо його поточний статус – «в обробці».
8. Вивести інформацію про замовлення (ID, ID товару, назва товару, кількість, дата замовлення, ПІБ замовника, E-mail, адреса доставки, статус), якщо його поточний статус завершений.

Наведений перелік не є вичерпним і може розширюватись, проте описаного достатньо для демонстрації основного функціоналу системи, оскільки це

мінімально необхідний набір функцій. Головна задача – відображення даних, отриманих з бази та перевірених системою підтримки прийняття рішень.

Етап інфологічного моделювання включає формування категорій предметної області. Категорії предметної області будуються на основі запитів користувача, але при цьому враховують зв'язки між сутностями, тому в описі однієї сутності можуть бути присутні інші, що характеризують її. На мові інфологічного моделювання основні категорії системи підтримки прийняття рішень можуть бути подані в такому вигляді:

1. Роль (ID, роль).
2. Рішення (ID, рішення).
3. Статус замовлення (ID, статус).
4. Користувач (ID, логін, пароль, E-mail, ПІБ, роль).
5. Товар (назва, опис, ціна, зображення, кількість, дата додавання, дата останнього замовлення, рішення).
6. Замовлення (ID, товар, кількість, дата замовлення, ПІБ замовника, E-mail, адреса доставки, статус).
7. Правила прийняття рішень (умова, опис, рішення).

Отже, для опису бази даних системи підтримки прийняття рішень керівництвом мереж магазинів необхідні такі сутності: роль, рішення, статус замовлення, користувач, товар, замовлення і правила прийняття рішень. Співвідношення між цими сутностями зручно представити у вигляді ER-моделі.

ER-модель (Entity-Relationship Model) є методологією моделювання баз даних, яка служить для опису структури та взаємозв'язків між даними в системі. Використання ER-моделі дозволяє абстрагувати ключові аспекти даних та їх відносини, щоб створити зрозумілу та легко адміністровану базу даних. В рамках ER-моделі бази даних виокремлюються кілька основних понять, зокрема: сутності, відношення та атрибути. Цю модель можна представити у вигляді діаграми, що ілюструє структуру та взаємозв'язки між сутностями бази даних.

ER-модель є важливим інструментом для розробки баз даних, оскільки вона дозволяє розуміти структуру даних і відносини між ними ще на етапі

проектування, що сприяє ефективній розробці та підтримці бази даних у майбутньому.

При подачі предметної області за допомогою ER-моделі за правилами побудови концептуальних схем сутності зображуються у вигляді прямокутників, асоціації – ромбів, зв'язки – ненаправлених ребер, над якими може проставлятися ступінь зв'язку та необхідне пояснення.

Представлення ER-моделі для бази даних системи підтримки прийняття рішень керівництвом мереж магазинів зображено на рисунку 2.6.

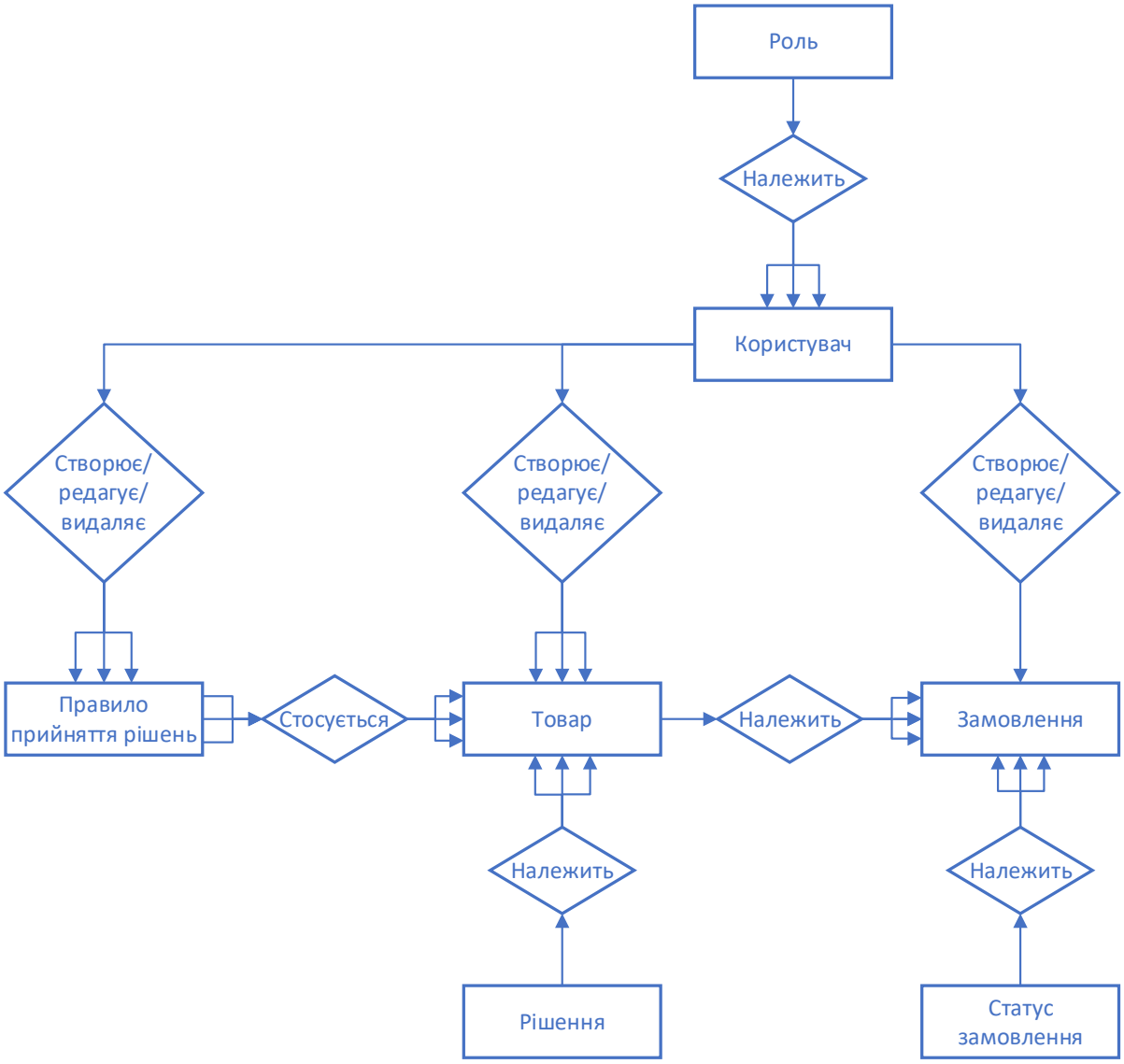

Рисунок 2.6 – ER-модель бази даних

Як видно з діаграми, користувачу належить роль, користувач може створювати, редагувати (в тому числі, приймати рішення) та видаляти товари, замовлення та правила прийняття рішень. Замовлення мають статус, товари

мають прийняті рішення. Правила прийняття рішення стосуються товарів. Товар може належати до замовлення. Таким чином взаємодіють елементи бази даних.

Проаналізувавши діаграму ER-моделі бази даних системи ідтримки прийняття рішень, можна зробити висновок про характеристики зв'язків бази даних. Для цього зручно використовувати таблицю, до якої вносяться сутності і опис їхніх зв'язків. Характеристики зв'язків для бази даних СППР представлено в таблиці 2.1.

Таблиця 2.1 – Характеристики зв'язків бази даних

Ім'я сутності 1	Ім'я сутності 2	Тип зв'язку	Ім'я зв'язку	Клас належності
Роль	Користувач	1:Б	Належить	Обов'язковий
Рішення	Товар	1:Б	Належить	Обов'язковий
Статус замовлення	Замовлення	1:Б	Належить	Обов'язковий
Правило прийняття рішень	Товар	Б:Б	Стосується	Обов'язковий
Користувач	Замовлення	1:Б	Створює/редагує/ видаляє	Обов'язковий
Користувач	Товар	1:Б	Створює/редагує/ видаляє	Обов'язковий
Користувач	Правило прийняття рішень	1:Б	Створює/редагує/ видаляє	Обов'язковий
Товар	Замовлення	1:Б	Належить	Обов'язковий

З таблиці видно, що більшість зв'язків у базі даних являють собою зв'язки типу один до багатьох. Це означає що, наприклад, один користувач може створити багато замовлень або товарів, а один статус можуть мати декілька замовлень. Беручи до уваги клієнт-серверну архітектуру, що розробляється,

варто відмітити, що всі алгоритм роботи з даними та їх обробки стосується серверної частини.

Наступним етапом моделювання бази даних є проєктування її структури з урахуванням отриманої ER-моделі. Для цього необхідно розглянути питання нормалізації бази даних та виконати її нормалізацію.

Визначення функціональних залежностей та оптимізація їх подання дозволяє створити найефективнішу структуру бази даних, яка забезпечує надійне зберігання та обробку даних. Це досягається через використання методів еквівалентних перетворень схем реляційної бази даних. Процес, що охоплює ці дії, називається нормалізацією, і він передбачає трансформацію об’єктів інформаційної моделі в логічні таблиці бази даних [17].

Основні принципи, що регулюють нормалізацію, включають [17]:

– первинні ключі повинні бути максимально мінімальними;

– кількість відношень в базі даних має бути такою, щоб зменшити надмірність даних, що підвищує надійність системи;

– число відношень не повинно впливати на продуктивність;

– уникнення конфліктів даних при виконанні операцій додавання, видалення або оновлення є критично важливим;

– схема відношень повинна бути стійкою і гнучкою, щоб адаптуватися до змін;

– час реакції на запити має бути стабільним;

– дані повинні точно відображати стан бази даних у кожен момент часу.

Розробка системи, яка відповідає всім цим критеріям, є складним завданням, яке часто не має однозначного вирішення. Теорія функціональних залежностей встановлює важливі правила для структурування таблиць у реляційних базах даних, які реалізуються через нормальні форми. Кожна з цих форм пов'язана з певними типами функціональних залежностей у відношеннях. Декомпозиція таблиць на окремі елементи є одним із способів уникнення конфліктів даних,

оскільки кожен компонент може мати лише одну функціональну залежність, що забезпечує цілісність даних [17].

Процес усунення потенційних конфліктів та надмірностей в реляційних базах даних також називається нормалізацією схем відношень. Він включає декомпозицію відношень, що відповідають попередній нормальній формі, на два або більше нових відношень, що відповідають наступним нормальним формам.

В даному випадку проведено нормалізацію бази даних до третьої нормальної форми, що означає врахування вимог першої, другої та третьої нормальних форм. Відношення вважається у третій нормальній формі, якщо воно відповідає вимогам другої нормальної форми, і всі неключові атрибути залежать виключно від первинного ключа. Це означає, що третя нормальна форма забороняє наявність транзитивних залежностей між неключовими атрибутами та ключем [17].

