Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Системотехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження та аналіз попиту для SRM-системи складського обліку кондитерського виробництва
(тема)

Виконав:
студент _6_ курсу, групи ___ІТПм-21-1_____
Савченко С.С.
(прізвище, ініціали)

Спеціальність _122 Комп'ютерні науки_____
(код і повна назва спеціальності)
Тип програми ____освітньо-професійна_____
Освітня програма _Інформаційні технології проектування__
(повна назва освітньої програми)

Керівник ____проф. Колесник Л.В_____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ __Гребеннік І.В___
(підпис) (прізвище, ініціали)

2022 р.

Я як студент ХНУРЕ розумію і підтримую політику закладу із академічної доброчесності. Я не надавала і не одержувала недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

14.12.2022 Савченко С.С.

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.

Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.

Попередній захист проведено 14 грудня 2022 р.

Керівник кваліфікаційної роботи проф. Колесник Л.В.

Харківський національний університет радіоелектроніки

Факультет __Комп'ютерних наук_____

Кафедра ____Системотехніки_____

Рівень вищої освіти _____другий (магістерський)_____

Спеціальність_122 – Комп'ютерні науки_____
(код і повна назва)

Тип програми _освітньо-професійна_____

Освітня програма __Інформаційні технології проектування_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _____Савченко Сергію Сергійовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи _Дослідження та аналіз попиту для SRM-системи складського обліку кондитерського виробництва_____
затверджена наказом університету від _21_ _листопада__2022 р. № __1504 СТ__
2. Термін подання студентом роботи до екзаменаційної комісії _15_ __грудня__ 2022 р.
3. Вихідні дані до роботи Дослідити та проаналізувати методи аналізу попиту для сфери кондитерського виробництва для послідуючої програмної реалізації та впровадження до вже реалізованої системи. Система складається з двох частин: серверної та клієнтської. Серверна частина являє собою реалізацію бази даних. Клієнтська частина включає реалізацію функцій користувачів системи, як основних покупців, і поставників продукції до фабрики. Додатковий модуль такого додатку має аналізувати попит на ринку з урахуванням факторів впливу
4. Перелік питань, що потрібно опрацювати в роботі 4.1 Вступ. 4.2 Аналіз предметної області. 4.3 Математичний опис задачі. 4.4 Опис прийнятих проектних рішень при розробці системи. 4.5 Тестування системи. 4.6 Висновки. 4.7 Перелік джерел посилання
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) 5.1 Концептуальна діаграма (1 аркуш формату А4). 5.2 Діаграма декомпозиції концептуальної діаграми (1 аркуш формату А4). 5.3 Діаграма декомпозиції процесу «Прогнозування ціни» (1 аркуш формату А4). 5.4 Діаграма потоків даних. Перший рівень (1 аркуш формату А4). 5.5 Діаграма потоків даних. Другий рівень (1 аркуш формату А4). 5.6 Діаграма класів (1 аркуш формату А4). 5.7 Діаграма послідовності (1 аркуш формату А4). 5.8 Криві, характеризуючі еластичність попиту (1 аркуш

формату А4). 5.9 Залежність еластичності товарів від зміни ціни (1 аркуш формату А4). 5.10 Приблизний графік реальної еластичності товарів (1 аркуш формату А4). 5.11 Робота системи з архітектурою «модель-вид-контролер» (1 аркуш формату А4). 5.12 Класична структура API (1 аркуш формату А4). 5.13 Фізична діаграма даних (1 аркуш формату А4). 5.14 Алгоритм послідовної поведінки постачальника (1 аркуш формату А4). 5.15 Алгоритм послідовної поведінки адміністратора та споживача (1 аркуш формату А4). 5.16 Уявлення «візуалізація результатів аналізу» (1 аркуш формату А4). 5.17 Результат навантажувального тестуваннння розробленого додатку для srm-системи (1 аркуш формату А4).

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина			

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання кваліфікаційної роботи	03.10.2022	Виконано
2	Аналіз предметної області та визначення основних бізнес-функцій інформаційної системи	04.10.2022	Виконано
3	Аналіз існуючих систем та проектування майбутнього додатку	10.10.2022	Виконано
4	Аналіз методів для аналізу попиту товарів на виробництві	20.10.2022	Виконано
5	Огляд та розробка технологій для практичної реалізації проєкту	07.11.2022	Виконано
6	Розробка проектних рішень по програмному та інформаційному забезпеченню системи	10.11.2022	Виконано
7	Тестування розроблення програмного забезпечення	25.11.2022	Виконано
8	Оформлення пояснювальної записки	01.12.2022	Виконано
9	Подача кваліфікаційної роботи на допуск до захисту	15.12.2022	Виконано
10	Підготовка докладу до захисту кваліфікаційної роботи	19.12.2022	Виконано
11	Подання кваліфікаційної роботи	20.12.2022	Виконано

Дата видачі завдання _03_ ____жовтня___2022 р.

Студент _____Савченко С.С._____
(підпис)

Керівник роботи _____ ____проф. Колесник Л.В_____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 139 с., 24 рис., 7 таблиць, 13 формул, 28 джерел інформації.

SRM-СИСТЕМА, БАЗА ДАНИХ, MYSQL, ІНФОРМАЦІЙНА СИСТЕМА, ІНТЕРФЕЙС ДОСТУПУ, СИСТЕМА ОБЛІКУ, ЕЛЕКТРОННА КОМЕРЦІЯ, ПАТЕРН MVC, ЦІНОВА ЕЛАСТИЧНІСТЬ ПОПИТУ, ПОПИТ, МНОГОКРИТЕРІАЛЬНИЙ ВИБІР, СТАТИСТИЧНИЙ АНАЛІЗ

Об'єктом дослідження кваліфікаційної роботи є один з основних бізнес-процесів SRM-системи для кондитерської фабрики, який є цільовим процесом для отримання прибутку – це складський облік мережі кондитерських.

Предметом досліджень кваліфікаційної роботи є методи та підходи до аналізу попиту виробництва та послідуючого використання статистичних даних для збільшення прибутковості такої системи.

Мета досліджень – підвищення ефективності прийняття рішень щодо аналізу попиту ринку збуту кондитерської продукції за рахунок комбінації методів лінійної регрессії, корегування помилок на основі дат та методу цінової еластичності попиту.

Методи дослідження: лінійна регрессія, корегування помилок, цінова еластичність попиту. Запропоновано етапи проведення прогнозування за допомогою методу аналізу цінової еластичності.

У ході написання кваліфікаційної роботи були проведені наступні етапи: детальний аналіз обраної предметної області, дослідження альтернативних методів аналізу попиту ринку збуту кондитерської продукції, розробка модифікованого модулю складського обліку, функціональне моделювання, практична реалізація інформаційної системи та додаткових модулів.

Результати роботи – інформаційна ситема, що спроможна аналізувати та прогнозувати ефективні ціни для покращення прибутковості виробництва., а також мати усі функціональні потреби для роботи такого типу виробництва.

ABSTRACT

Explanatory note to the qualification work: 139 pages, 24 fig., 7 tables, 13 formulas, 28 information sources.

SRM-SYSTEM, DATABASE, MYSQL, INFORMATION SYSTEM, ACCESS INTERFACE, ACCOUNTING SYSTEM, E-COMMERCE, MVC PATTERN, PRICE ELASTICITY OF DEMAND, DEMAND, MULTICRITERIA SELECTION, STATISTICAL ANALYSIS

The object of the qualification work is one of the main business processes of the SRM-system for a confectionery factory, which is a target process for making a profit - it is the warehouse accounting of the confectionery chain.

The subject of research qualification work are methods and approaches to the analysis of production demand and the subsequent use of statistical data to increase the profitability of such a system.

The purpose of the research is to increase the efficiency of decision-making on the analysis of demand for confectionery products by combining linear regression methods, error correction based on dates and the method of price elasticity of demand.

Research methods: linear regression, error correction, price elasticity of demand. The stages of forecasting using the method of price elasticity analysis are proposed.

In the course of writing the qualification work the following stages were carried out: detailed analysis of the selected subject area, research of alternative methods of demand analysis of the confectionery market, development of a modified module of warehouse accounting, functional modeling, practical implementation of the information system and additional modules.

The results of the work are an information system that is able to analyze and forecast effective prices to improve the profitability of production, as well as have all the functional needs for this type of production.

ЗМІСТ

ВСТУП

Зрозуміло, що в останнє десятиліття веб зазнає значних темпів підйому репутації серед людей, що мають різний соціальний стан, хоч якого б віку і національностей вони не були. Веб вважається дуже комфортним і швидким засобом передавання інформації і при цьому досить доступним. З кожним роком в Україні чисельність користувачів мереж веб зростає порівняно з попереднім. Вже задовго до нашого часу в інтернеті пішов ажіотаж на інтернет-магазини, найбільшою перевагою якого заведено вважати те, що покупки дозволено робити, не виходячи з дому.

Становище ринку кондитерських продуктів в Україні останніми роками визначалося 2-ма обставинами. По-перше, українці обожнюють солодке, і тому на ринку величезна кількість виробників різної кондитерської продукції. По-друге, кондитерські продукти все-таки ніяк не входять до списку першочергових товарів харчування, що спричиняє зменшення їхнього вживання під час фінансових потрясінь.

Під час проведення аналізу виявляємо, ніби інтернет-магазини кондитерських продуктів вважаються дуже популярними. Серед великих інтернет-магазинів в Україні дозволено відзначити досить чимало різних конкуруючих компаній, одні з них напевно: "Солодкий мир" [3], "Світоч" [4], "АВК" [5]. Однак, якщо розглянути весь кондитерський бізнес найглибше, то можемо відзначити певну кількість наймасштабніших проблем. Напевно поставки, робота з, конкретно, постачальниками, а також робота з тими, хто повинен збути продукцію. Але перед тим як займатися будь-яким плануванням такого виробництва треба враховувати, наскільки такий бізнес буде фінансово-вигідний та наскільки рентабельна ця сфера. З цією задачею нам допоможе глибинний аналіз ринку збуту такого виробництва.

Через актуальність даної проблеми було вирішено визнати метою роботи підвищення ефективності прийняття рішень щодо аналізу попиту ринку збуту кондитерської продукції за рахунок комбінації методів лінійної регрессії, корегування помилок на основі дат та методу цінової еластичності попиту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області, яка визначається діяльністю підприємства

Кондитерська фабрика - підприємство, зайняте створенням і збутом кондитерських продуктів у середніх і великих масштабах. Досить нерідко процеси виробництва кондитерського продукту досить розрізнені один від одного, тому дослідження системи, яка об'єднає всі галузі - пріоритетна мета для власників такого бізнесу. Для цього було підібрано систему SRM, що дасть змогу керувати практично всіма частинами компанії в одній системі. Предметною областю наданої кваліфікаційної роботи вважається «Дослідження та аналіз попиту для SRM-системи складського обліку кондитерського виробництва».

Програмне рішення управління взаємовідносинами з постачальниками (SRM) - напевно, найкращий вибір для виробників і торговців упродовж усього ланцюжка постачань.

На прикладі того, як керування взаємовідносинами з покупцями (CRM) покликане відрегулювати і поліпшити процеси між виробником і його покупцями, SRM спрямоване на формування вигідних відносин між організацією та її постачальниками, насамперед з тими яких вважають більш стратегічно необхідними для організації. Він ще покликаний сприяти збільшенню якості, віддачі та інтеграції інновацій. Впровадження SRM діє не просто на економію витрат, а саме на максимізацію значення постачальників для отримання конкурентноздатної переваги[6].

У сучасних критеріях жорсткої конкурентної боротьби успішна активність компанії харчової індустрії неможлива за відсутності прогнозування попиту, що дозволяэ сформувати обсяг, структуру, ступінь попиту, можливі тенденції його конфігурації, коло причин, що характеризують попит у послудуючому періоді та перспективи розвитку компанії харчової індустрії. Одночасно передбачення

попиту вважається важливим аспектом доцільних інвестицій у створення продуктів. Тому прогнозування і передбачення споживчого попиту на продукти, що виробляються підприємством, вважається важливим, а введення таких моделей дасть змогу збільшити економічну результативність роботи компанії харчової індустрії. Передбачення попиту передбачає визначення обсягу продукції в кількісному і вартісному вираженні на конкретні періоди часу в наступному визначеному періоді часу. Головні способи моделювання поділяють за 2 показниками: ступенем свободи моделювання від суб'єктивності до об'єктивності, і найбільшим або найменшим ступенем аналітичності цього процесу.