Отже, розглянемо загальну структуру бази даних і зв’язки між всіма таблицями окремо, наведемо їх детальний опис з урахуванням всіх співвідношень. Для цього необхідно створити базу даних з назвою StoreWebApp. У цій базі будуть зберігатись таблиці, відповідно до розглянутих раніше структур. Для відображення співвідношень між таблицями зручно використовувати стандартні засоби, вбудовані до програмного забезпечення для роботи з системами управління базами даних. Оскільки список таблиць вже розглядався раніше, перейдемо одразу до розгляду кожної таблиці окремо та її зв’язків з іншими таблицями у вигляді графіків. Почерговий розгляд кожної таблиці дає можливість детально дослідити структуру бази даних.

Таблиця roles призначена для зберігання інформації про ролі користувачів. Вона має два поля: ідентифікатор ролі та назва ролі. Ідентифікатор ролі користувача дозволяє пов’язувати роль з користувачем. Це ціле число типу INT. Ідентифікатор має бути створений з параметром AUTO_INCREMENT, що необхідно для коректної роботи бази даних. Назва ролі – поле типу VARCHAR. Максимальна довжина рядка з назвою ролі становить 50 символів. Цього достатньо для більшості стандартних випадків, в тому числі для збереження

ролей «Адміністратор» та «Менеджер». Зв’язки таблиці roles з іншими таблицями представлені на рисунку 2.7.

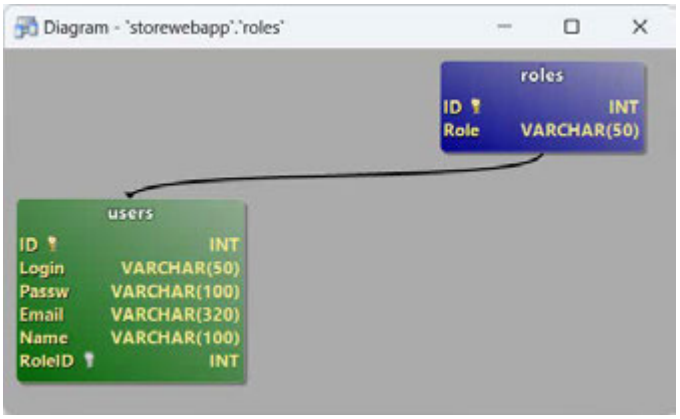

Рис. 2.7 Співвідношення таблиці roles з іншими таблицями

Аналогічними за своєю структурою є таблиці acts та status, що зберігають рішення та статуси замовлень. Такі дані зручно зберігати в окремих таблицях тому, оскільки це дозволяє не набирати щоразу вручну рішення та статус товару, використовувати стандартні рішення і статуси для всіх користувачів системи. Отже, таблиця acts наведена на рис. 2.8, а таблиця status наведена на рис. 2.9.

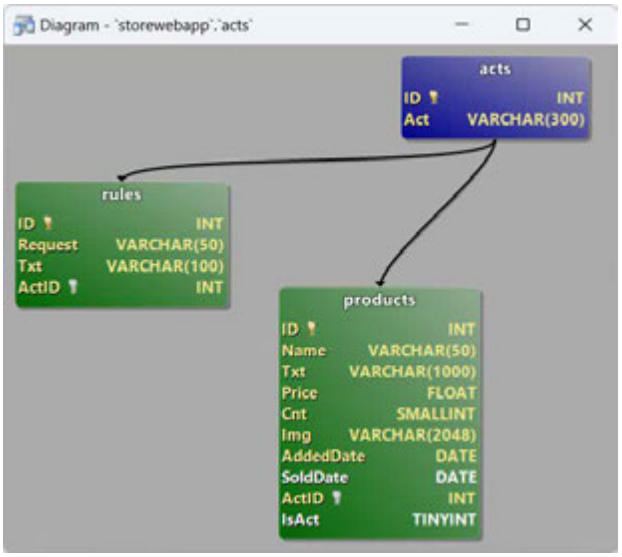

Рис. 2.8 Співвідношення таблиці acts з іншими таблицями

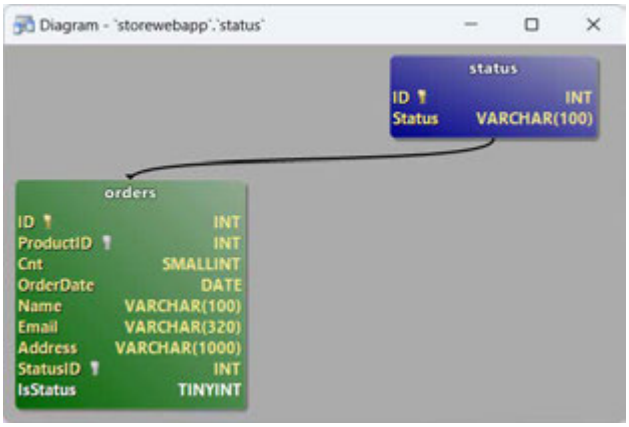

Рис. 2.9 Співвідношення таблиці status з іншими таблицями

Таблиця users призначена для зберігання інформації про користувачів системи. Таблиця має поля: ідентифікатор користувача, логін, пароль, E-mail, ПІБ користувача та ID ролі (зв’язок з таблицею roles). Як і минулого разу, ідентифікатори є цілими числами з властивістю автоінкременту. З іншими таблицями користувачі не мають безпосереднього зв’язку. Схема відношень таблиці users та roles наведено на рис. 2.10.

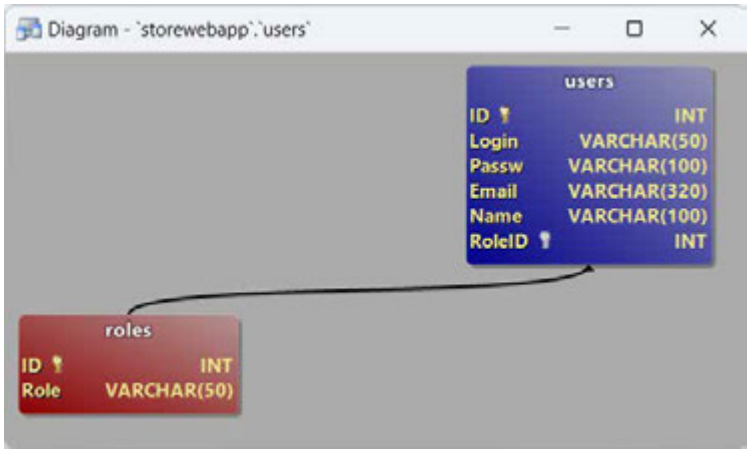

Рис. 2.10 Співвідношення таблиці users з іншими таблицями

Таблиця products призначена для зберігання інформації про товари. Вона містить наступні поля: ідентифікатор товару, назва товару, опис товару, ціна, кількість товару на складі, зображення товару (посилання на зображення), дата додавання, дата останнього продажу (не обов’язкове поле), ID прийнятого (запропонованого) рішення, рішення прийнято (за замовчуванням 0, якщо рішення запропоновано, але не прийнято – 2, якщо прийнято – 1). Таблиця товарів пов’язана з двома іншими таблицями через ключові поля. Для створення нового товару необхідно обов’язково мати створену таблицю з рішеннями. Якщо

рішення ще не прийнято, у таблиці має бути збережено значення за замовчуванням – «Не визначено». Користувач може змінювати його самостійно під час роботи в системі. На рис. 2.11 наведено схемо співвідношення таблиці товарів (products) з таблицями рішення (acts) та замовлення (orders). Цей зв’язок необхідний для коректної обробки замовлень та автоматизованого прийняття рішень системою.

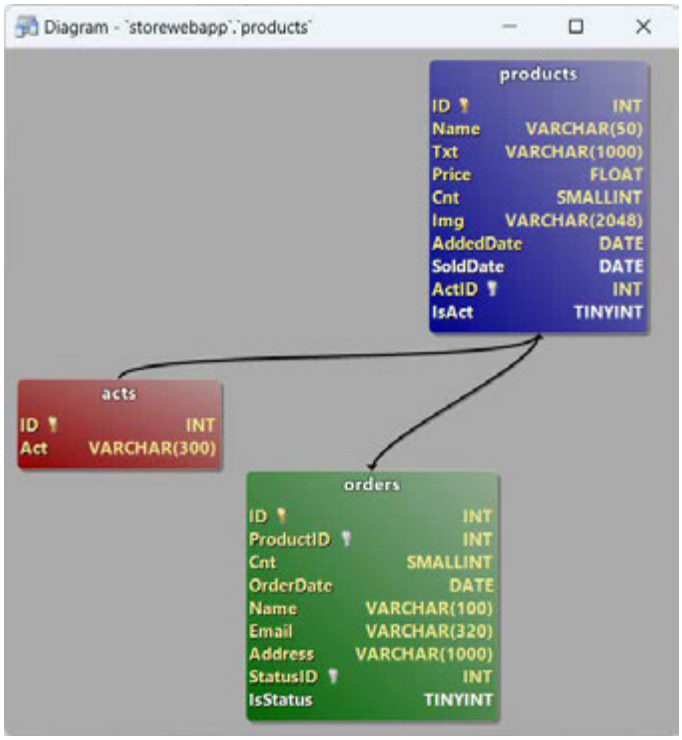

Рисунок 2.11 Співвідношення таблиці products з іншими таблицями

Таблиця orders призначена для зберігання інформації про замовлення. Таблиця має поля: ідентифікатор замовлення, ідентифікатор продукту, кількість продукту, дата замовлення, ПІБ замовника, адреса доставки, ID статусу замовлення, змінна, що означає виділення статусу користувачем (1 або 0, відповідно). Зв’язок таблиці orders з іншими таблицями наведено на рис. 2.12.

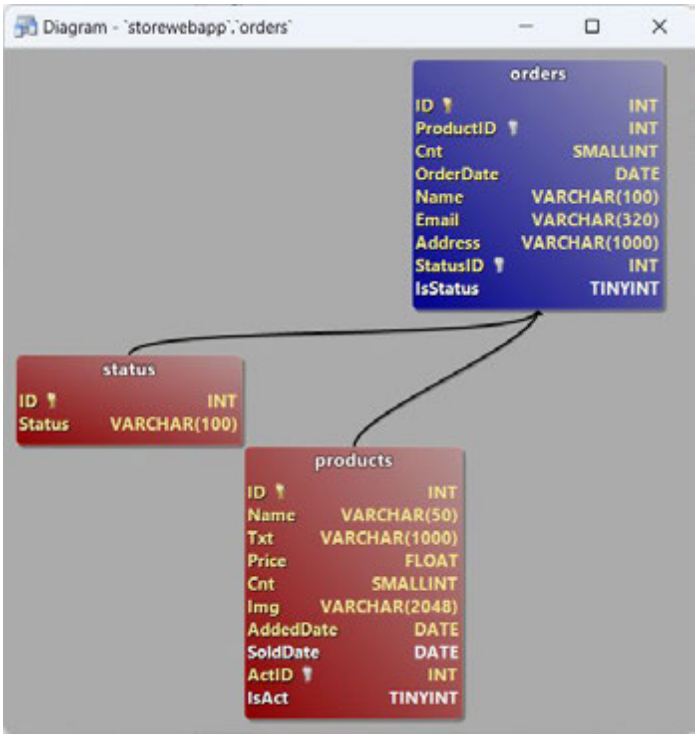

Рис. 2.12 Співвідношення таблиці orders з іншими таблицями

Останньою пропонується розглянути таблицю rules. Вона призначена для зберігання правил, за якими автоматично приймаються рішення. Таблиця має наступні поля: ідентифікатор правила, запит (умова), опис правила, ідентифікатор рішення, що приймається у разі виконання умови. Умова являє собою логічний вираз у форматі, сумісному з мовою СУБД. Взаємозв'язок таблиці з іншими таблицями наведено на рис. 2.13.

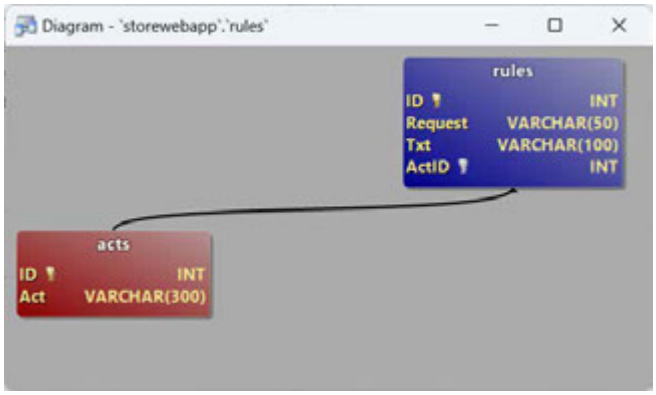

Рис. 2.13 Співвідношення таблиці rules з іншими таблицями

Отже, в результаті виконання моделювання було створено базу даних для зберігання інформації та функціонування системи підтримки прийняття рішень в мережі магазинів. База даних складається з 7 пов'язаних між собою таблиць.

На основі визначеної структури бази даних та сформульованих в даному розділі схемах, що відображають структуру системи та її функціонування, можлива подальша розробка програмного забезпечення, відповідно до поставлених задач.

3 РОЗРОБКА СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

3.1 Обґрунтування вибору засобів розробки

Під час розробки важливо правильно визначити оптимальні засоби для реалізації отриманих моделей, відповідно до поставлених задач. Необхідно враховувати архітектуру системи, її структуру, взаємодію елементів, вимоги до кінцевого програмного продукту тощо. Вибір засобів розробки необхідний для того, щоб підібрати оптимальні технології, які можна застосувати для досягнення поставленої мети.

В першу чергу варто відзначити, що розроблювана система є веб-додатком, а значить має клієнт-серверну архітектуру, що складається з клієнта, сервера та бази даних. Отже, і вибір засобів розробки має відповідати структурі системи. Технології, що використовуються для розробки програмного забезпечення на стороні клієнта та сервера можуть мати суттєві відмінності. Незважаючи на існування технологій, що дозволяють розробляти програмне забезпечення як на стороні клієнта, так і на стороні сервера, в даному випадку доцільно розглянути окремо мови розмітки та програмування, як для клієнта, так і для сервера.

Також під час вибору програмних засобів розробки важливо враховувати вимоги до кінцевого програмного продукту. Наприклад, якщо є умова збереження адаптивності на різних пристроях, необхідно підбирати такі засоби розробки, які дозволяють це забезпечити.

Отже, вибір засобів розробки програмного забезпечення можна умовно розділити на три основні частини: вибір засобів розробки клієнтської частини, вибір засобів розробки серверної частини та вибір системи управління базами даних. Послідовне виконання цих трьох етапів дозволить підібрати оптимальні засоби розробки та створити якісний програмний продукт. Порядок виконання цих етапів наразі не має значення, тому в роботі пропонується спочатку розглянути бази даних, потім клієнт і в останню чергу сервер. Враховуючи вимоги до програмного забезпечення, сформульовані в попередніх розділах, підберемо також технології для оптимальної клієнт-серверної взаємодії.

Оскільки в даній системі найважливішу роль відіграють дані, очевидно, що для її роботи необхідне застосування системи управління базами даних. База даних (БД) – це впорядкований набір логічно пов’язаних даних, що може використовуватись спільно, відповідно до інформаційних потреб користувачів. БД призначені для задоволення користувацьких інформаційних потреб [18].

Застосування БД пов’язане з численними перевагами, серед яких одна з найважливіших – оптимізація та полегшення зберігання та організації великих обсягів інформації. БД дозволяють зберігати дані в таблицях та структурах, що значно спрощує управління ними. Права доступу, що застосовуються в базах даних, дозволяють забезпечити належну конфіденційність та захист інформації. Крім цього, застосування баз даних дозволяє формувати і виконувати різноманітні запити до даних для швидкого та ефективного отримання доступу до необхідної інформації. Використання БД забезпечує цілісність даних та уникнення дублювання інформації.

Бази даних можуть засновуватись на різних моделях. Модель даних являє собою деяку абстракцію, що відображає найважливіші аспекти функціонування предметної області, ігноруючи другорядні. Іншими словами, модель даних є деякою цільовою моделлю предметної області. У моделях даних можна розрізнити три основні складові: структурна, управляюча частина та класи обмежень цілісності даних. Процес створення логічного представлення структури БД називається моделюванням даних [19]. Таке моделювання було виконано в попередньому розділі, але уточнень моделі та без конкретизації вибору системи управління базами даних.

Однією з найбільш розповсюджених моделей даних є реляційна модель, що заснована на математичному понятті відношення і представлення відношень у формі таблиць. Саме така модель буде використана в системі підтримки прийняття рішень. Такий вибір обумовлений логічністю, простотою та інтуїтивністю в структуруванні даних, що полегшить розробку та експлуатацію програмного забезпечення. Отже, необхідно обґрунтувати вибір реляційної системи управління базами даних.

Мови реляційного числення засновані на класичному численні предикатів і надають користувачу правила для написанні запитів до БД. Серед загальних характеристик для різних видів серверів баз даних можна відзначити використання реляційної мови SQL для реалізації запитів до даних [18].

SQL (Structured Query Language) – це стандартна мова для зберігання, обробки та отримання даних в базах даних. За допомогою цієї мови можна описати набори даних, що допомагають відповісти на запити. Використовуючи SQL, важливо дотримуватися правильного синтаксису – набору правил, що забезпечують правильне поєднання елементів мови.

Існує велика кількість систем управління базами даних, серед яких найбільш відомими є MS SQL, MySQL та PostgeSQL. Ці системи мають багато спільного і використовують мову SQL, проте можуть мати незначні відмінності. Вибір оптимальної реляційної СУБД для системи підтримки прийняття рішень.

Прийнято рішення про доцільність використання СУБД MySQL, що є вільною системою управління реляційними базами даних і спроектована для оптимізації обробки великих обсягів даних. Вона створена як альтернатива комерційним системам і відзначається своєю швидкодією та ефективністю. Починаючи як аналог mSQL, з часом MySQL стала однією з найпоширеніших СУБД, використовуваною у різних галузях. Особливої уваги заслуговує її відмінна підтримка для створення динамічних веб-сторінок, що дозволяє використовувати різноманітні мови програмування. Крім того, MySQL є популярним серед багатьох хостинг-провайдерів, включаючи безкоштовні, і має високу рівень документації та розповсюдження. Отже, така система управління базами даних ідеально підходить для реалізації системи підтримки прийняття рішень керівництвом мережі магазинів.

Отже, прийнято рішення про використання реляційної системи управління базами даних MySQL. Розробка серверної частини програмного забезпечення буде виконуватись в тому числі з використанням цієї технології для забезпечення доступу до бази даних. Таким чином, обґрунтування вибору технологій для роботи з базою даних завершено.

На стороні клієнта програмне забезпечення має надавати користувачу зручний та інтуїтивно зрозумілий графічний інтерфейс. Його розробляють з використанням мови розмітки HTML, доповнюючи її каскадними таблицями стилів та скриптами, що дозволяють робити сторінки інтерактивними.

На цьому етапі важливим фактором є те, що веб-додаток (вебсайт), що розробляється, має бути адаптивним і працювати однакого на всіх пристроях від настільних комп'ютерів до мобільних пристроїв. Такий результат може бути забезпечений завдяки спеціальним технологіям, таким як шаблони стилів, каскадні таблиці стилів та скрипти тощо. Проте спочатку варто розглянути загальні відомості про мову розмітки HTML.

HTML (Hypertext Markup Language) – це стандартна мова розмітки, що застосовується для розробки та відображення вебсторінок у веб-браузерах. Вона складається з різних тегів, які визначають структуру та зовнішній вигляд контенту на сторінці (текстові дані, зображення, гіперпосилання тощо). Сьогодні HTML використовується як основна мова для створення вебсторінок та їхньої взаємодії з користувачами через гіпертекстові посилання та форми. Це одна з основних технологій веб-розробки, яка разом з CSS та JavaScript надає розробникам можливості створення структурованих та доступних сторінок, які можна легко інтерпретувати та відображати у веб-браузерах різноманітних пристроїв [20].

CSS (Cascading Style Sheets) – це мова розробки стилів, що використовується для оформлення та контролю зовнішнього вигляду вебсторінок. CSS описує, як елементи відображатимуться у браузері. Застосування CSS дозволяє відокремити оформлення від структури сторінки, спрощуючи підтримку та модифікацію коду. Стилі можна застосовувати безпосередньо до елементів сторінки або використовувати зовнішні стилі, які потім включаються в код сторінки. Крім того, CSS підтримує каскадне спадкування стилів, завдяки чому можна визначати загальні стилі для батьківських елементів і змінювати їх для дочірніх, спрощуючи управління виглядом вебсайтів [20].

JavaScript – це скриптова мова програмування, призначена для розробки динамічних вебсторінок та інтерактивних додатків. Вона застосовується для реалізації різноманітних функціональностей на вебсторінках, таких як обробка подій користувача, динамічне оновлення вмісту сторінок, анімація, валідація форм, інтерактивність з користувачем та взаємодія з сервером без перезавантаження сторінки. JavaScript виконується у веб-браузері клієнта, що дозволяє створювати багатофункціональні та захоплюючі веб-додатки [20].