Одна з цілей SRM-системи – налагодження вигідних відносин з постачальниками [7]. Така система дозволяє автоматично обробляти і відправляти замовлення та запити на закупівлю. Це дає можливість компаніям зменшити час обробки. Ручна подача заявки, складання технічного завдання і дотримання всіх регламентів може займати до 2 місяців і навіть більше. Автоматизація істотно спрощує всі процеси і скорочує час до декількох днів.

Впровадження SRM системи на будь-яке підприємство – досить складний процес. Він потребує грошових затрат і не всі компанії готові йти на такі дії. Але ця інвестиція швидко виправдовується, тому що істотно знижуються витрати на логістику, організацію поставок, оптимізується безліч процесів. SRM це є комплексний підхід, що передбачає налагодження процесів роботи з постачальниками [7]. Суть методу полягає в розробці стратегії. Система ж дозволяє впровадити цю стратегію у виробництво.

Крім того, система оцінює ризики, збирає дані про постачальників, розділяє їх на класифікації, веде реєстр вже укладених угод. При виникненні спірних ситуацій за допомогою CRM системи розробляються регламенти [7]. Вона допомагає підбирати методологію роботи з ключовими партнерами. Це ресурс для повного контролю взаємодії компанії з постачальниками. Для розбору причини конфігурацій ринкової конкуренції та передбачення ймовірних

конфігурацій що залежать від тенденцій наявних на ринку, здійснюють класифікацію причин, які мають більший вплив на попит продукції. Найважливіші причини, які впливають на створення попиту, згруповані в наступні категорії [7]:

а) економічні фактори – містять ціни на товар і базові потреби, їх задовольнення, рівні фінансової спроможності клієнтів, ціни на конкурентні товари, рекламу товару;

б) соціальні фактори – займаються врахуванням соціальних і професійних структур клієнтної сфери, розвиток клієнтських фондів споживання, соціальне забезпечення клієнтів, рівні освітнього розвитку і культури учасників торгівельних відносин та інші;

в) демографічні фактори – враховують частини виробництва і торгівельних відносин шляхом аналізу кількісного і статево-вікового показника населення, відношення між населенням за параметром знаходження, розміри і склади сімей та інше;

г) природно-географічні фактори – враховують клімат, тривалість кожного з сезонів, регіональні споживчі особливості;

ґ) традиційно-культурні фактори – вираховуються з побутових та історико-сформованих звичок, традицій та релігій тощо.

Взяти до уваги всі фактори впливу за кожним з товарів здебільшого неможливо, тому їх варто об'єднати до якоїсь структури, що дозволить зробити оцінку та отримати результати їх впливу на попит такої продукції.

Аналіз ринку сучасних товарів визначається урахуванням основних напрямків його підйому, тому фактори, що визначають цей розвиток, дотриманні у цілеспрямованій об'єктивності та ретельній оцінці даних і результатів передбачень [8].

1.2 Аналіз реалізованих систем та методів дослідження

За для більш детального розуміння області потрібно зробити аналіз ринку конкурентів в досліджуваній сфері. У ході такого аналізу ринку за приклад була обрана компанія «Бісквіт-Шоколад»[9]. Вони мають свій сайт каталогу, а також там знаходиться пункт для взяття участі у тендерній программі під назвою «Співпраця». Там можна стати постачальником продукції до підприємства або постачати вже готову продукцію.

Проаналізувавши їх систему більш детальніше можна основні недоліки їх розробки:

– пункт «Співпраця» являє собою доступ до тендерної системи, що не є як такової співпрацею з фабрикою;

– система дозволяє займатися збутом продукції, як, наприклад, у інших інтернет-магазинах виробниках кондитерської продукції;

– вони не впроваджують роботу з кур'єрськими фірмами та не мають функції доставки їх продукції навіть у невеликих масштабах.

При розробці успішної системи слід врахувати ризик того, що при розробці такого програмного забезпечення можуть виникнути деякі проблеми, що й будуть названі далі «ризиками».

Враховуючи перераховані пункти, можна виділити основні напрямки, котрі зроблять нашу систему більш кращою за конкурентів. При аналізі кожної проблеми такої системи вираховується ймовірність її прояву й збиток, що вона може нести. Нажаль, простих методів виконання передбачення ризиків ще не вигадали тому, значною мірою такі способи засновані на думці й досвіді людини котра керує системою. Якщо не орієнтуватися на виняткову точність, можна привести подібну шкалу ймовірностей ризиків і їх наслідків:

– дуже низька, якщо у них значення менш 10%;

– низька, якщо їх значення від 10 до 25 %;

– середня при значеннях від 25 до 50%;

– висока, якщо їх значення коливається від 50 до 75%;

– дуже високою при значеннях більше 75%.

Можливі збитки від таких ризиків та ситуацій можна підрозділити на «» (катастрофічна), «жовтий рівень» (серйозний), «зелений рівень» (терпимий) і «сірий рівень» (незначний) [10].

Результати аналізу ризиків представлені у вигляді таблиці ризиків, відсортованих за ступенями можливого збитку для підприємства. У табл. 1.1 наведений такий перелк ризиків, а також зазначені ймовірності їх виникнення.

Таблиця 1.1 – Перелік ризиків після детального аналізу

Приблизний ризик	Імовірність виникнення	Ступінь збитку
Фінансові труднощі, що призвели до зменшення бюджету проекту	Низька	Червоний рівень
Неможливість підбору працівників з необхідним професійним рівнем	Висока	Червоний рівень
Програмні компоненти, що використані з старої системи, мають проблеми, що обмежують їх можливості	Середня	Жовтий рівень
Часті зміни вимог, що призводять до перепроектування системи	Середня	Жовтий рівень
Відбулася реорганізація у компанії котра розроблює ПЗ, у результаті чого були змінені пріорітети керування проектом	Висока	Жовтий рівень
Вже використовувана база даних, не забезпечує обробку достатього обсягу транзакцій	Середня	Жовтий рівень
Недооцінка часу на виконання усіх пунктів проекту	Висока	Жовтий рівень

Продовження таблиці 1.1

Приблизний ризик	Імовірність виникнення	Ступінь збитку
Інтеграція CASE-засоби неможлива у зв'яку з використанням інших засобів підтримки проекту	Висока	Зелений рівень
На останніх етапах розробки проекту було виявлено нечітке формулювання користувацьких вимог, що призвело до суттєвих змін системних вимог	Середня	Зелений рівень
Неможливо організувати навчання персоналу до достатнього рівня	Середня	Зелений рівень
Дефекти виявляються у системі з незапланованою швидкістю (менше ніж задана)	Середня	Зелений рівень
Значна зміна розміру готової системи вцілому	Висока	Зелений рівень
Погано-сгенерований код, що сгенеровано CASE-засобами розробки, котрий виявився неефективним	Середня	Сірий рівень

1.3 Методи аналізу попиту на ринку кондитерської продукції

Технології ланцюжка поставок, особливо ті, що мають відношення до прогнозування попиту і запасів, використовують штучний інтелект і машинне навчання і працюють тим точніше і якісніше, чим більше даних ви їм даєте. Але не варто покладатися тільки на дані про минулі події, наприклад на дані про продажі або ефективність виробництва. У наші дні для ефективного аналізу дані

не обов'язково мають бути лінійними і простими. Сучасні інструменти управління даними дають змогу контролювати й обробляти великі та складні набори даних.

Концепцію передбачення попиту на продукцію будь-якого підприємтсва кондитерського виробництва економіко-математичними методами будемо вважати у вигляді слідуючого алгоритму [8]:

а) визначення факторів, котрі включені у модель та впроваджують найсуттєвіші особливості моделі. Їх характеристики зводимо до загального виду за одиницями виміру, цінами, тощо. Оскільки попит на кожен товар є залежним від 2-х або більше факторів, тоді багатофакторне моделювання відбувається диференційовано, отже для кожного з товарів у модель вводяться зовсім різні аргументи. У найбіль загальному вигляді такі залежності між попитом (його величчю) і його факторами формулюють наступним чином:

$$Y = f(\alpha, \alpha 1 \dots \alpha n, t, \mathrm{W}, \mathrm{Z}, \mathrm{N}, F, \mathrm{T}, X), \qquad (1.1)$$

де Y – велич попиту на товар, що популюється виробництвом;

α – ціна одного товару; α1 ... αn – ціни на усі інші товари; t – час;

W – фінансова вигода споживача;

Z – спроможність споживача;

N – вподобання споживачів;

F – очікування, щодо майбутніх цін;

T – кількість покупців;

X – інші фактори.

Чим більше факторів фігурує у моделі тим більше буде її наближеність до реальності, але це значно збільшує трудомісткість розробки та уможливлює мультиколінеарність факторів. При цьому основним критерієм точності моделі є більший обсяг спостережень до попиту на товар. Простішими моделями є однофакторні, для прикладу: трендові чи адаптивні моделі (Брауна, Хольта,

Тейла – Вейджа) у котрих тільки один фактор впливу – час, але такі моделі не з'ясовують причини зміни попиту, тому за допомогою них робиться тільки короткострокове прогнозування. При використанні адаптивних моделей головним чинником є відмінна економічна цінність кожного з рівнів динамічного ряду та результати попереднього кроку прогнозу. У структурних моделях попит представляється функцією, яка залежна тільки від доходу. Якщо ми знаємо частоту розподілу клієнтів за рівнем доходу, такі моделі відразу дають змогу розраховувати головну побудову попиту;

б) встановлення математичної залежності між попитом та кожним обраним фактором та визначення модельованих параметрів за методом найменших квадратів;

в) перевірка моделі на адекватність та точність. Модель є адекватною тоді і тільки тоді, коли її залишкова компонента задовольняє властивостям випадкової компоненти [8]:

– за критерієм Кендела

– за критерієм Серій;

– за критерієм Дарбіна – Уотсона.

– за показниками асиметрії та ексцесу;

– пілся перевірки на основі F-критерію Фішера;

Якщо модель є адекватною, для формулюється задача оцінки точності. Для такою задачі статистичні показники точності: коефіцієнт детермінації, середньоквадратичне відхилення, середня відносна помилка апроксимації. Усі вони дозволяють обирати з декількох адекватних моделей найбільш точну;

г) побудова прогнозу попиту за допомогою побудованого рівняння регресії, у яке було підстановлено прогнозовані значення факторів, врахованих у обрахованій моделі. Для того, щоб оцінити якості прогнозу моделі доцільно використати ретроспективний прогноз. У такому випадку така оцінка прогнозних якостей моделі є досить корисною при зіставленні інших різних адекватних моделей прогнозування;

г) верифікація моделі, яка є переліком сумісних факторів і методів, що, на основі багатофакторного аналізу дозволяють оцінити якість результативного прогнозу. Одночасно з цим оцінюється метод прогнозування, котрим був здійснений прогноз.

Найбільш доцільним та ефективним методом прогнозування попиту, на нашу думку, є комплексне дослідження обсягів продажу з використанням групи методів. Варто зазначити, що такі дослідження є більш затратними, але в кінцевому результаті їх ефективність буде доведена, оскільки "мінуси" одного методу перекриваються "плюсами" іншого.