На JavaScript засновані також інші технології, які можуть бути застосовані під час розробки системи підтримки прийняття рішень. Однією з таких технологій є AJAX (Asynchronous JavaScript and XML) – це технологія, яка дозволяє веб-додаткам обмінюватися даними з сервером асинхронно, без необхідності перезавантаження всієї веб-сторінки. Завдяки AJAX, користувачі можуть взаємодіяти з веб-сайтом більш інтерактивно, оскільки лише частини контенту оновлюються в реальному часі. Це здійснюється за допомогою JavaScript, який надсилає запити до сервера і отримує відповіді, які можуть бути у форматах XML, JSON або HTML. AJAX робить веб-додатки швидшими.

Bootstrap – це фреймворк, призначений для розробки стильних і адаптивних веб-сайтів. Заснований на HTML, CSS і JavaScript, він пропонує розмаїття готових компонентів та стилів, що спрощує швидке створення інтерфейсів. Bootstrap має широкий вибір гнучких компонентів, таких як кнопки, навігаційні панелі, форми, таблиці, каруселі, модальні вікна тощо. Особливістю є його сіткова система, що дозволяє створювати респонсивні макети, які легко адаптуються до різних розмірів екранів та пристроїв. Bootstrap широко використовується веб-розробниками завдяки простоті використання та багатому функціоналу, що допомагає зберегти час та зусилля під час створення сучасних веб-додатків.

Отже, клієнтська частина системи буде розроблена з використанням мови розмітки HTML, каскадних таблиць стилів CSS та JavaScript, зокрема технології AJAX, що дозволить динамічно оновлювати дані на сторінках без необхідності повного перезавантаження. Для забезпечення адаптивності сайту вирішено

використовувати технологію Bootstrap. На цьому обґрунтування вибору засобів розробки програмного забезпечення на стороні клієнта завершено.

Серверна сторона повинна обробляти великі обсяги інформації. На сервер надходитимуть запити від клієнтів, формуватимуться та виконуватимуться запити до бази даних, виконуватиметься обробка даних, формування та відправлення відповідей. Для цього необхідне спеціалізоване серверне програмне забезпечення, для роботи з яким існують спеціалізовані серверні мови розробки. Для розробки серверної частини можуть бути використані різні мови програмування. Серед них PHP, NodeJS, ASP.Net та інші. Кожна з цих мов має свої особливості, які важливо враховувати під час вибору технологій для розробки сервера. Прийнято рішення використовувати мову програмування PHP для розробки серверної частини системи підтримки прийняття рішень.

PHP (Hypertext Preprocessor) – мова сценаріїв загального призначення з відкритим вихідним кодом, що отримала широке застосування у веб-розробці. Однією зі значних особливостей PHP є те, що такий код може бути вбудований безпосередньо в HTML, спрощуючи розробку динамічних веб-сторінок. Основна відмінність PHP від клієнтських мов програмування, полягає в тому, що PHP-код виконується на стороні сервера, що дозволяє генерувати HTML, який потім надсилається клієнту. Тобто клієнт отримує результати виконання сценарію, але не має доступу до вихідного коду. Завдяки можливості налаштування веб-сервера для обробки всіх HTML-файлів через PHP, можна забезпечити, що користувачі не отримають доступ до вихідного коду. Одна з найбільш значних переваг PHP – його легкість в опануванні для початківців, а також наявність багатьох додаткових функцій для професійних розробників [21].

Однією з особливостей мови програмування PHP є її підтримка взаємодії з базами даних MySQL. Це дозволяє легко реалізувати сервер, здатний обробляти значні обсяги даних, без використання стороннього програмного забезпечення.

Отже, для розробки серверної частини системи підтримки прийняття рішень керівництвом у мережі магазинів буде застосовано мову програмування PHP та реляційну СУБД MySQL.

3.2 Розробка клієнтської частини системи

Для розробки програмного забезпечення на стороні клієнта, як було визначено в попередньому підрозділі, буде використовуватись мова розмітки HTML. Проте у клієнт-серверних системах, що використовують мову програмування PHP, є можливість поєднання клієнтського та серверного коду в одному файлі. Зазвичай такі файли мають розширення .PHP і містять код обома мовами одночасно. Сервер приховує від користувачів частину, розроблену з використанням PHP, і відправляє виключно HTML-код. Таке рішення обумовлено більшою простотою розробки та більш широкими можливостями щодо об’єднання клієнтських та серверних функцій в одній сторінці.

В першу чергу визначимо основні форми, які мають бути доступні користувачу на стороні клієнта. Користувач повинен бачити головну сторінку, на якій знаходиться навігаційна панель та головне меню, що відображає основний функціонал системи. Навігаційна панель розміщується у верхній частині сторінки і має такі основні пункти: назва системи, головна сторінка, інструкція користувача, кнопка входу/виходу. Навігаційне меню має надавати можливість швидкого переходу до наступних розділів: товари, замовлення, користувачі.

Отже, клієнтська частина системи передбачає реалізацію таких сторінок:

1. Головна сторінка з навігаційним меню. Сторінка доступна всім користувачам без авторизації.
2. Форма входу, де користувач може виконати авторизацію. Форма доступна всім користувачам, які ще не авторизувались на сайті.
3. Розділ роботи з товарами, що дозволяє переглядати, додавати, редагувати та видаляти товари, а також приймати щодо них рішення. Доступний також пошук товарів за назвою. Розділ доступний тільки адміністратору.
4. Розділ роботи з замовленнями. Доступний як адміністратору, так і менеджеру. В цьому розділі користувачам надається можливість перегляду, створення, редагування, видалення замовлень, а також прийняття рішень щодо

них. Рішення для замовлень передбачають ручне маркування, що може бути зручним для користувачів системи. Доступний також пошук за назвою товару, за ПІБ або E-mail замовника, відображення замовлень за категоріями (наприклад, тільки завершені або тільки ті, що мають статус «В обробці»).

5. Розділ роботи з користувачами. Розділ доступний тільки адміністратору. Надає можливість перегляду, додавання, редагування та видалення користувачів системи.

6. Підрозділ роботи з правилами прийняття рішень. Доступний тільки адміністратору. Вхід до підрозділу здійснюється зі сторінки товарів. Передбачена можливість перегляду, додавання, редагування та видалення правил для автоматизованого прийняття рішень.

7. Інструкція користувача. Відображення детальної інструкції з користування системою підтримки прийняття рішень. Сторінка доступна всім користувачам, не потребує авторизації.

Всі сторінки повинні мати дизайн в одному стилі. У верхній частині екрану завжди знаходиться верхня навігаційна панель, що дозволяє в будь-який момент переключитись на головне меню або інструкцію користувача. Нижче на деяких сторінках може бути розташоване спеціальне навігаційне меню, призначене для швидких переходів між розділами. Вхід, додавання та редагування окремих елементів виконується через форми, що відкриваються на новій сторінці.

Головна сторінка знаходитиметься у файлі index.php, форма входу – login.php, для роботи з товарами – products.php, для роботи з замовленнями – orders.php, для роботи з користувачами – users.php, для роботи з правилами прийняття рішень – rules.php. При цьому для додавання та редагування нових товарів, замовлень, користувачів, правил, використовуватимуться форми з назвами f_products, f_orders, f_users та f_rules. Одна форма може використовуватись як для додавання, так і для редагування, що дозволяє зменшити обсяг роботи під час розробки та спростити програмне забезпечення.

Отже, визначивши структуру клієнтської частини системи, можна перейти до розробки сторінок з використанням мови розмітки HTML.

Для всіх даних, що вводяться користувачем, у системі має бути передбачена можливість валідації. Це необхідно для того, щоб на сервер відправлялись тільки коректні дані. Валідація здійснюється завдяки JS-коду на сторінці форми. Цей код відстежує стан форми та перевіряє дані перед відправленням на сервер. Розглянемо валідацію даних з використанням JS на прикладі форми входу:

```
const togglePassword = document.getElementById('togglePassword');
const passwordField = document.getElementById('password');
const loginField = document.getElementById('login');
const form = document.querySelector('form');
const errorMessage = document.getElementById('error-message');
togglePassword.addEventListener('click', () => {
  const type = passwordField.getAttribute('type') === 'password' ? 'text' : 'password';
  passwordField.setAttribute('type', type);
  togglePassword.textContent = type === 'password' ? 'Показати' : 'Приховати';
});
form.addEventListener('submit', (event) => {
  const login = loginField.value;
  const regex = /^[a-zA-Z0-9]+$/;
  if (!regex.test(login)) {
    event.preventDefault();
    errorMessage.classList.remove('d-none');
  } else {
    errorMessage.classList.add('d-none');
  }
});
```

Наведений код відстежує вміст полів введення перед відправленням запитів на сервер, уникаючи таким чином виникнення помилок на стороні сервера та підвищуючи якість програмного забезпечення в цілому.

Всі елементи сторінок створюються з використанням стандартних стилів бібліотеки Bootstrap. Наприклад, для повідомлення про неправильне введення логіну, відповідно до наведеного вище коду, можна використовувати спливаюче вікно, яке з’являтиметься завдяки JavaScript, а його стиль буде сформований завдяки бібліотекам Bootstrap:

```
<div class="container alert alert-danger d-none" id="error-message" role="alert">Логін може містити лише букви латиниці та цифри.</div>
```

Також система підтримує динамічне оновлення всіх елементів, що відображаються на екрані. Таблиці оновлюються одразу після натискання на кнопки прийняття рішень, видалення або додавання нових елементів. Перезавантаження всієї сторінки не потрібне. Для цього використовується технологія AJAX. Розглянемо код завантаження таблиці товарів.

Спочатку необхідно створити таблицю, точніше її шапку:

```
<table id="products" class="table">
  <thead>
        <tr>
              <th scope="col">Назва</th>
              <th scope="col">Опис</th>
              <th scope="col">Ціна</th>
              <th scope="col">Зображення</th>
              <th scope="col">Кількість</th>
              <th scope="col">Дата додавання</th>
              <th scope="col">Дата останнього замовлення</th>
              <th scope="col">Рішення</th>
              <th scope="col">Дії</th>
        </tr>
  </thead>
  <tbody></tbody>
</table>
```

Після цього перейдемо до JS-коду, винесеному в зовнішній файл. Цей код виконується автоматично, використовуючи id таблиці.

```
function loadProducts(searchTerm = '') {
$.ajax({
  url: 'c_products.php',
  method: 'GET',
  data: { search: searchTerm },
  dataType: 'json',
  success: function(data) { // TODO }
  });
}
```

Наведений код є спрощеною версією, його основна частина прихована, оскільки не має безпосереднього відношення до динамічного оновлення AJAX. Далі здійснюється виконання функції динамічного створення елементів інтерфейсу та їхнє відображення на сторінці. Все інше відбувається на стороні сервера.