1.4 Постановка задачі

Ключовим завданням підприємств-товаровиробників є аналіз потреб споживачів. Керівникам організацій варто здійснювати достатнє фінансування даної роботи, адже від цього залежить стратегічний план управління підприємством, а від кількості проданої продукції залежить прибуток фірми. Основним завданням цієї роботи є дослідження та аналіз попиту для SRM-системи складського обліку кондитерського виробництва, яка може використовуватися в різноманітних сферах кондитерського виробництва, а також для будь якого розміру такого виробництва. На основі аналізу побудованої моделі та розробленого прогнозу попиту будуть розроблені рекомендації щодо управлінських рішень. ПМП (прогнозні моделі попиту) при впроваджені дозволять ОПР враховувати загальні тенденції ринку, конкуренцію на ринку виробництва кондитерських виробів, а також обмеження які встановлює ринок. Такі обмеження накладаються декількома факторами, такими як, наприклад, платоспроможність споживачів та ступінь розвитку цільового, виробництва. Для точності випробувань вхідною інформацією до такої системи повинна стати інформація щодо зміни ціни за останній рік, а віхідною ж буде прогнозована рекомендована ціна на наступний місяць, з урахуванням даних по святам.

2 ОГЛЯД МЕТОДІВ ТА ТЕХНОЛОГІЙ, ЯКІ ЗАСТОСОВУЮТЬСЯ В ПРЕДМЕТНІЙ ОБЛАСТІ

2.1 Розробка системних вимог до розроблюваної системи

В процесі цілісного аналізу системи та ризиків щодо її впровадження було опрацьовано декілька проблем які необхідно враховувати у вимогах до створюваної системи, а саме: багаторівневість авторизації у системі для того щоб розділити користувачів по ролях, формування стандарту аутентифікації котрий забезпечить достатній рівень захисту, шифрування даних які транспортуються по системах, визначити очікуване навантаження на систему яке буде критичним і врахувати цей параметр, визначити розмір сховища даних, розрахувати час на обробку запиту та швидкість відповіді додатку навіть при передачі великих обсягів даних, яке є більш відоме як напрацювання на відказ. Низка цих проблем вирішиться у процесі тестування, інші – в процесі проєктуваня та розробки.

Тому, виходячи з цього сформуємо такий перелік вимог:

– багаторівневий доступ до системи за паролем;

– аналіз та впровадження найнадійнішого та ефективного стандарту захисту паролю та, можливе, використання 2FA (двох-факторної) аутентифікації;

– проектування архітектури системи та схеми доступу до даних;

– визначити методи аналізу попиту ринку збуту;

– інтуїтивно-зрозумілий інтерфейс програми що чітко відображає кожну функцію системи;

– швідкість обробки запитів має бути максимальною;

– обробка запиту на аналіз та дослідження статистики продажу та вже виконаного збуту;

– інструкція користувача з усіма аспектами роботи з розроблюванною системою згідно до ГОСТ 19.505-79.

2.2 Опис поточного стану інформаційної системи підприємства

Уся реєстрація надходження, переміщення продукції на складі відбувається в паперовому вигляді, на складі немає програмного забезпечення для ведення обліку. Усю звітність складають вручну.

Для цього працівник складу записує в журналі інформацію про передпродажну підготовку, прибулі, відвантажені верстати в офіс. На підставі отриманого документа менеджер формує документацію і звітність за кожним клієнтом і на підставі цих даних формує відомість про продажі, купівлю продукції у письмовому вигляді, яка передається директору підприємства.

На початковому етапі роботи підприємства директор не розглядав варіанти впровадження інформаційної системи управління, оскільки ручна робота не становила проблем з обліком через невелику кількість продукції. Нині, у зв'язку зі збільшенням потоку клієнтів і замовлень на верстати, підвищилася складність обліку. Ба більше, ручний спосіб складання звітності не є надійним через можливість помилок, тому що людина може помилитися, і через це на складі можлива низка проблем, як-от невідповідність кількості матеріалів для ремонту або помилково не зазначити важливі дані для коректної й точної роботи складу. Перед керівництвом постало питання про впровадження ІС для автоматизації складського обліку з метою підвищення якості управління діяльністю складу і підприємства в цілому[10].

Аналіз наявної системи обліку дав змогу виявити такі недоліки:

– висока трудомісткість у підрахунку матеріалів, необхідність особистої участі в такому підрахунку, формування звітності про матеріали;

– неможливість надавати повну інформацію про верстати в потрібний момент;

– наявність персонального комп'ютера не дає можливості спростити й автоматизувати роботу складу без потрібної інформаційної системи;

– неможливість надати повну звітність щодо функціонування складу: звіт щодо витрат матеріалу (для передпродажної підготовки), а також про отримані та збережені верстати на складі.

Управління підприємством будь-якого виду діяльності потребує обов'язкової автоматизації. На багатьох підприємствах використовують стандартні програми для управління.

На сьогодні немає грамотного обліку на складі, який дає змогу дуже швидко отримати інформацію про кількість продукції на складі, продукції відремонтованих, що дуже важливо для роботи підприємства, управління ним.

Таким чином, можна зробити висновок про те, що наявна на складі практика обліку є незручною, трудомісткою і малопродуктивною в поточних умовах підприємства і потребує заміни.

2.3 Моделювання та SWOT-аналіз

Після ретельного аналізу системи можуть бути визначені і розроблені функціональні вимоги до компонентів розроблюванної SRM-системи. Побудована вони будуть з використанням станадрту IDEF0. Модель SADT, що включає IDEF0, є переліком діаграм, що, з кожним рівнем обробки, розбивають один складний об’єкт на багато складових частин та подані у вигляді блоків[11].

Перш за все розролюється концептуальна діаграма (рис 2.1). Вона побудована з точки зору менеджера. Тут можна побачити основний бізнес-процес та інтерфейси які приймають участь в роботі ІС. Тут представлені такі інтерфейси:

– керуюча інформація, що є, по суті, документацією до процесу

а) порядок робіт;

б) каталог товарів.

– інформація для обробки, які вводить безпосередньо користувач системи

а) переваги клієнта.

б) дані цін у проміжку часу.

– результати роботи інформаційної системи

а) сповіщення користувачу;

б) готове замовлення.

– механізми (це може бути як людина, так і система безпосередньо)

а) менеджер/адміністратор;

б) інформаційна система;

в) користувач;

На схемі декомпозиції концептуальної діаграми (рис 2.2), можна побачити такі блоки:

– враховувати замовлення;

– сформувати корзину;

– сформувати замовлення;

– прогнозування ціни;

– обробити замовлення.

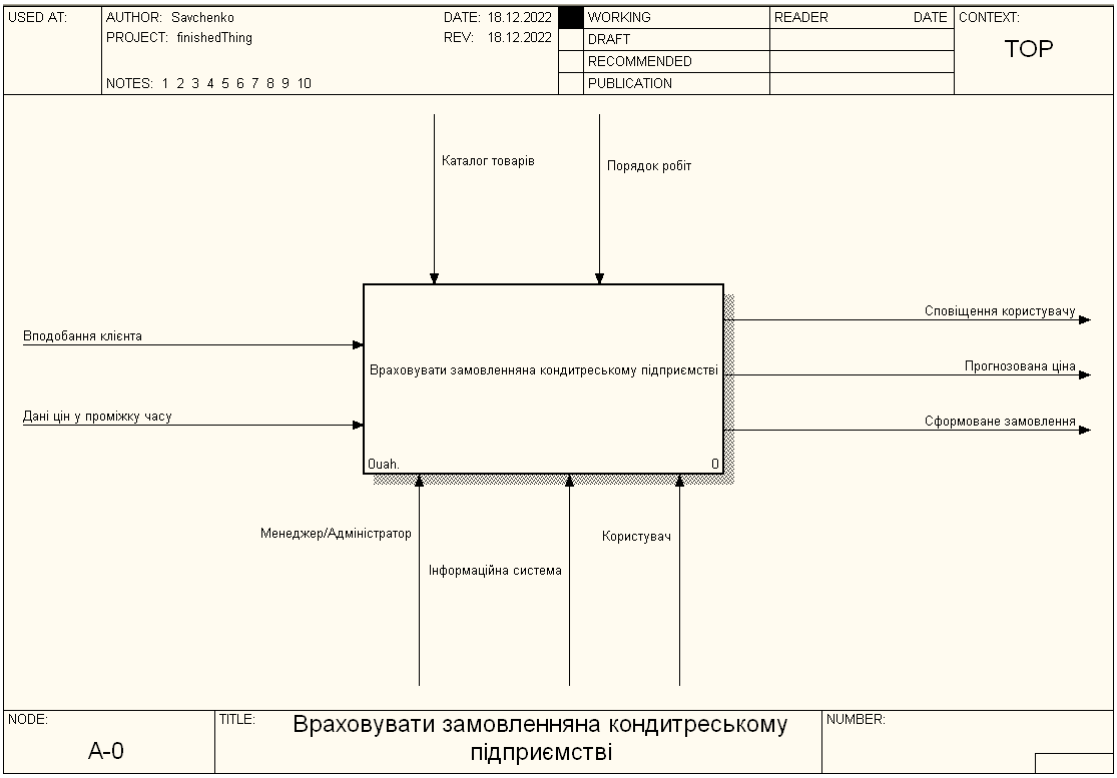

Рисунок 2.1 – Концептуальна діаграма

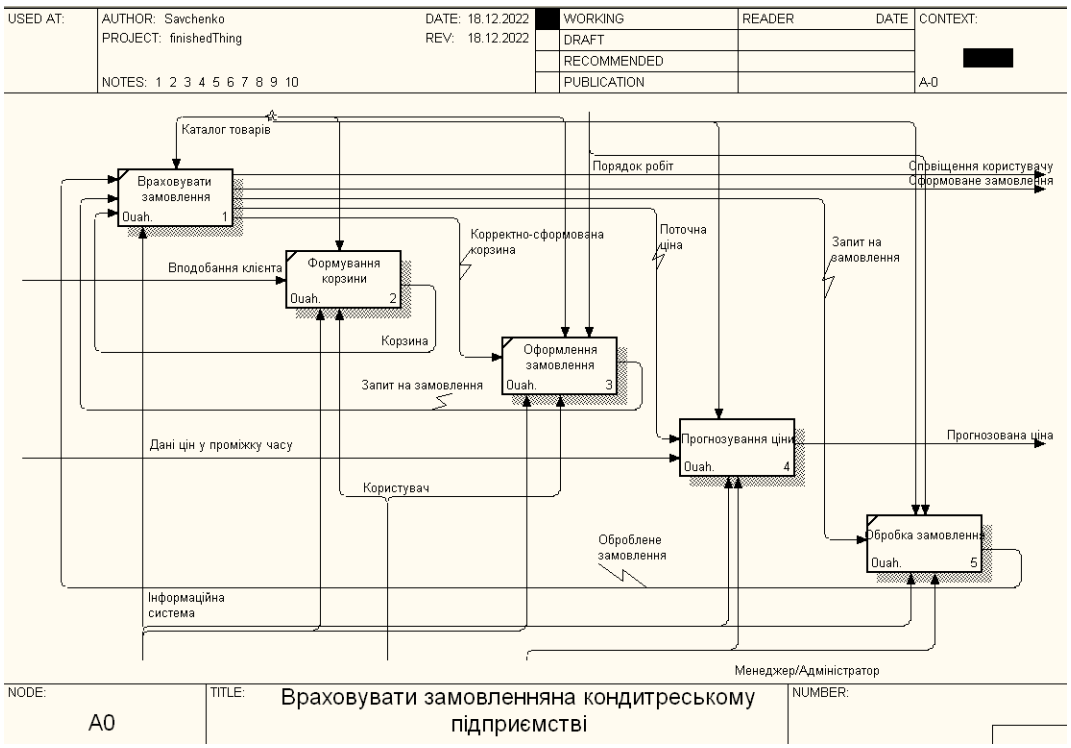

Рисунок 2.2 – Діаграма декомпозиції бізнес-процесу «Враховувати замовлення»

Як ми бачимо на декомпозиції концептуальної діаграми, ми маємо об’єкт під назвою «Прогнозування ціни». Це відображення модулю для дослідження ціни в який поступає інформація з-зовні про дані цін за певний проміжок часу, вони будуть завантажені адміністратором, а також буде враховуватись увесь каталог товарів, щоб обрати певні категорії для прогнозування. На наступній схемі (рис 2.3) можна побачити дкомпозицію цього процесу.

Проаналізувавши отриману діаграму декомпозиції процесу «Прогнозування ціни» можна побачити що цей блок виконується у три етапи. Перший – аналіз цін за минулий час, що буде виконано за допомогою лінійної регрессії. Другим етапом буде корегування помилок та викидів у результатах першого процесу. Та останній і найголовніший етап – прогнозування ціни за допомогою методу цінової еластичності. Уся ця сукупність процесів взаємозалежна і не може бути виключена з основного блоку. В іншому випадку це призведе до невірніх прогнозування, а ,в подальшому, до втрати прибутку.

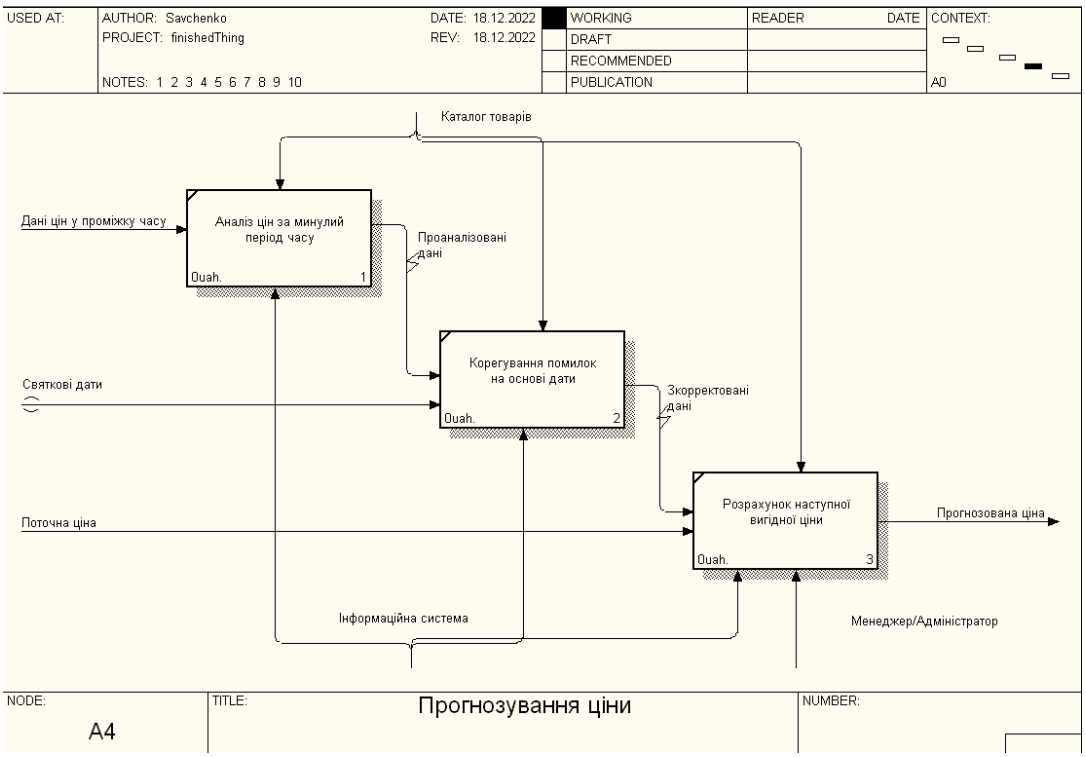

Рисунок 2.3 – Декомпозиція процесу «Прогнозування ціни»

Перед впровадженням інформаційної системи необхідно провести аналіз наявної інформаційної системи.

SWOT-аналіз - це інструмент стратегічного аналізу та планування, який застосовують для оцінки явищ і чинників, що впливають на компанію або проєкт [12].

SWOT-аналіз дає змогу виявити і структурувати сильні та слабкі сторони складу, а також потенційні можливості та загрози. Досягається це за рахунок порівняння внутрішньої сили і слабкості з можливостями, які дає їм ринок. Виходячи з якості відповідності робиться висновок про те, в якому напрямку організація має розвивати свій бізнес.

Об'єктом аналізу є склад. Предметом аналізу - процес обліку на складі. SWOT розшифровується:

– Strengths - сильні сторони.

– Weaknesses - слабкі сторони.

– Opportunities - можливості.

– Threats - загрози.

Головною перевагою такого аналізу є оцінка, чи володіє компанія внутрішніми силами та ресурсами, щоб реалізувати наявні можливості та протистояти зовнішнім загрозам.

Перед тим як приступити до складання SWOT-аналізу, необхідно засвоїти низку правил:

а) Вибір максимально конкретної області дослідження, якщо обрано занадто широку область, то висновки будуть не конкретними і слабо застосовними.

б) Чіткий поділ елементів SWOT, що передбачає, плутанину сильних сторін та можливостей самих елементів. Слабкі чи сильні, немає різниці, сторони - це особливості організації які їй підвладні. Можливості ж пов'язані більш із зовнішнім середовищем і, безпосередньо, непідвладні організації. Організація ж у свою чергу може лише змінювати підхід і підлаштовуватися.

в) Уникайте суб'єктивності. Наївно буде спиратися на вашу думку,якщо з нею не згоден ринок. Можливо, ви вважаєте ваш продукт унікальним, але про це варто насамперед запитати споживачів. Без них ваша особиста думка не має жодного значення.

г) Намагайтеся використовувати думку якомога більшої кількості людей. Що більша вибірка, то точніше дослідження.

г) Необхідно використовувати максимально конкретні й точні формулювання.

У процесі проведення аналізу внутрішньої ситуації підприємства оцінюються його бізнес-процеси, конкурентоспроможність і ресурси.

Проаналізуємо поточний стан обліку на складі ТОВ нашого виробництва до впровадження автоматизованої системи (таблиця 2.1).

Таблиця 2.1 – Аналіз ситуації до впровадження розроблюваної системи

Внутрішні сильні сторони (S)	Внутрішні слабкі сторони (W)
1. Інформація доступна на паперових носіях і не залежить від технічного стану техніки.	Висока ймовірність втрати даних. Не реалізовано жодних звітів для аналізуроботи складу. Швидкість і якість бізнес-процесу на пряму залежить від фахівця.
Зовнішні можливості (O)	Зовнішні загрози (T)
1. Бізнес-процес дав змогу фіксуватироботу складу.	Недоліки в процесі збору, занесення,зберігання інформації. Втрата даних під час знищення паперових носіїв.

Далі розглянемо ситуацію в розрізі тих самих чинників після впровадження системи (таблиця 2.2).

Таблиця 2.2 – Аналіз ситуації після впровадження системи

Внутрішні сильні сторони (S)	Внутрішні слабкі сторони (W)
Підвищення якості обліку складу. Зручний інтерфейс. Автоматизація основних функцій працівника складу. Можливість проводити аналіз. Збільшення швидкості роботи персоналу.	Первинні дані про верстати вносятьсявручну. Потрібне навчання персоналу.
Зовнішні можливості (O)	Зовнішні загрози (T)
Скорочення часу на роботу з обробкою і внесенням первинної інформації. Використання сучасного програмового засобу.	Конкуренція на ринку ПЗ.

Як показує порівняльний аналіз таблиць 2.1 і 2.2 при впровадженні інформаційної системи з обліку складу компанії ТОВ нашого виробництва автоматизуються деякі функції з ведення документації, з'явиться можливість складання звітності, для аналізу роботи складу. Таким чином, SWOT-аналіз виявив необхідність увпровадженні нової інформаційної системи.

2.4 Розробка моделі даних розроблюваної системи

Діаграма потоків даних (DFD) відображає потік інформації для будь-якого процесу або системи. Вона використовує визначені символи, такі як прямокутники, кола та стрілки, а також короткі текстові мітки, щоб показати вхідні та вихідні дані, точки зберігання та маршрути між кожним пунктом призначення. Схеми потоків даних можуть варіюватися від простих, навіть намальованих від руки оглядів процесів, до поглиблених, багаторівневих DFD, які поступово заглиблюються в те, як обробляються дані. [11].

Головна функція розроблюваної системи є облік та архівування поставок товару зі складу мережі, а також архівація поточних цін продукції для послідючого вилучення та аналізу. Інформаційна система має п’ять основних функцій:

– «складський обіг продукції»;

– «обробка заявок (на поставку, замовлення тощо)»;

– «формування переліку позицій»;

– «адміністрування»;

– «архівування та збереження цін для послідуючого вилучення».

Перш за все побудуємо контекстну DFD-діаграму. За основний процес буде відповідати блок «Складський облік мережі магазинів». В ньому приймають участь усі зовнішні сутності що впроваджені до системи, але як і планувалося вони усі розгалужені за ролями, тож мають окремий доступ до кожної функції. Ці сутності це: мережа магазинів, поставник та адміністратор.

Після планування отримали концептуальну діаграму (рис 2.4) на якій розміщенні необхідні сутності та відображені потоки даних у системі. Але, звичайно для більш детального опису системи потрібна декомпозиція цієї діаграми. Вона також була проведена і відображена на рисунку 2.5.

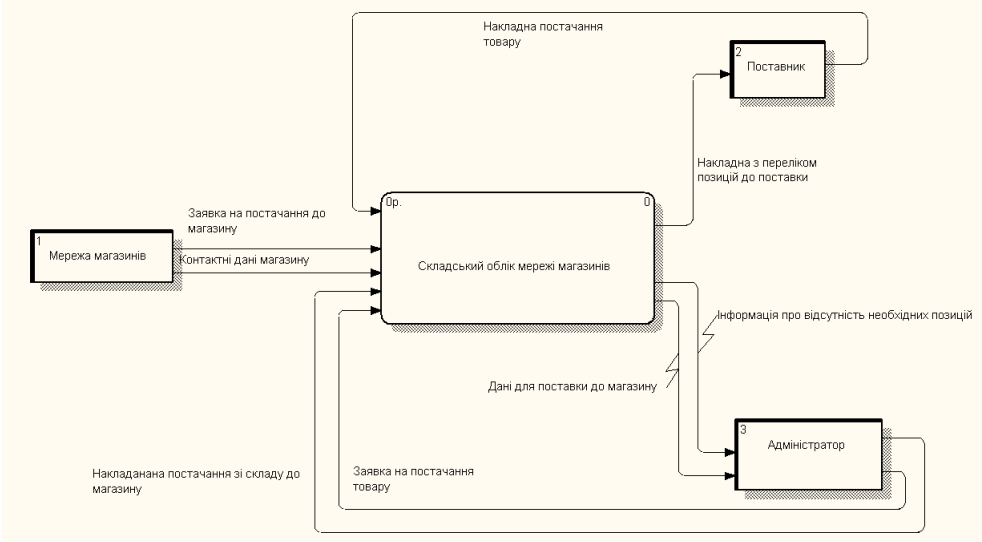

Рисунок 2.4 – Основний процес «Складський обіг фабрики кондитерських виробів»

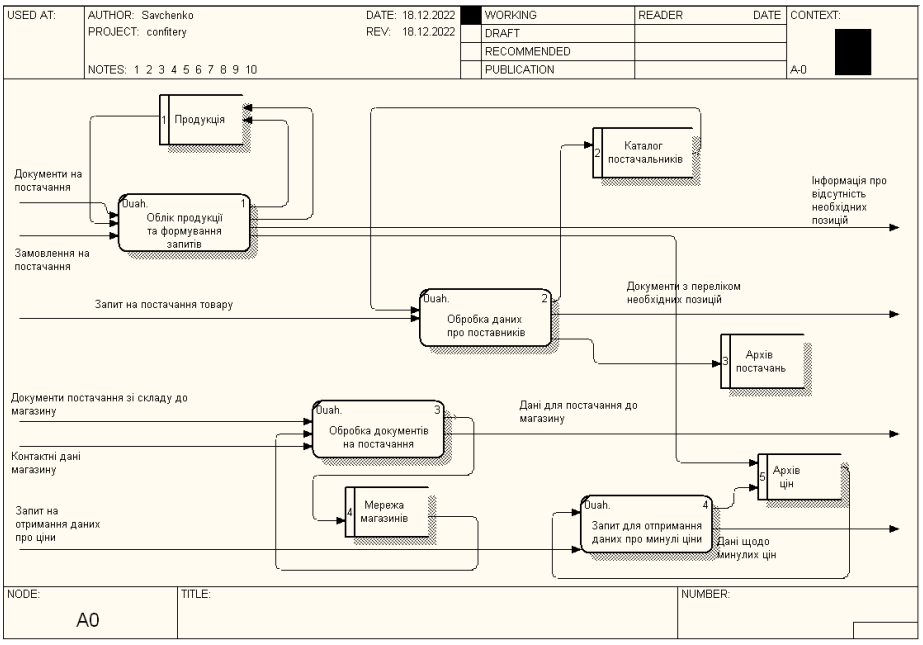

Рисунок 2.5 – Декомпозиція основного процесу «Складський обіг фабрики кондитерських виробів»

2.5 Рекомендація щодо вдосконалення інформаційної системи

Кількість інформації, яку необхідно переробити для вироблення ефективних управлінських рішень, настільки велика, що вона давно перевищила людські можливості. Неможливість далі працювати на складі без ІС, заповнюючи всі журнали вручну, примушує до впровадження ІС.