Оскільки сайт використовує стилі Bootstrap, він не потребує обов’язкового написання власних каскадних таблиць стилів. Але для покращення зовнішнього вигляду сайту та його адаптивності, зокрема, було розроблено також власні стилі для деяких елементів інтерфейсу. Розглянемо розроблений код стилів для деяких об’єктів, що згадувались раніше:

```
.card { height: 100%; }
.error-code {
  font-size: 10rem;
  font-weight: bold;
  color: #dc3545;
}
```

Таким чином було розроблено клієнтську частину системи. Розширена версія вихідного коду HTML та JS (на прикладі інтерфейсу роботи з замовленнями) наведена в Додатках А і Б, відповідно.

3.3 Розробка серверної частини

Серверна частина системи, як було обґрунтовано раніше, розроблена з використанням мови програмування PHP. Ця мова підтримує взаємодію з базами даних MySQL, що також використовується в проєкті. Розробку серверної частини пропонується розпочати зі створення бази даних за допомогою SQL-запитів. Це необхідно для коректної роботи коду, розробленого з використанням PHP. Створення бази даних є достатньо швидким процесом і потребує написання запитів для всіх необхідних таблиць. Розглянемо код запитів для створення деяких елементів бази даних:

```
CREATE DATABASE StoreWebApp; -- Створення бази даних
USE StoreWebApp; -- Використання створеної бази даних
CREATE TABLE Acts(
  ID INT AUTO_INCREMENT PRIMARY KEY,
  Act VARCHAR(300) NOT NULL
);
CREATE TABLE Products(
  ID INT AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR(50) NOT NULL,
  Txt VARCHAR(1000) NOT NULL,
  Price FLOAT(10,2) NOT NULL,
  Cnt SMALLINT(5) NOT NUll,
  Img VARCHAR(2048) NOT NULL,
  AddedDate DATE NOT NULL,
  SoldDate DATE,
  ActID INT NOT NULL, FOREIGN KEY (ActID) REFERENCES Acts(ID),
  IsAct TINYINT
);
CREATE TABLE Rules(
  ID INT AUTO_INCREMENT PRIMARY KEY,
```

```
  Request VARCHAR(50) NOT NULL,
  Txt VARCHAR(100) NOT NULL,
  ActID INT NOT NULL, FOREIGN KEY (ActID) REFERENCES Acts(ID)
);
```

Даний код створює не всі необхідні таблиці, обмежуючись виключно тими, що необхідні для роботи з товарами та автоматизованого прийняття рішень. Повна версія коду створення бази даних наведена в Додатку В.

Проте основна частина серверного програмного забезпечення розроблена з використанням мови програмування PHP. Запити, що надходять зі сторони клієнта, обробляються контролерами, назва яких відповідає назві форми, але має літеру c: c_users, c_products, c_orders тощо.

Важливою частиною роботи сервера є забезпечення регулювання прав користувачів та інші заходи захисту. Всі дані, що обробляються сервером, мають бути перевірені, а всі форми захищені від SQL-ін’єкцій та CSRF-атак. Для забезпечення обмежень доступу та захисту від CSRF-атак використовуються сесії. Розглянемо на прикладі:

```
session_start();
if (empty($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
if (!isset($_SESSION['role']) || $_SESSION['role'] === 'Unknown') {
    header('Location: login.php');
  exit();
}
if ($_SESSION['role'] != 'Admin') {
    header('Location: AccessDenied.php');
  exit();
}
$DBhost = 'localhost';
$DBuser = 'root';
```

```
$DBpassword = '';
$DBname = 'StoreWebApp';
mysqli_report(MYSQLI_REPORT_ERROR | MYSQLI_REPORT_STRICT);
try {
  $db = new mysqli($DBhost, $DBuser, $DBpassword, $DBname);
} catch (mysqli_sql_exception $e) {
  $error = 'Помилка відкриття бази даних: ' . $e->getMessage();
}
```

Наведений вище уривок коду створює сесію, генерує CSRF-токен, перевіряє роль користувача і виконує підключення до бази даних. Також, з метою підвищення безпеки, всі паролі зберігаються виключно у хешованому вигляді. Розглянемо приклад коду перевірки пароля:

```
if (password_verify($_POST['password'], $hashedPassword)) {
  $_SESSION['role'] = $userRole;
  $_SESSION['name'] = $userName;
  header('Location: index.php');
  exit();
} else {
  $error = 'Неправильний пароль.';
  }if (password_verify($_POST['password'], $hashedPassword)) {
  $_SESSION['role'] = $userRole;
  $_SESSION['name'] = $userName;
  header('Location: index.php');
  exit();
} else {
  $error = 'Неправильний пароль.';
}
```

Таким чином здійснюється перевірка правильності введення облікових даних для входу в акаунт. У випадку помилки користувач отримає відповідне сповіщення.

В цілому серверне програмне забезпечення працює за наступним алгоритмом: на сервер надходить запит від клієнта, сервер перевіряє отримані дані, виконує запит до бази даних, формує і повертає клієнту відповідь. Тобто алгоритм достатньо простий. Система ускладняється виключно кількістю інтерфейсів, пов’язаних з обробкою різних типів даних, що робить неможливим використання універсальних рішень.

Розглянемо приклад вихідного коду, що відповідає за автоматизоване прийняття рішень:

```
function updateActID($conn) {
    // Отримуємо всі правила з таблиці rules
    $result = $conn->query("SELECT Request, ActID FROM rules")
    if (!$result) {
        echo "Error fetching rules: " . $conn->error;
        return; // Виходимо, якщо не вдалося отримати правила
    }
    while ($row = $result->fetch_assoc()) {
        // Розділяємо умови за комами
        $conditions = explode(',', $row['Request']);
        // Об'єднуємо їх через OR
        $request = implode(' OR ', array_map('trim', $conditions));
        $actID = $row['ActID'];
        // Формуємо SQL запит для оновлення Products
        $sql = "UPDATE Products SET ActID = $actID, IsAct = 2 WHERE ($request) AND (IsAct NOT IN (1, 2) OR IsAct IS NULL)";
        // Виконуємо запит
        if (!$conn->query($sql)) {
            echo "Error updating ActID: " . $conn->error;
        }
    }
}
```

Наведений вище вихідний код запускається щоразу, коли оновлюється таблиця, для того, щоб оперативно реагувати на можливі зміни. Код працює наступним чином: якщо виконується умова, задана адміністратором у спеціальному інтерфейсі, то пропонується прийняти відповідне рішення і товар, до якого це рішення відноситься виділяється в таблиці жовтим кольором. Для підтвердження прийняття рішення користувач натискає кнопку «Прийняти рішення» і рядок забарвлюється в зелений колір, що означає, що рішення прийнято. За замовчуванням всі рядки мають звичайний білий колір фону.

Вихідний код, що відповідає за обробку товарів, наведений в Додатку Г.

Таким чином було завершено розробку програмного забезпечення системи підтримки прийняття рішень керівництвом у мережах магазинів. Отримано вебсайт, що дозволяє автоматизувати процес прийняття рішень у сфері роздрібної торгівлі. Структура сайту включає інтерфейси для роботи з товарами, замовленнями, користувачами та правилами автоматизованого прийняття рішень. Результатом виконання роботи є розроблене програмне забезпечення, готове для використання.

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

4.1 Апаратні і програмні вимоги

Розгортання сучасних веб-додатків вимагає врахування системних вимог як до апаратного, так і до програмного забезпечення. Оскільки розроблена система підтримки прийняття рішень використовує технології HTML, CSS, JavaScript, Bootstrap, jQuery, AJAX, PHP та MySQL, важливо правильно налаштувати середовище для її функціонування.

Слід зазначити, що для цієї системи рекомендується використовувати стек LAMP, який включає Linux, Apache, MySQL та PHP. Цей набір технологій забезпечує надійний фундамент для розробки динамічних веб-додатків, і він відомий своєю простотою та доступністю.

Для ефективного розгортання LAMP-системи, слід забезпечити базове апаратне забезпечення. Мінімальні вимоги включають:

– процесор з тактовою частотою не нижче 1 ГГц, що дозволить обробляти запити з адекватною швидкістю;

– 1 ГБ оперативної пам’яті для підтримки роботи веб-сервера та бази даних. Хоча це може бути обмежуючим фактором для обробки великих обсягів даних, така кількість пам’яті є достатньою для невеликих проектів.

– жорсткий диск обсягом не менше 20 ГБ, що дозволить зберігати не тільки систему, але й базу даних, а також додаткові файли проекту;

Ці вимоги є мінімальними, і для кращої продуктивності рекомендується використовувати більш потужні конфігурації.

Для розгортання LAMP-системи потрібно попередньо встановити певне програмне забезпечення. Рекомендується почати з вибору операційної системи. Найчастіше для цього використовують дистрибутиви Linux, такі як Ubuntu або CentOS, завдяки їхній стабільності та широкій підтримці спільноти.

Після установки Linux слід перейти до налаштування веб-сервера. Apache є одним з найпопулярніших веб-серверів, і його установка є досить простою. Зазвичай достатньо виконати одну команду в терміналі для його установки.

Після установки Apache необхідно налаштувати його конфігураційні файли, щоб забезпечити правильну маршрутизацію запитів.

Наступним кроком є установка MySQL, що є системою управління базами даних. MySQL дозволяє зберігати, обробляти та організовувати дані. Важливо налаштувати базу даних і створити користувача з необхідними правами доступу, щоб забезпечити безпечну та ефективну роботу з даними.

Після цього слід встановити PHP, мову програмування, яка відповідає за обробку серверних запитів. PHP тісно інтегрується з Apache, що дозволяє обробляти динамічний контент. Важливо встановити необхідні розширення для PHP, такі як MySQLi або PDO, для забезпечення зв'язку з базою даних.

Коли основні компоненти LAMP-системи встановлені, варто звернути увагу на додаткове програмне забезпечення, яке може підвищити продуктивність і зручність використання. Наприклад, для полегшення роботи з базою даних можна встановити phpMyAdmin, інтерфейс для адміністрування MySQL. Це забезпечить зручний графічний інтерфейс для роботи з базами даних.

Порядок встановлення компонентів LAMP є важливим для коректної роботи всієї системи. Рекомендується дотримуватися наступної послідовності:

1. Встановлення Linux.
2. Налаштування Apache, включаючи конфігурацію основних параметрів.
3. Встановлення MySQL та налаштування бази даних.
4. Встановлення PHP та необхідних модулів.
5. Установка додаткових інструментів, таких як phpMyAdmin, для полегшення адміністрування.

Забезпечення належних системних вимог є ключовим етапом у розгортанні веб-додатків на базі LAMP. Виконання всіх вищезгаданих кроків дозволить створити стабільне та продуктивне середовище для роботи вашої системи, що використовує HTML, CSS, JavaScript, Bootstrap, jQuery, AJAX, PHP та MySQL. Кожен з цих компонентів відіграє важливу роль у формуванні якісного продукту, здатного задовольнити потреби користувачів.