Без інформаційної системи дуже складно обробляти інформацію у великому обсязі інформації про дані, що надходять. Так само багато документів, написаних вручну, стають непридатними.

Дані про верстати і роботу, виконану на складі, мають бути достовірними, а дані про залишки матеріалів актуальними. Своєчасно отримані та проаналізовані дані дають змогу підприємству нормально функціонувати, не порушуючи бізнес-процеси.

Для ефективності збору даних та їх використання використовують автоматизовані інформаційні системи.

Після впровадження ІС з'явиться можливість організувати електронний документообіг, який вирішить частину описаних вище проблем.

Для ліквідації недоліків необхідно впровадити систему, яка має дозволити:

– скоротити час проведення всіх складських операцій;

– скоротити кількість помилкових складських операцій;

– оптимізувати використання площі складу;

– скоротити витрати на зберігання товару на складі;

– підвищити точність обліку;

– зменшити залежність від "людського фактора".

Сучасні програмні рішення можуть бути налаштовані абодоопрацьовані індивідуально з урахуванням потреб і побажань конкретного клієнта.

2.6 Розробка діаграми класів системи

Діаграма класів - це статична діаграма. Вона являє собою статичне представлення програми. Діаграма класів використовується не тільки для візуалізації, опису та документування різних аспектів системи, але й для побудови виконуваного коду програмного додатку.

Діаграма класів описує атрибути та операції класу, а також обмеження, що накладаються на систему. Діаграми класів широко використовуються при моделюванні об'єктно-орієнтованих систем, оскільки вони є єдиними діаграмами UML, які можуть бути відображені безпосередньо об'єктно-орієнтованими мовами.

Призначенням діаграми класів є моделювання статичного представлення додатку. Діаграми класів є єдиними діаграмами, які можуть бути безпосередньо відображені об'єктно-орієнтованими мовами і тому широко використовуються при проектуванні.

Діаграми UML, такі як діаграма діяльності, діаграма послідовності, можуть дати лише послідовність виконання програми, однак діаграма класів дещо відрізняється від них. Це найпопулярніша діаграма UML у спільноті програмістів.[13]

У діаграми класів є низка переваг:

– аналіз та проектування статичного вигляду додатку;

– опис обов'язків системи;

– база для діаграм компонентів і діаграм розгортання;

– пряме та зворотне проектування.

Для розроблюванної SRM-системи було спроектовано таку діаграму (рис 2.6). В ній відображені основні взаємодіючі класи програми, а також їх поля, властивості та методи.

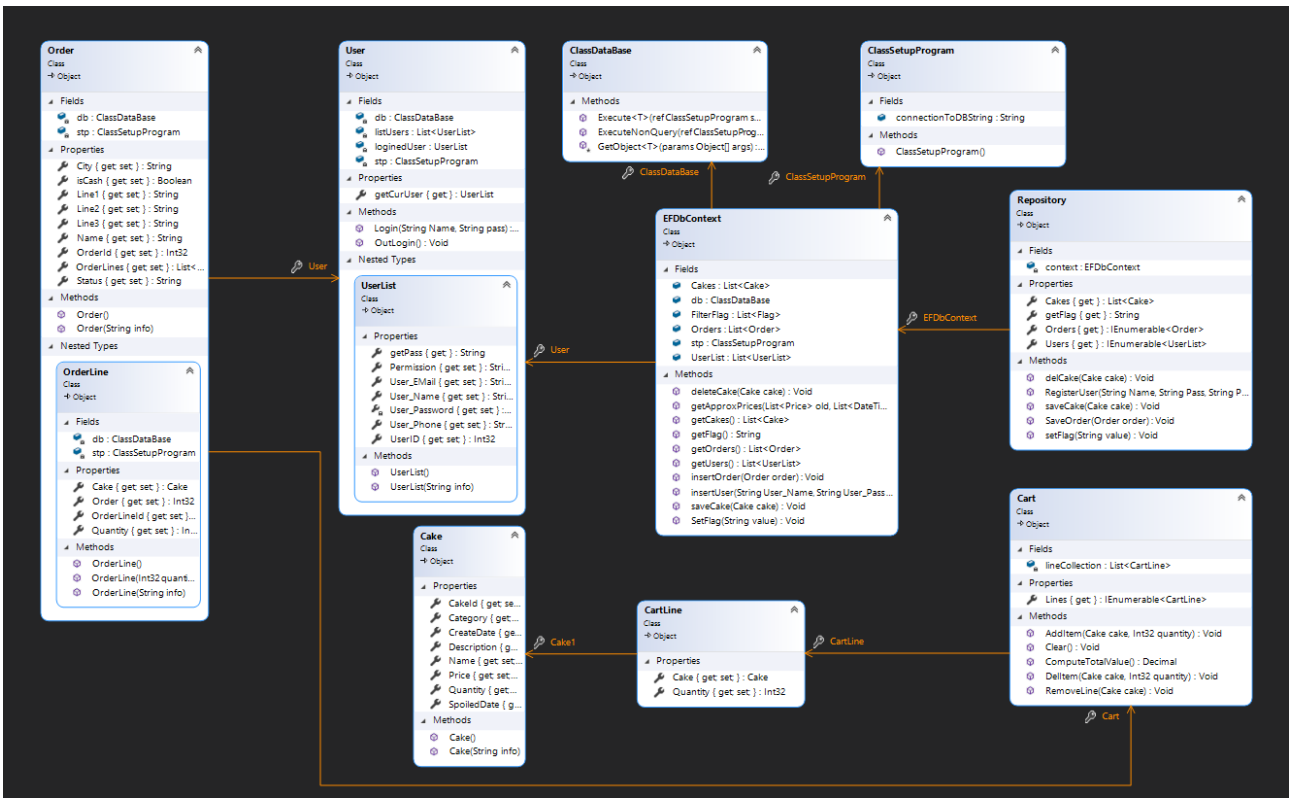

Рисунок 2.6 – Діаграма класів розроблюваної системи

У діаграмі класів було спроектовано 9 основних класів, що в програмі будуть реалізовувати низку патернів:

– патерн проектування «Фасад» є інтерфейсом для роботи із безліччю класів одразу. Фасад має дві прояви: урізаний інтерфейс, що не є 100% реалізацією та повна реалізація, що є багаторівневою організацією класів.

– патерн ActiveRecord – патерн, котрий надає доступ до бази даних та реалізовує основні CRUD-операції;

– патерн Repository – є посередником між двома шарами додатку. Перший є шаром визначення дій та зміних, а другий – шар розподілу їх у програмному забепеченні. Патерн сховища пропонує інший погляд на збереження. Це погляд з точки зору об'єктів, він говорить їхньою мовою і його не цікавить саме Сховище. Об'єкт додається або видаляється з існуючої колекції об'єктів. Крім того, саму колекцію можна ітеративно переглядати та застосовувати бізнес-критерії, щоб у ній були представлені лише цікаві об'єкти. З такою простою

перспективою того, як ми отримуємо, додаємо або видаляємо об'єкти домену, прикладний рівень вирішує випадки використання, не маючи залежності від будь-яких знань про технологію персистентності, але діючи так, як ніби об'єкти знаходяться там, в пам'яті.

При быльш детальному аналізу цієї діаграми можемо виділити певний перелік основних класів:

а) клас EFDbContext.cs – механізм відповідаючий за безпосередній доступ до бази даних. Він надає доступ до зовнішньої бази даних. Також він є реалізацією патерну «Фасад» і тому містить 3 допоміжних класи:

– клас ClassDataBase.cs – тут розташовані основні методи звернення до бази даних.

– клас ClassSetupProgram.cs – зберігає усі дані для з’днання з базою даних;

– клас Repository.cs – містить реалізацію патерну «Репозиторій».

б) клас User.cs – зберігає опис сутностей користувачів. Знаходимо тут як розподіл за ролями так і дані користувачів що потрібні для коректної роботи додатку. Цей клас має властивості:

– getCurUser –отримання поточного користувача;

– getPass – отримання паролю користувача що користується системою. Є скритою властивістю з огляду на безпеку системи в цілому;

– UserID – ідентифікатор користувача (унікальний для кожного запису);

– Permission – роль користувача, що трактує рівень доступу до функцій;

– User_Name – ім’я, що користувач вказав при реєстрації;

– User_Email – пошта користувача;

– User_Phone – телефон користувача;

– User_Password – пароль користувача;

в) клас Cake.cs – реалізація сутності «Продукт». Вона у собі містить такі властивості:

– CakeID – ідентифікатор виробу (унікальний для кожного запису);

– Name – назва виробу (від виробника);

– CreateDate – коли був створений доданий виріб;

– Description – опис виробу, як можна корочше;

– Category – категорія виробу;

– Price – ціна на виріб, що встановлена виробником;

– Quantity – кількістна характеристика наявності на складі;

– SpoiledDate – строк просрочення виробу;

г) клас Cart.cs – реалізація магазинного кошику. Таким чином до цього класу попадають тільки продукти що додані корситувачем безпосередньо для придбання. Включає у собі:

– Lines – цільова властивість, що є реалізацією класу CartLine (кожна окрема строка корзини);

г) клас CartLine.cs – опис кожної окремої строки замовлення. Має такі властивості:

– Cake – виріб/продукція, тобто реалізація класу Cake;

– Quantity – опис кількості замовленного виробу/обраної продукції;

д) клас Order.cs – основний клас, що є описує модель даних «Замовлення», тобто основне джерело даних. Зберігає у собі дані замовлень системи і має набір таких властивостей:

– OrderId – ідентифікатор замовлення (унікальний для кожного запису);

– Name – на кого було зроблено замовлення, ім’я;

– City – куди було зроблено замовлення, яке місто;

– Line1, Line2, Line3 – набір складових адреси;

– Status – у якому статусі перебуває замовлення;

– OrderLines – перелік позицій замовлення;

– isCash – семафор для оплати картою чи готівкою;

2.7 Розробка діаграми послідовності дій системи

Для того щоб бачення розроблюємого модулю було більш зрозумілим – розроблена діаграма послідовності дій під час процесу аналізу попиту (рис 2.7).

На цій діаграмі більш детально показані потоки які будуть працювати паралельно для отпримання прогнозованної ціни. Тут відображені як користувачі системи які будуть робити запит на отримання прогнозованної ціни, так і усі модулі що будуть учасниками цього процесу.

Проаналізувавши цю діаграму можемо зробити висновок, що основне навантаження буде йти на сервер тому що, більшість задач чи установ буде робити саме він. Це як запити до зовнішніх сервісів якими є API та база даних, так і обробка вже отриманих від цих модулів даних.

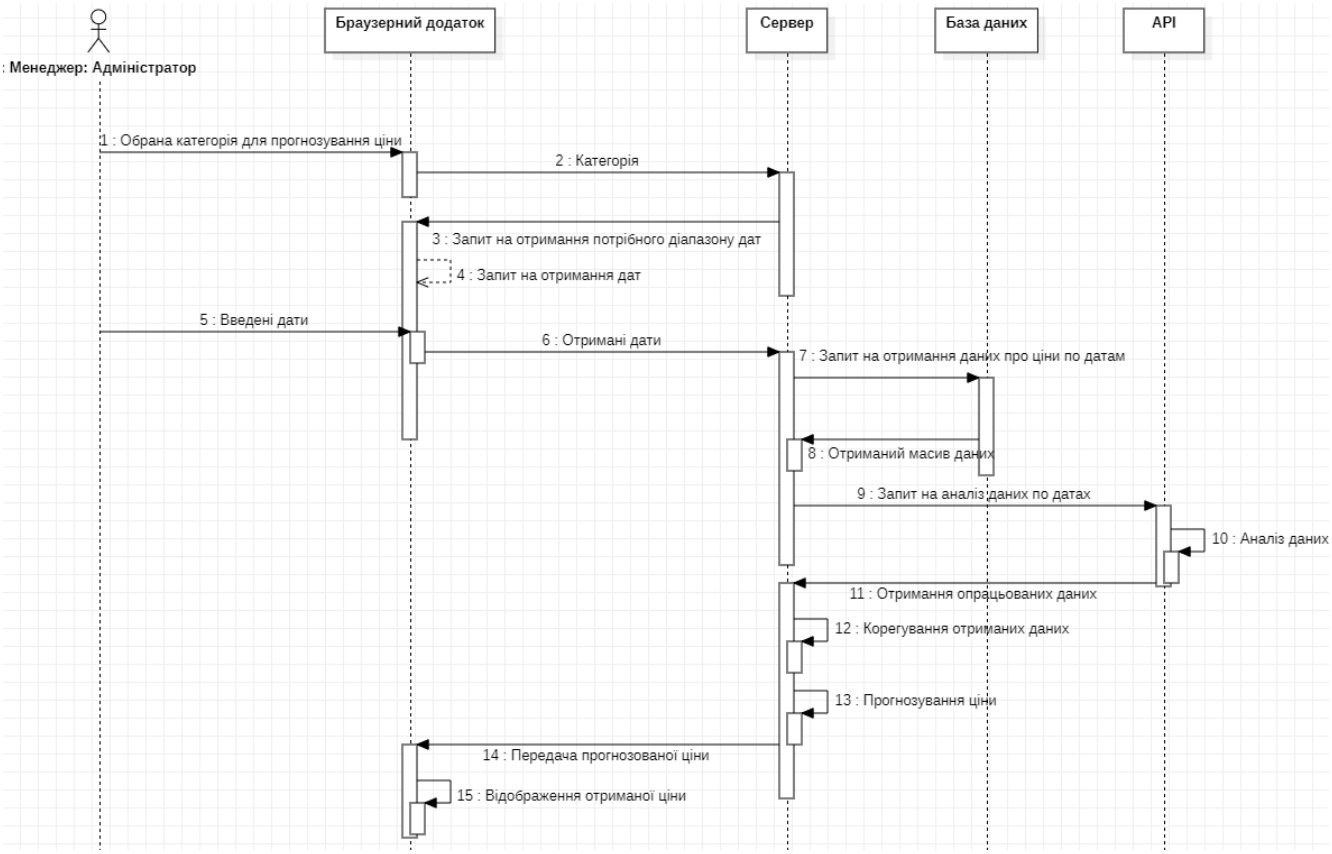

Рисунок 2.7 – Діаграма послідовності для події «Прогнозування ціни»

3 МАТЕМАТИЧНИЙ ОПИС ЗАДАЧІ

3.1 Постановка та опис методів аналізу на основі математичних моделей

При описі задачі як математичної моделі, отримуємо те, що головна задача такоого опису — аналіз попиту продукції у майбутньому. Якщо повернутися до обраної предметної області, наша задача знайти найбільш вигідну криву попиту на кожну позицію з обраних.

Якщо йдеться річ про нововведення, то аналіз попиту за часом проведення має декілька категорій, такі як: попередній, поточний і наступний щодо періоду новітньості продукції.

З них, найважливішим є попередній аналіз, особливо на новітні проекти у науково-технічній сфері продукції. На ньому базується виробнича програма, а також будуються просування, здебільшого стратегічні, на ринку нової продукції.

Попередній аналіз у його класичносу уявленні використовує базу даних, що отримана за допомогою вибіркових обстежень. Проводяться вони у сфері іновації потрібних категорій продукції. Також слід враховувати, що такі дослідження проводяться коли продукція ще є дослідним зразком або знаходиться на етапі запуску у виробництво, а іноді навіть на етапі виведення її на ринок. Якщо у споживача вже є уся поточна інформація про вироблення продукту і її виробництво вже розпочато, то для дослідження попиту вже використовуються інше, а саме - дані журналів обліку попиту або замовлень на нову продукцію[22].

Дані «Журналу обліку» замовлень на продукцію доцільно використовувати і в процесі поточного аналізу, а також зіставити такі кон'юнктурні індикатори, як середні ціни запасів ($Р_з$) і середні ціни продажу ($Р_п$) продукції підприємства. Якщо спостерігається стійкість співвідношення $Р_з > Р_п$ у динаміці, то очевидно, що попитом користується дешевша продукція, якщо $Р_з < Р_п$, то споживач віддає перевагу дорожчій.

На заключній стадії виробничо-господарського циклу – етапі підбиття підсумків – аналіз попиту на нововведення необхідний для визначення відповідності нової продукції потребам ринку, ступеня задоволення в ній попиту і досягнення поставлених цілей.

В аналізі попиту на нову продукцію можуть застосовуватися різні методи. Один із них - аналіз чутливості попиту. Він дає змогу визначити зміну величини попиту залежно від зміни будь-якого з його чинників. Із цією метою розраховують коефіцієнт еластичності попиту (3.1), який показує, на скільки зміниться попит за 1%-вої зміни будь-якого його чинника:

$$E = \frac{\Delta Y}{\Delta X} * \frac{\bar{X}}{\bar{Y}}, \qquad (3.1)$$

де, $\bar{X}$, $\bar{Y}$ – середні значення відносного попиту та впливаючи на них фактори;

ΔX, ΔY – зміни відповідно величини попиту і впливаючого на нього факторного признаку.

Формула (3.1) дає змогу визначити еластичність попиту за способом дугової еластичності. Узяті в другому співмножнику значення величини попиту і його фактора на дату дадуть формулу точкової еластичності, яка видається менш точною. X і Y обчислюються як півсуми значень цих показників до і після зміни факторної ознаки.

Покажемо розрахунок цінової еластичності попиту на нові телевізори за даними табл. 3.1. У коефіцієнта цінової еластичності знак опускається, а його значення трактують за абсолютною величиною.

Якщо коефіцієнт цінової еластичності попиту дорівнює нулю, попит абсолютно нееластичний. Інакше кажучи, за будь-якої зміни ціни попит залишається постійним.

Таблиця 3.1 – Показники цінової еластичності попиту на кондитерські вироби

Ціна на виріб, грош. од.	Кількість попиту, шт.	Абсолютний приріст		Середнє значення		Цінова еластич-ність попиту
		ціни, грош. од.	кількість, шт	ціни, грош. од.	кількість, ш	
P	Q	ΔP	ΔQ	$\bar{P}$	$\bar{Q}$	$E = \frac{\Delta Y}{\Delta X} * \frac{\bar{X}}{\bar{Y}}$
355	60	–	–	–	–	–
360	51	+5	-9	357,5	55,5	11,59
370	35	+10	-16	365,0	43,0	13,58
375	25	+5	-10	372,5	30,0	24,83
459	20	+84	-5	417,0	22,5	1,09

Коефіцієнт цінової еластичності що є менше одиниці, свідчить про нееластичний попит. Якщо коефіцієнт $E_p = 1$ еластичність попиту є одиничною. Таке може траплятися, якщо процент зміни попиту є рівним до проценту зміни ціни. При коефіцієнті еластичності більше одиниці, маємо на увазі що попит відносно еластичний. Якщо при попиті за ціною, коефіцієнт еластичності прагне до нескінченності, то такий попит вважається абсолютно еластичним. Це відбувається досить рідко, і тільки у тому випадку, коли при тій самій ціні, попит необмежено росте, що сприймаємо як наслідок дії нецінових факторів таких мода на щось, реклама товару, валютного курсу, екологічної обстановки, тощо[21].

Також, для охарактеризації цінової еластичності попиту доцільно використовувати графічний метод. Так, на графіку абсолютно нееластичний попит представлений як пряма лінія, яка паралельна осі ординат (рис. 3.1, а), абсолютно еластичний - у вигляді прямої, паралельної осі абсцис (рис. 3.1, б). чим більше кут нахилу кривої попиту, тим еластичніше попит (рис. 3.1, в).

Взагалі прийнято вважати що збільшення кута нахилу сприяє збільшенню еластичності попиту.

Цікавою інформацією для вивчення та аналізу попиту є вплив на нього доходів споживачів. Залежно від призначення продукції (тобто будь-то виробнича, технічна чи побутова) буде відрізнятися інформаційна база аналізу. Перші два випадка можуть бути знайдені у офіційної звітності підприємств, третій у свою чергу - публікується і надається за запитами до Держкомстату.

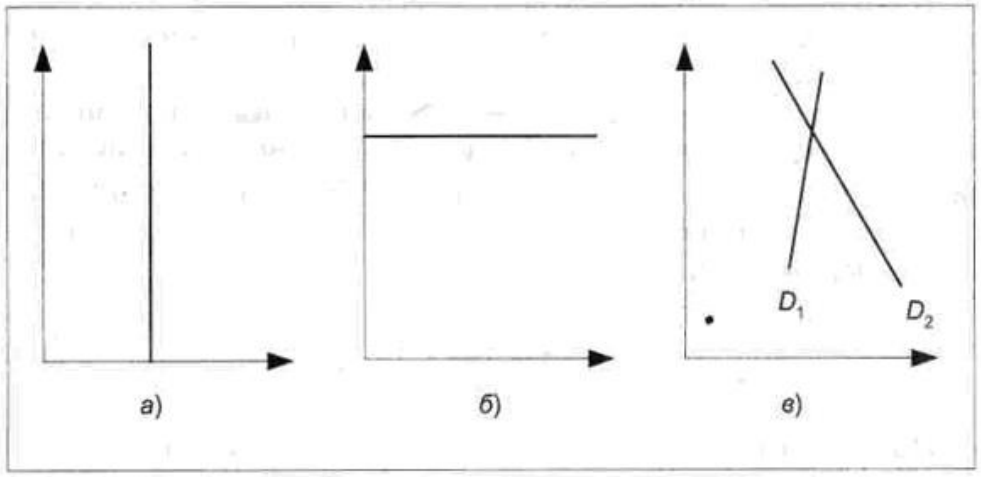

Рисунок 3.1 – Криві, які характеризують еластичність попиту: а) не еластичний попит (абсолютне значення), б) еластичний попит (абсолютне значення), в) D_1 – відносно нееластичний попит; D_2 – відносно еластичний попит

Формула коефіцієнта еластичності (3.2) попиту по доходу має вигляд:

$$E_I = \frac{\Delta Q \bar{I}}{\Delta I \bar{Q}}, \qquad (3.2)$$

де $\bar{I}$ – показник, який характеризує велич прибутку користувача, грош. од.

Для товарів нижчої категорії є дві категорії коефіцієнту еластичності попиту за доходом: позитивний та негативний. Він відіграє значну роль для підприємств і полягає в тому, що допомагає здійснити віднесення продукції до певної групи за рівнем розвитку.

Розвиток підприємтсва як раз визначаєтсься величчю еластичності попиту до доходності самого підприємства. Якщо динаміка даного показника поступово, але стабільно, зростає, можемо заключити що це підприємство стабільно розвивається.

Постійна величина коефіцієнта еластичності попиту по доходу свідчить про стан застою на підприємстві, що виробляє науково-технічну продукцію. Оскільки на попит досить сильно впливає велика кількість факторів-змінників, важливо визначити спряженість нововведень з існуючою на ринку аналогічною за призначенням продукцією. З цією метою застосуємо коефіцієнт перехресної еластичності попиту (3.3):

$$E_{ji} = \frac{\Delta Q_i \overline{P_J}}{\Delta P_j \overline{Q_i}}, \qquad (3.3)$$

Він показує зміну попиту на і-й товар при зміні ціни j-го товару, і значення його може бути як позитивним, так і від'ємним.