Після встановлення LAMP необхідно встановити саму систему, створити базу даних і заповнити таблиці, недоступні для редагування через графічний інтерфейс. Пропонується виконати наступні запити до бази даних для першого запуску:

```
USE StoreWebApp;
INSERT INTO Roles(Role) VALUES ('Admin'), ('Manager');
INSERT INTO Acts(Act) VALUES ('Не визначено'),
('Виставити на видне місце'),
('Провести акцію для стимулювання продажу'),
('Зробити знижку 10%'),
('Перевірити наявність на складі'),
('Утилізувати');
INSERT INTO Status(Status) VALUES ('Прийнято'),
('В обробці'),
('Відправлено'),
('Відхилено');
-- Користувачі (Login: admin, Password: Admin | Login: manager, Password: Manager)
INSERT INTO Users(Login, Passw, Email, Name, RoleID) VALUES ('admin',
'$2y$10$o6l1UsZo0S0wFPhzq2s6qu1EHvlFuwR845FXvuaEVsmwu0f9sytjC',
'admin@example.com', 'Адміністратор Системи', 1),
('manager',
'$2y$10$S9dmhXanvmp2RfJkEZoZ/.RH6RrX0SUGUM1tlgFrSvdhKvt.ifcXK',
'manager@example.com', 'Менеджер магазину', 2);
```

Виконання цього коду створить стандартні ролі користувачів, рішення та статуси замовлень. Також будуть створені два користувача – Адміністратор та Менеджер з логіном і паролем «Admin» та «Manager», відповідно. Важливо враховувати, що логін до регістру не чутливий, а пароль чутливий, тому необхідно вводити паролі саме з великої літери.

Крім розгортання апаратного та програмного забезпечення, необхідно також приділити достатню увагу інструкціям для користувачів системи. У системі передбачено можливість ознайомлення з інструкцією щодо використання за посиланням у верхній навігаційній панелі.

На сторінці наведено детальний опис системи та інструкції з виконання тих чи інших операцій, передбачених у системі. Скріншот сторінки інструкцій наведено на рис. 4.1.

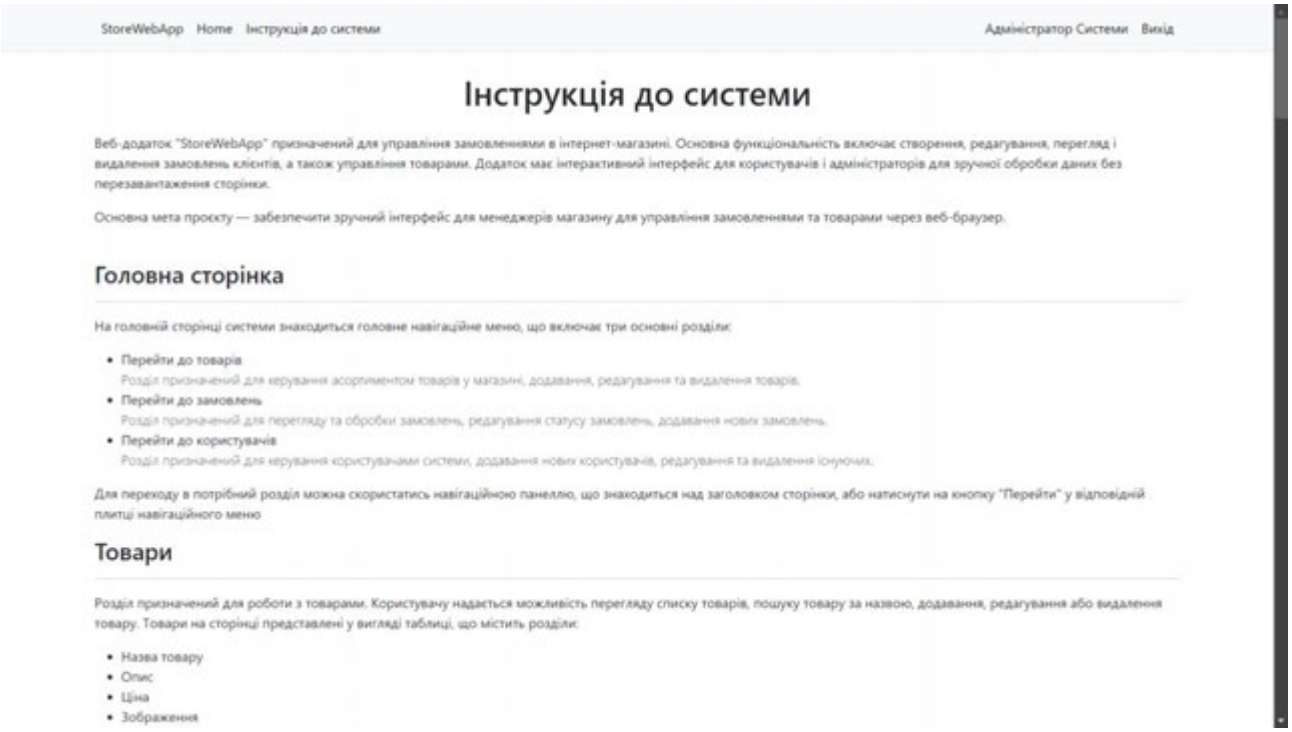

Рис. 4.1 Інструкція користувача

Таким чином користувач може проходити навчання користуванню системою самостійно, а також знаходити відповіді на питання, які можуть виникати в процесі роботи, переходячи за посиланням у верхній навігаційній панелі. В цілому ж, графічний інтерфейс користувача є простим та інтуїтивно зрозумілим, а також представлений українською мовою. Тому користувач може почати працювати без довготривалого навчання.

Розроблена система орієнтована на використання у сфері роздрібної торгівлі і може бути зручною альтернативою існуючим універсальним системам для аналітики даних. Не рекомендується використовувати систему в поєднанні зі сторонніми розширеннями для браузерів.

4.2 Тестування розробленої системи

Для тестування програмного забезпечення відомі різні підходи. Одним з найбільш поширених варіантів є тестування методами чорного та білого ящиків. В першому випадку передбачається відсутність знань про те, як влаштований об’єкт, а в другому – про об’єкт відома максимальна кількість інформації. В рамках виконання роботи доцільно застосувати обидва підходи. Також, в процесі розробки було використане модульне тестування, коли перевіряється працездатність кожної складової програми окремо.

На завершальному етапі виконання роботи пропонується перевірити працездатність розробленого програмного забезпечення шляхом його запуску та виконання типових робочих операцій, перелічених у попередніх розділах. При цьому необхідно моделювати різні ситуації, що можуть виникати під час роботи системи та аналізувати отримані результати.

Розпочати тестування пропонується із запуску головної сторінки сайту без проходження авторизації (рис. 4.2).

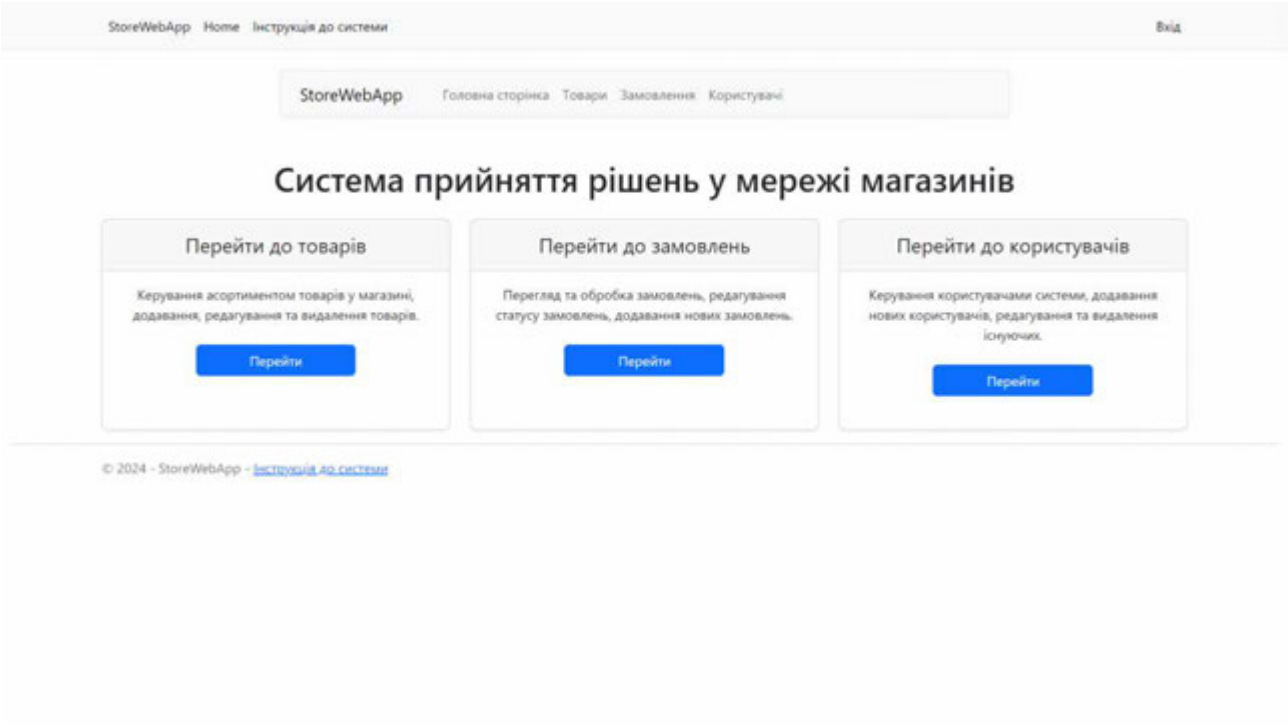

Рис. 4.2 Головне меню системи

На сторінці відображається головне меню системи з можливістю навігації. Доступний вибір з трьох розділів: Товари, Замовлення та Користувачі. Сторінка

відображається коректно, всі посилання працюють, помилки в консолі браузера відсутні. Дизайн сайту адаптивний, під час масштабування зміщення відсутні.

Наступним корком є перевірка авторизації. Пропонується ввести правильний логін від одного користувача і пароль від іншого. Результат виконання тесту наведено на рис. 4.3.

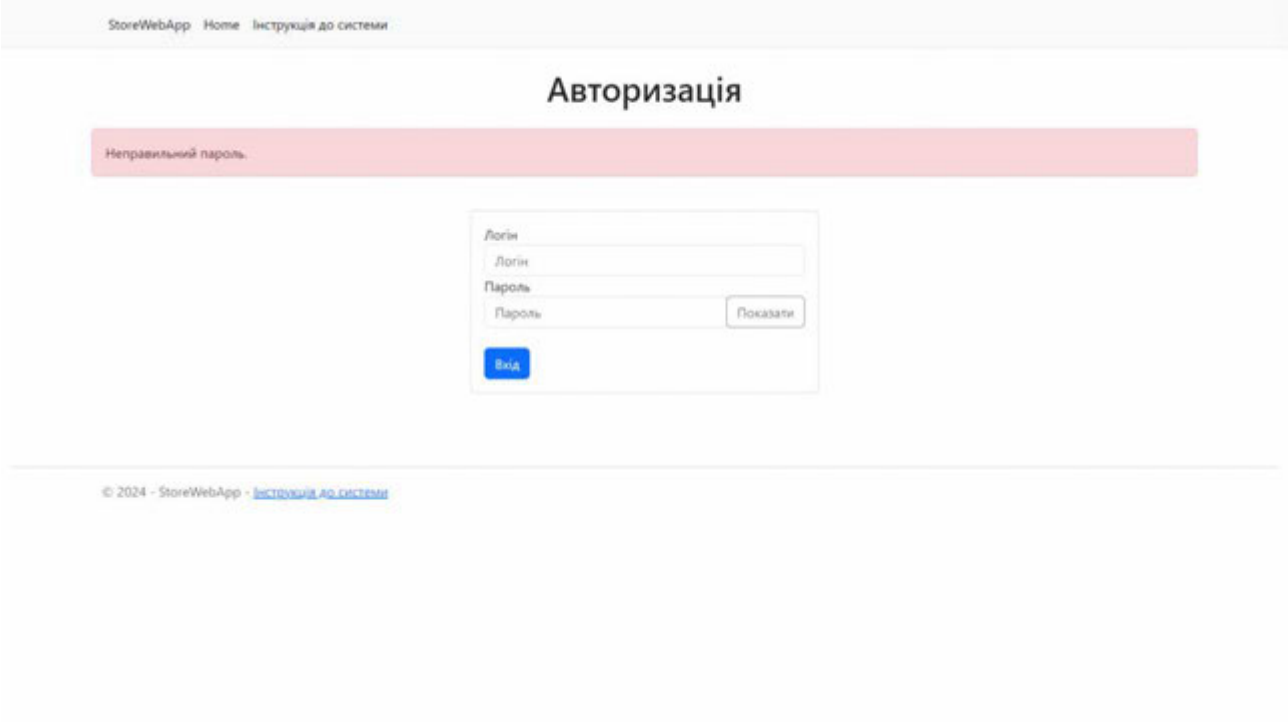

Рис. 4.3 Авторизація з неправильним паролем

В результаті виконання тесту поля введення очищуються, а на екрані відображається повідомлення про помилку. У повідомленні уточнюється, що неправильним був саме пароль. Кнопка «Показати/Приховати» також працює коректно, відповідно до заданих алгоритмів.

Наступним шляхом тестування є спроба отримати доступ до робочих розділів сайту в обхід авторизації. Для цього, не виконуючи вхід, перейдемо за посиланнями на розділи для роботи з товарами, замовленнями або користувачами з головного меню сайту або через навігаційну панель, яку також можна побачити у головному меню. Очікується, що сайт має перенаправити користувача на сторінку авторизації і не пропустити невідомих користувачів у заборонені розділи. Форма авторизації має відображатись шляхом заміщення сторінки, а не накладання. В результаті спрацьовує перенаправлення.

Наступним шляхом буде тестування обмеження доступу для авторизованих користувачів. Необхідно авторизуватись як Менеджер і перейти в розділ, призначений для Адміністратора. Наприклад, розділ редагування даних користувачів. Результат виконання тесту наведено на рис. 4.4.

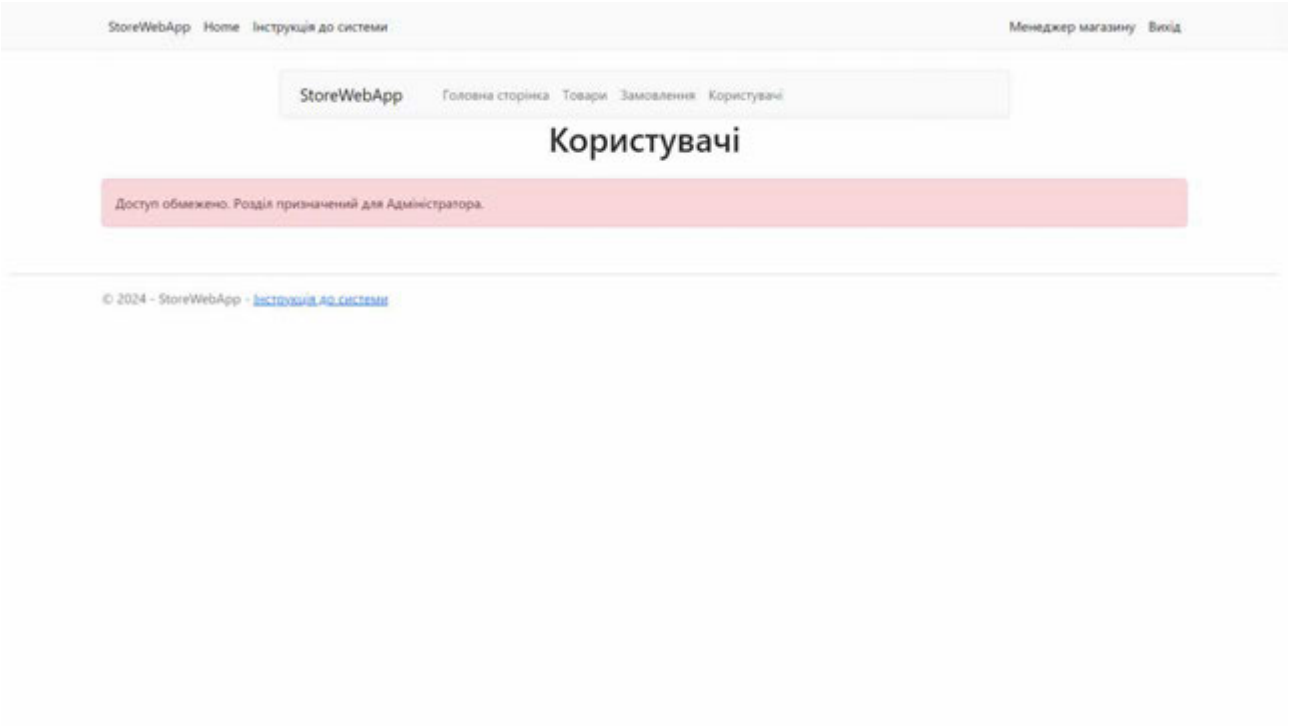

Рис. 4.4 Обмеження доступу

Обмеження спрацьовує, доступу нема. Користувач отримує повідомлення про обмеження доступу, як і очікувалось. Тест пройдений успішно.

Після цього пропонується виконати авторизацію як Адміністратор і перейти у розділ роботи з товарами. У розділі роботи з товарами мають виконатись правила автоматизованого прийняття рішення. Товари, для яких система пропонує рішення підсвічуються жовтим кольором, а після прийняття рішення такі товари стають зеленими. Якщо рішення скасувати, фон знову стане білим.

Для виконання цього тесту також необхідно попередньо створити правила прийняття рішень для товарів. Щоб це зробити, необхідно перейти у розділ товарів, а з нього відкрити правила за допомогою відповідної кнопки меню. На даному екрані необхідно скористатись інтерфейсом створення або редагування існуючих рішень, задаючи правила, які мають спрацювати. Для того, щоб дізнатись імена полів таблиці товарів, необхідно прочитати інструкцію.

Результат виконання тесту автоматизованого прийняття рішень у системі наведено на рис. 4.5.

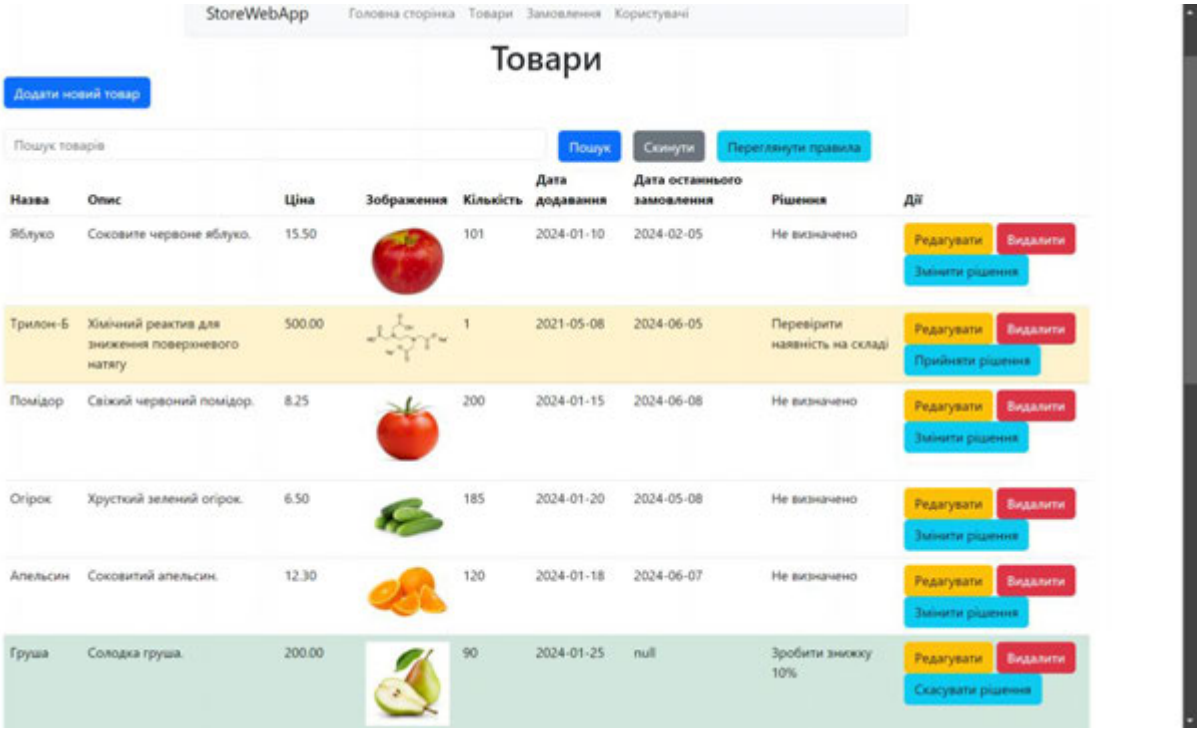

Рис. 4.5 Результат автоматизованого прийняття рішень

Всі елементи інтерфейсу працюють коректно, система пропонує приймати рішення, відповідно до заданих умов. Виконання тесту можна вважати успішним.

Перейдемо до розділу замовлень та виконаємо пошук за іменем замовника. Результат тесту наведений на рис. 4.6.