Якщо $E_{ji} > 0$, то попит на і-у продукцію прямо залежить від зміни попиту на j-у, тобто і-та та j-а продукції взаємозамінні.

При $E_{ji} < 0$ товари взаємодоповнювані. Тоді в разі підвищення ціни на j-й товар знижується попит на і-й товар.

Якщо $E_{ji} = 0$, то і-й і j-й види продукції не залежать один від одного.

Коефіцієнт перехресної еластичності має особливо велике значення для аналізу попиту на новітню наукову та технічну продукцію, і з цього приводу, він повинен постійно реаналізуватися та оновлюватися. Для цього можна використовувати співвідношення темпів зміни попиту, чинники, які впливають на нього, а також факторів, що представлені у вигляді динамічного ряду.

При побудові таких рядів з’являється змога побудувати графік зміни кривої попиту що підпорядкована обраному фактору, а також, можна обчислити показники динамічних рядів: абсолютні значення одного відсотка зміни попиту за певний період, середні темпи приросту, а також провести аналіз впливу факторів на зміну попиту кореляційним способом і зробити прогноз на його

розвиток у найближчій перспективі. Якщо в кожному етапі життєвого циклу продукту динамічні темпи зростання збільшуються по споживачам або збутовим каналам, то такий попит є інтенсивним. Якщо ж ці темпи зростання наближуються до одиниці – попит стабілізувався (в відношенні розглянутого аспекту). Якщо темпи зростання після розрахунків опинилися менше за одиниця, – попит йде на скорочення. Такий аналіз попиту має неоцінне значення, оскільки від його висновків залежить як точність розробки майбутньої розробленої системи так і стратегія з обсягами реалізації продукції а, отже, і фінансові результати діяльності.

3.2 Дослідження методу аналізу цінової еластичності та багатокритеріального аналізу

Цінова еластичність попиту - це вплив попиту на товар, підвищення або зниження його ціни. Цінова еластичність може демонструвати, що зі збільшенням ціни споживачі будуть купувати менше товару так і навпаки. Основний чинник, який дає змогу з'ясувати цінову еластичність, полягає втому, наскільки змінюється попит з плином часу. [14].

Попит може зрости або на товар-замінник, або на зовсім іншу категорію товарів, або споживач може взагалі не купувати товар і заощадити ці гроші. Встановлення ціни є важливим аспектом бізнесу та продукту. Вона повинна бути відповідною, щоб покрити витрати та зберегти бажану маржу. Компанія повинна знати, хто є її споживачем і наскільки чутливою до ціни є його демографічна група. Компанія також повинна враховувати економічний клімат і бути обізнаною з економічними факторами, які можуть вплинути на витрати.

Дане дослідження забезпечує краще розуміння кондитерського бізнесу, використовуючи інформацію з бази даних IRI/Nielson. На основі інформації про продажі та конкурентів для всієї категорії кондитерських виробів, це дослідження аналізує фінансову звітність компанії, відстежуючи зростання цін

протягом двох фінансових кварталів поспіль, щоб порівняти результати діяльності компанії з іншими компаніями в тій же самій галузі. Дослідження має на меті зробити внесок у літературу щодо цінової еластичності відносно недорогого та імпульсивного товару.

Еластичність попиту за ціною (Price elasticity of demand - PED) вимірює реакцію попиту товару на зміну його ціни. Вона розраховується як процентна зміна попиту, поділена на процентну зміну ціни. Якщо попит еластичний, невелика зміна ціни призводить до великої зміни обсягу продажів. Таким чином еластичність попиту за ціною характеризує здатність покупців відмовитися від товару (або замінити на інший товар) у відповідь на підвищення ціни. Формула еластичності попиту за ціною:

$$PED = \frac{\frac{\text{ПродажіНові} - \text{ПродажіПоточні}}{\text{ПродажіПоточні}}}{\frac{\text{ЦиниНові} - \text{ЦиниПоточні}}{\text{ЦиниПоточні}}} = \frac{\text{ЗмінаПродажів\%}}{\text{ЗмінаЦіни\%}}. \quad (3.4)$$

Наведена вище формула зазвичай дає від'ємне значення через зворотний характер відношення між ціною і кількістю попиту.

У міру того, як різниця між двома цінами або кількостями збільшується, точність PED, що визначається наведеною вище формулою, зменшується з двох причин. По-перше, еластичність товару не обов'язково постійна, вона змінюється в різних точках кривої попиту через свій відсотковий характер. По-друге, відсоткові зміни несиметричні; натомість відсоткова зміна між будь-якими двома значеннями залежить від того, яке з них обрано як початкове, а яке як кінцеве.

Цінова еластичність застосовується в компаніях в основному для ціноутворення. У разі вмілого використання цінової еластичності можна домогтися підвищення виручки і прибутку компанії при збереженні лояльності покупців [14].

Розглянемо приклад: у вас є інтернет-магазин, у якому ви продаєте кондитерську продукцію. Для прикладу візьмемо 2 товари: 3-х ярусний торт за ціною 100 $ і набір кексів, який теж за ціною 100 $.

Якщо ціна на торт збільшується зі 100 $ до 110 $, а обсяг попиту при цьому зменшується з 20 до 19 шт., то еластичність (рис 3.2):

$$PED = \frac{\frac{19-20}{20}}{\frac{110-100}{100}} = -0.5.$$

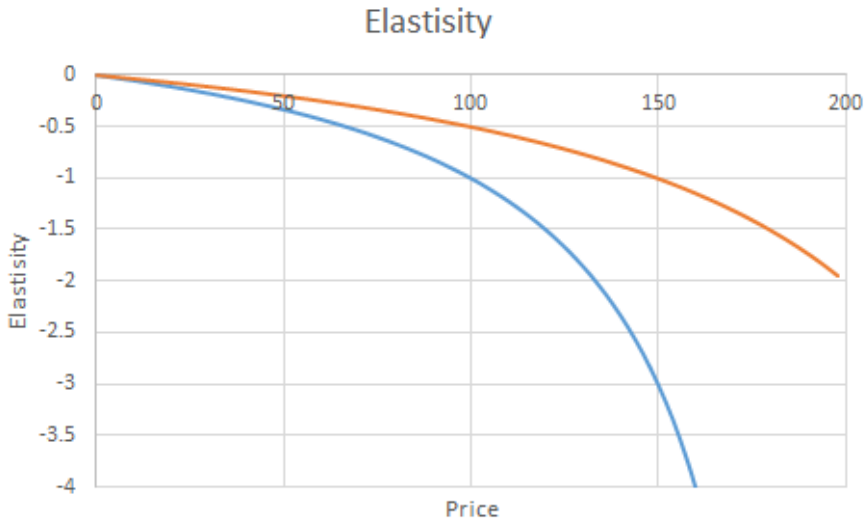

Рисунок 3.2 – Залежність еластичності товарів від зміни ціни

Де, помаранчева лінія це торти, а синя – кекси. Під час обчислення еластичності попиту важливо визначитися з вихідною точкою відліку ціни та продажів, тому що відносно цих даних розраховуватимуться відсотки змін ціни та продажів. Так, наприклад, для кексів, якщо обсяг попиту збільшується з 20 до 25 одиниць, відсоткова зміна становить (25-20)÷20=25%. Але якщо обсяг попиту знизиться з 25 одиниць до 20 одиниць, відсоткова зміна становитиме (20-25)÷25= -20%.

Перехресна еластичність попиту за ціною - відсоткова зміна попиту у відповідь на зміну в ціні конкурентного товару або послуги. Перехресна еластичність попиту за ціною майже завжди позитивна, тому що в разі підвищення ціни на товар покупці переключаються на конкурентний товар, підвищуючи його продажі [15]. На практиці є кілька складнощів при застосуванні еластичності в ціноутворенні:

а) крива залежності продажів від ціни має складну нелінійну форму через присутність конкурентних товарів з конкретною ціною, через насичення попиту, через, наприклад, термін придатності для продуктів харчування тощо;

б) наразі ми не знаємо форму цієї кривої повністю, тому що в історичних продажах у нас є тільки продажі при вузькому діапазоні зміни ціни. А в деяких випадках може бути, що товар завжди продавався за однієї й тієї самої ціни і тоді ми взагалі не знаємо його еластичність навіть у поточній точці;

в) цю криву важко будувати, тому що всі вищевказані вище графіки припускають, що всі інші фактори залишаються незмінними;

г) форма кривої еластичності з плином часу змінюється, і для деяких товарів це може відбуватися досить швидко.

Реальні залежність продажів від ціни та цінова еластичність можуть вигляд, який представлено на рисунку 3.3.

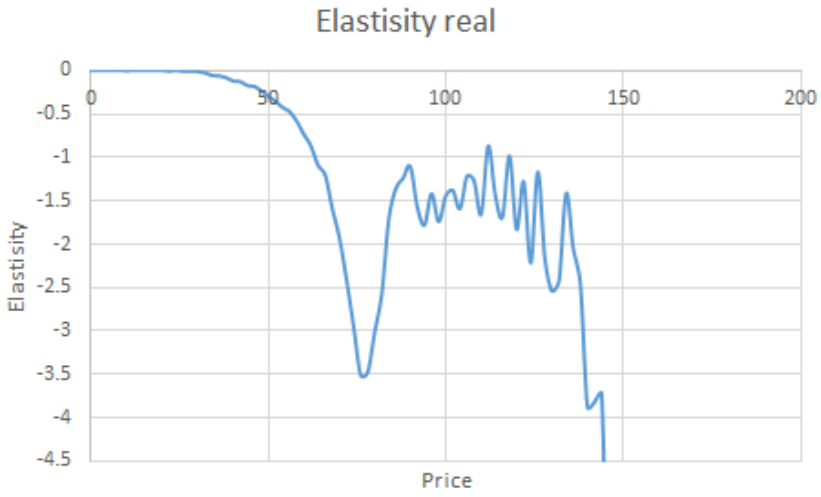

Рисунок 3.3 – Приблизний графік реальної еластичності товарів

З усіх вищевказаних причин на практиці зараз широко застосовується машинне навчання для застосування еластичності ціни в ціноутворенні. З його допомогою, наприклад, можна відновити форму кривої еластичності товару за даними схожих товарів.

Для розрахунку еластичності попиту за ціною в Excel можна застосувати рівняння =linest(known_y's; known_x's, ...)/SalesCur*PriceCur. Де known_y's - це

ваші історичні продажі, a known_x's - це ваші ціни для цих продажів. У підсумку застосування ви отримаєте коефіцієнт, який показує, на скільки одиниць зміняться ваші продажі при підвищенні цін на 1 одиницю. Однак будьте обережні, перед застосуванням цієї формули переконайтеся що:

а) у вас є достатньо статистики продажів для кожного рівня ціни (якщо період виміру продажів 1 день, то не менше 7 днів поспіль);

б) продажі були значними (якщо період виміру продажів 1 день, то не менше 10 штук на день);

в) зміни цін були відчутними (не менше ніж 5%, і не менше ніж 10% для дешевих товарів).

Наприклад, якщо ви продавали якийсь товар за 20 євро, то бажано, щоб у вас була інформація:

а) 7 днів продажів не менше 10 штук на день за ціни 19 євро;

б) 7 днів продажів не менше 10 штук на день за ціни 20 євро;

в) 7 днів продажів не менше 10 штук на день при ціні 21 євро.

В іншому випадку занадто велику роль може зіграти ефект випадковості. Також постарайтеся уникнути експериментів зі з'ясування еластичності попиту у святкові дні та дні великих коливань продажів сезонних товарів [23].

Також застосовується тактика постійних невеликих змін цін, для подальшого уточнення кривої еластичності в близькому діапазоні.

Еластичність попиту застосовується для задач ціноутворення, прогнозування попиту, максимізації виручки. Найчастіше це вирішується за допомогою застосування моделей машинного навчання, які виявляють залежність продажів від різних чинників: ціни, дня тижня, погоди тощо. Найпоширенішими моделями для таких завдань є:

а) моделі на основі дерев прийняття рішення;

б) моделі на основі лінійної регресії;

в) моделі на основі нейронних мереж.