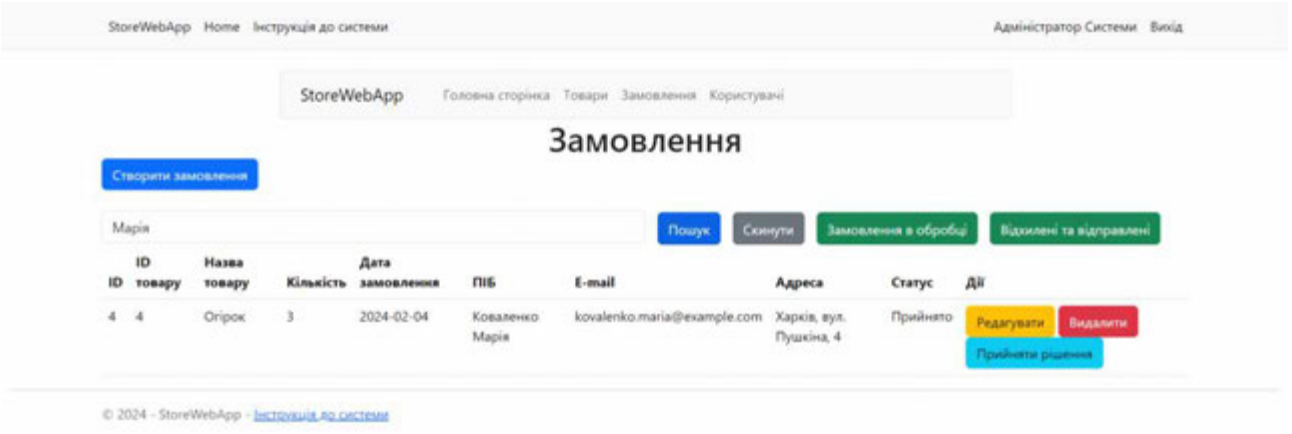

Рис. 4.6 Пошук за іменем замовника

На екрані відображається результат пошуку. Система працює коректно. Отже, можна зробити висновок про коректну роботу системи в цілому.

4.3 Аналіз отриманих результатів

Аналізуючи отримані результати роботи, можна зробити висновок про доцільність проведення досліджень систем підтримки прийняття рішень у різних галузях. На прикладі мереж магазинів було отримано результати, що можуть підвищити ефективність діяльності підприємства за умови впровадження розробленого програмного рішення.

В ході виконання роботи було розроблено клієнт-серверне програмне забезпечення. На сьогодні подібне програмне забезпечення не отримало широкого застосування, якщо йдеться про мережі магазинів. Відома порівняно невелика кількість наукових публікацій, пов’язаних з темою дослідження. Це означає, що дане дослідження має практичну користь і може стати основою для подальшого вивчення застосування систем підтримки прийняття рішень в управлінні магазинами.

Система складається з клієнта, сервера та системи управління базою даних. Користувачам доступний зручний та інтуїтивно зрозумілий графічний інтерфейс з детальною інструкцією щодо використання. Сайт є адаптивним і працює на будь-яких пристроях.

В результаті тестування програмного забезпечення були досягнуті всі поставлені цілі, ПЗ працює коректно, відповідно до технічного завдання, сформульованого на основі вимог сучасних мереж магазинів.

Отримані внаслідок роботи системи рішення виглядають адекватно і відповідають умовам, які може задавати користувач вручну. Тобто, система виконує поставлені задачі і може бути застосована в якості допоміжного програмного забезпечення для керівництва мереж магазинів.

ВИСНОВКИ

В ході виконання магістерської кваліфікаційної роботи було проведено комплексне дослідження систем підтримки прийняття рішень керівництвом мереж магазинів. Розглянуто як теоретичні питання теорії прийняття рішень, так і виконано практичну реалізацію спроектованої системи.

Робота складається з чотирьох структурних розділів. Перший розділ присвячено системному аналізу предметної області досліджень. Було розглянуто поняття рішень, їх прийняття та систем підтримки прийняття рішень, наведено загальні теоретичні відомості. Розглянуто існуючі проблеми в управлінні мережами магазинів і доведено відсутність ефективних рішень, спрямованих саме на цю галузь. Проведено аналіз літературних джерел, в тому числі наукових публікацій за темою дослідження. В результаті було з’ясовано, що тема систем підтримки прийняття рішень добре досліджена, проте питання впровадження таких систем у галузі роздрібної торгівлі недостатньо досліджено і не отримало відображення у науковій літературі. Також було розглянуто існуючі програмні рішення і зроблено висновок про відсутність рішень, які можна було б легко запровадити для управління мережею магазинів. Після проведення аналізу предметної області, було поставлено задачі до розроблюваного програмного забезпечення.

У другому розділі роботи було виконано моделювання системи. Розглянуто питання клієнт-серверної архітектури, описано інформаційні потреби та функції системи. Розроблено UML-діаграми, що відображають структуру системи та взаємодію між її елементами. Найбільшою за обсягом складовою моделювання було моделювання бази даних. Було сформульовано оптимальну структуру бази даних, складено діаграми, що відображають взаємозв’язки між таблицями.

Третій розділ роботи присвячено розробці програмного забезпечення. В першу чергу було запропоновано обґрунтування вибору засобів реалізації ПЗ, після чого було розглянуто безпосередньо розробку. В якості засобів розробки клієнта було обрано мову розмітки HTML, каскадні таблиці стилів CSS,

скриптову мову програмування JavaScript, включаючи технології Bootstrap, jQuery та AJAX. Для сервера було обрано мову програмування PHP, а базу даних створено на базі MySQL. Це оптимальне поєднання для сучасного веб-проєкту.

Розробка програмного забезпечення складалася з трьох основних, тісно пов’язаних, частин: розробка клієнтського ПЗ, розробка серверного ПЗ та розробка бази даних. Взаємодія з базою даних здійснюється в межах одного сервера. База даних змодельована у третій нормальній формі.

Четвертий розділ роботи включає огляд системних вимог системи до апаратного та програмного забезпечення, надання рекомендацій щодо експлуатації ПЗ, проведення тестування та аналіз отриманих даних.

Розроблена система підтримки прийняття рішень може бути застосована у мережах магазинів з метою оптимізації процесів управління. Виконана робота може стати основою для подальших наукових досліджень систем підтримки прийняття рішень.

ПЕРЕЛІК ЛІТЕРАТУРНИХ ДЖЕРЕЛ

6. Маслов А.О. Інформаційна економіка: становлення, структура та теоретичне осмислення : монографія. 2-ге вид., випр. і доп. Київ : ВПЦ "Київський університет", 2016. 512 с.

7. Нестеренко О.В., Савенков О.І., Фаловський О.О. Інтелектуальні системи підтримки прийняття рішень: Навч. посібн. / За ред. П.І. Бідюка. Київ: Національна академія управління, 2016. 188 с. ISBN 978-966-8406-94-2.

8. Бідюк П.І., Тимощук О.Л., Коваленко А.Є., Коршевнюк Л.О. Системи і методи підтримки прийняття рішень : підручник. Київ : НТУУ КПІ ім. Ігоря Сікорського, 2022. 610 с.

9. Хорошун В.В. Системи підтримки прийняття рішень. Навчально-методичний посібник для студентів ЗДІА, галузь знань 0305 – «Економіка та підприємництво» напрям підготовки 6.030502 – «Економічна кібернетика». Запоріжжя: Видавництво ЗДІА, 2012. 150 с.

10. Бідюк П.І., Коршевнюк Л.О. Проектування комп'ютерних інформаційних систем підтримки прийняття рішень: Навчальний посібник. Київ. ННК "ІПСА" НТУУ "КПІ", 2010. 340 с.

11. Демиденко М.А. Системи підтримки прийняття рішень : навч. посіб. Нац. гірн. ун-т. Дніпро, 2016. 104 с. ISBN 978-912-350-293-9.

12. Бідюк П.І., Кузнєцова Н.В., Терентьєв О.М. Система підтримки прийняття рішень для аналізу фінансових даних. Наукові вісті Національного технічного університету України "Київський політехнічний інститут". 2011. №1. С. 48-61. URL: http://nbuv.gov.ua/UJRN/NVKPI_2011_1_9.

13. Рузакова О.В. Система підтримки прийняття рішень у задачах фінансового аналізу. Агросвіт. 2019. №5. С. 67–72. DOI: 10.32702/2306-6792.2019.5.67.

14. Шафоренко І.Ю. Система підтримки прийняття рішень (СППР) як інструмент управління рішеннями в галузі ІТ в умовах невизначеності. Наука і

техніка сьогодні. 2023. №3(17). С. 175-188. URL: https://doi.org/10.52058/2786-6025-2023-3(17)-175-188.

15. Андрющенко Т.Ю. Автоматизація та системи підтримки прийняття рішень на поліграфічних підприємствах. Системи обробки інформації. 2010. Вип. 7. С. 134-141. URL: http://nbuv.gov.ua/UJRN/soi_2010_7_32.

16. Климчук С.О. Розроблення прецедентної системи підтримки прийняття рішень для діагностики мостових кранів. SISN. 2010. Вип. 689. №2. С. 169-176.

17. Пакет SAP BusinessObjects Business Intelligence. SAP. URL: https://www.sap.com/ukraine/products/technology-platform/bi-platform.html.

18. What is Power BI?. Microsoft. URL: https://learn.microsoft.com/uk-ua/power-bi/fundamentals/power-bi-overview.

19. IBM Planning Analytics. IBM. URL: https://www.ibm.com/products/planning-analytics.

20. Tableau. Tableau. URL: https://www.tableau.com/products/tableau.

21. Клієнт-серверна архітектура. QATestLab training center. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/.

22. Організація баз даних. Тема 6 – Теорія нормалізації реляційної моделі даних. Сумський державний університет. URL: https://elearning.sumdu.edu.ua/free_content/lectured:89b3d175c06a6b137e410cb14821d0e94549ad5a/20151013153156/44494/index.html.

23. Бутрин Л., Маєвський О. Реляційна модель бази даних. Матеріали VIII Всеукраїнської студентської науково - технічної конференції / В 2 т. Тернопіль: Тернопільський національний технічний університет ім. І. Пулюя (м. Тернопіль, 23-24 квітня 2015 р.), 2015. Т. 1. С. 119-120.

24. Моделі даних. Реляційна модель даних. Луцький національний технічний університет. URL: https://elib.lntu.edu.ua/sites/default/files/elib_upload/BD_2016_3/page5.html.

25. HTML, CSS і JavaScript: основи веб-розробки. Pravda твого міста. URL: https://pravda.if.ua/html-css-i-javascript-osnovy-veb-rozrobky/.

26. What is PHP? The PHP Group. URL: https://www.php.net/manual/en/intro-whatis.php.

27. Голуб Б.Л., Баранова Т.А. Методичні рекомендації щодо підготовки та оформлення кваліфікаційної магістерської роботи. Національний Університет біоресурсів і природокористування України. Київ, 2021. 51 с.