Вибір моделі залежить від багатьох факторів, також кожна модель має свої плюси і мінуси [16], наприклад:

а) якщо вихідних даних мало, то варто застосовувати простіші моделі, наприклад лінійну модель;

б) якщо необхідно екстраполювати залежність продажів від ціни (тобто припустити продажі за ціни більшої, ніж будь-коли була в історії), то моделі на основі дерев ухвалення рішення не підходять для таких завдань, тому що вони не призначені для екстраполяції залежностей;

в) якщо вихідні дані сильно зашумлені, залежність нелінійна і вихідних даних небагато, то навпаки варто спробувати Моделі на основі дерев ухвалення рішення;

У разі лінійної моделі ви отримаєте залежність продажів від факторів, наприклад, такого виду:

$$Sales = W0 + W1 * \text{Ціна} + W2 * \text{Сезон} + W3 * < \text{показник} > + \cdots . \quad (3.5)$$

Отримавши таку модель, ви можете зафіксувати всі інші фактори, крім ціни, і намалювати графік залежності продажів від ціни в діапазоні цін, який вас цікавить. Тобто методи машинного навчання дають змогу очистити залежність "продажі-ціна" від інших шумових чинників і в такий спосіб отримати точніший коефіцієнт залежності. Далі, отримавши цей графік і коефіцієнт, можна обчислити еластичність попиту за ціною для вирішення завдання ціноутворення. Далі можна також максимізувати виручку, отримавши для неї реальні графіки залежно від ціни. Якщо припустити, що залежність лінійна як у наведеній вище формулі, то виторг буде максимальним за ціни:

$$\text{Ціна} = \frac{W0 + W1 * \text{Ціна} + W2 * \text{Сезон} + W3 * < \text{показник} > + \cdots}{-2 * W1}. \quad (3.6)$$

4 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ ПРИ РОЗРОБЦІ СИСТЕМИ

4.1 Обґрунтування вибору мови програмування

Вже минули часи кол компанії, будь-яких розмірів та штатів не могли додавати бажані функції до їх додатків та комплексних рішень, особливо коли вони розроблювали якийсь інтернет-додаток. Раніше для того щоб такі функції отримати для веб-додатків, доводилося значні суми, котрі навіть великі компаніїї не могли собі дозволити, вже не кажучт про малі підприємтсва.

Але, з появою компанії Microsoft, все змінилося. Для зниження витрат і одночасно створення власних веб-додатокі з необхідними функціями, Microsoft випускає, мабуть, найпродуктивнішу і найвигіднішу з точки зору бізнесу систему, відому як ASP.NET. Цьому фреймворку вже більше десяти років і має місце той факт, що саме за допомогою цього фреймворку Microsoft у своєму класичному уявленні як постачальника розробницьких додатків стала найпопулярніщою.

ASP.NET - це платформа для веб-розробки, що надається компанією Microsoft. Використовується для створення веб-додатків. Вперше ASP.NET була випущена у 2002 році.

Першою розгорнутою версією ASP.NET була 1.0. Останньою версією ASP.NET є версія 4.6. ASP.NET призначений для роботи з протоколом HTTP. Це стандартний протокол, який використовується у всіх веб-додатках.

Додатки ASP.NET також можуть бути написані на різних мовах .Net. До них відносяться C#, VB.Net і J#. Повною ж формою є ASP - це Active Server Pages, а .NET - Network Enabled Technologies.

ASP.NET - це фреймворк, який використовується для розробки веб-додатків. Базова архітектура фреймворку ASP.NET показана нижче.

Архітектура фреймворку .NET базується на наступних ключових компонентах:

– Мова – для фреймворку .NET існує безліч мов. Це VB.NET та C #. Вони можуть бути використані для розробки веб-додатків.

– Бібліотека – фреймворк .NET Framework включає в себе набір стандартних бібліотек класів. Найбільш поширеною бібліотекою, що використовується для веб-додатків в .NET, є веб-бібліотека. Веб-бібліотека має всі необхідні компоненти, що використовуються для розробки веб-додатків .net.

– Common Language Runtime – Інфраструктура спільної мови або CLI є платформою. Програми .NET виконуються на цій платформі. CLR використовується для виконання ключових дій. Діяльність включає обробку винятків та збір сміття.

Нижче наведені деякі ключові характеристики фреймворку ASP.NET:

– Code Behind Mode – це концепція поділу дизайну та коду. Здійснюючи це розділення, стає легше підтримувати додаток ASP.NET. Загальний тип файлу ASP.NET – .aspx. Припустимо, у нас є веб-сторінка з назвою MyPage.aspx. Існує ще один файл MyPage.aspx.cs, який буде позначати кодову частину сторінки. Таким чином, Visual Studio створює окремі файли для кожної веб-сторінки, один для дизайнерської частини, а інший для коду.

– Управління станом - ASP.NET має можливість контролювати управління станом. HTTP відомий як протокол без стану. Візьмемо приклад додатку кошика для покупок. Тепер, коли користувач вирішує, що він хоче купити на сайті, він натискає кнопку "Відправити", додатку необхідно запам'ятати елементи, які користувач вибрав для покупки. Це називається запам'ятовуванням стану додатка в поточний момент часу. HTTP - це протокол без стану. Коли користувач переходить на сторінку покупки, HTTP не зберігає інформацію про товари в кошику. Необхідно виконати додаткове кодування, щоб гарантувати, що елементи кошика можуть бути перенесені на сторінку покупки. Така реалізація часом може стати складною. Але ASP.NET може виконувати управління станом

від вашого імені. Таким чином, ASP.NET може запам'ятати елементи кошика і передати його на сторінку покупки.

– Кешування - ASP.NET може реалізувати концепцію кешування. Це покращує продуктивність програми. Завдяки кешуванню ті сторінки, які часто запитуються користувачем, можуть зберігатися в тимчасовому місці. Ці сторінки можуть бути отримані швидше і користувачеві можуть бути надіслані кращі відповіді. Таким чином, кешування може значно підвищити продуктивність програми.

ASP.NET - це мова розробки, яка використовується для побудови веб-додатків. ASP.NET призначена для роботи зі стандартним протоколом HTTP.

Python має багато чудових можливостей, які роблять її простою у використанні, тому не дивно, що навіть з появою інших мов програмування високого рівня, Python все ще залишається однією з найпопулярніших мов програмування у світі [19].

Витримавши випробування часом і будучи такою високо оціненою мовою програмування, можна було б очікувати, що Python буде досконалою, але, як і будь-яка інша мова програмування, Python також має деякі недоліки.

Люди майже не говорять про зворотній бік Python, але сьогодні ми розкриємо хороші і не дуже аспекти розробки програмного забезпечення на Python.

Переваг мови програмування Python дуже багато, але давайте розглянемо найпопулярніші плюси, які активно підкреслюють Python-розробники.

– Python легко вивчається і читається.

У міру того, як все більша кількість людей починає займатися програмуванням, ми можемо спостерігати, що вони стикаються з труднощами, тому що починають з мов, які складні для вивчення.

Python виділяється в цьому плані тим, що більшість новачків можуть легко зрозуміти, як працює синтаксис. Це може полегшити їм шлях до того, щоб стати провідним розробником Python.

Навіть будучи досвідченим розробником програмного забезпечення, Python полегшує вам життя, тому що його легко читати. Ви можете без особливих труднощів розібратися в коді, написаному іншим розробником.

Легкість вивчення та читання Python робить його таким приємним, тому що він вимагає від вас написання меншої кількості рядків коду, даючи при цьому той самий результат, що й інші мови програмування, які вимагають більшої кількості рядків коду.

– Python підвищує продуктивність.

Всі мови програмування призначені для підвищення продуктивності, але Python виводить її на більш високий рівень.

Оскільки Python настільки легкий для читання, ви можете зосередитися на фактичному створенні рішень замість того, щоб розшифровувати природу мови програмування. Одна тільки ця функція може допомогти вам заощадити час і зменшити кількість стресу, який зазвичай приходить з роботою.

Завдяки динамічній типізації Python призначає тип даних під час виконання програми. Таким чином, вам не потрібно турбуватися про оголошення змінних або вказівку типів даних під час написання коду.

З таким тягарем, знятим з ваших плечей, ви просто повинні визнати, що Python дійсно підвищує продуктивність, оскільки змінна не турбує нас до тих пір, поки нам не потрібно запускати код.

У порівнянні з іншими мовами програмування високого рівня, такими як Java, ми можемо обрати Python через його потужні інтеграційні можливості, які роблять його кращим вибором для створення корпоративних програмних додатків.

4.2 Обґрунтування вибору платформи промислової СУБД

База даних - це окремий додаток, який зберігає набір даних. Кожна база даних має один або декілька окремих API для створення, доступу, управління, пошуку та реплікації даних, які вона зберігає.

Інші види сховищ даних також можуть використовуватися, наприклад, файли у файловій системі або великі хеш-таблиці в пам'яті, але отримання та запис даних не буде таким швидким і простим у системах такого типу.

Сьогодні більшість розробників використовує реляційні системи управління базами даних (СУБД) для зберігання та управління величезними обсягами даних. Це називається реляційною базою даних, тому що всі дані зберігаються в різних таблицях, а зв'язки між ними встановлюються за допомогою первинних ключів або інших ключів, відомих як зовнішні ключі.

Для нашого проекту найбільш ефективною буде СУБД MySQL. MySQL – найпопулярніша система управління базами даних з відкритим вихідним кодом (Open Source SQL), розробляється, розповсюджується та підтримується корпорацією Oracle.

Перевагами системи управління базами даних MySQL є те, що:

– Бази даних MySQL є реляційними. Реляційна база даних зберігає дані в окремих таблицях, а не в одному великому сховищі. Структури бази даних організовані у фізичні файли, оптимізовані для швидкості. Логічна модель з такими об'єктами, як бази даних, таблиці, подання, рядки та стовпці, пропонує гнучке середовище програмування. Ви встановлюєте правила, що регулюють зв'язки між різними полями даних, такими як "один до одного", "один до багатьох", унікальні, обов'язкові або необов'язкові, а також "покажчики" між різними таблицями. База даних забезпечує дотримання цих правил, так що з добре спроектованою базою даних ваша програма ніколи не побачить непослідовних, дублюючих, "сирітських", застарілих або відсутніх даних.

– Програмне забезпечення MySQL має відкритий вихідний код. Відкритий вихідний код означає, що будь-хто може використовувати та модифікувати програмне забезпечення. Будь-хто може завантажити програмне забезпечення MySQL з Інтернету і використовувати його, не сплачуючи за це жодної копійки. За бажанням, ви можете вивчити вихідний код і змінити його відповідно до ваших потреб. Програмне забезпечення MySQL використовує ліцензію GPL (GNU General Public License), яка визначає, що можна і що не можна робити з програмним забезпеченням у різних ситуаціях.

– Сервер баз даних MySQL є дуже швидким, надійним, масштабованим і простим у використанні. Якщо це те, що ви шукаєте, вам варто спробувати. Сервер MySQL може комфортно працювати на настільному комп'ютері або ноутбуці, поряд з іншими вашими додатками, веб-серверами і т.д., вимагаючи мало уваги або взагалі не вимагаючи її. Якщо ви присвятите MySQL цілу машину, ви можете налаштувати параметри, щоб скористатися всіма перевагами пам'яті, потужності процесора та пропускної здатності вводу/виводу. MySQL також може масштабуватися до кластерів машин, об'єднаних в мережу.

4.3 Опис архітектури розробленої системи

Фреймворк Model-View-Controller (MVC) - це архітектурний патерн, який розділяє додаток на три основні логічні компоненти Model, View і Controller. Звідси і походить абревіатура MVC. Кожен компонент архітектури створений для обробки певного аспекту розробки додатку. MVC відокремлює бізнес-логіку та рівень представлення один від одного. Традиційно він використовувався для настільних графічних інтерфейсів користувача (GUI). Сьогодні архітектура MVC у веб-технологіях стала популярною для розробки веб-додатків, а також мобільних додатків. [18] (рис. 4.1).

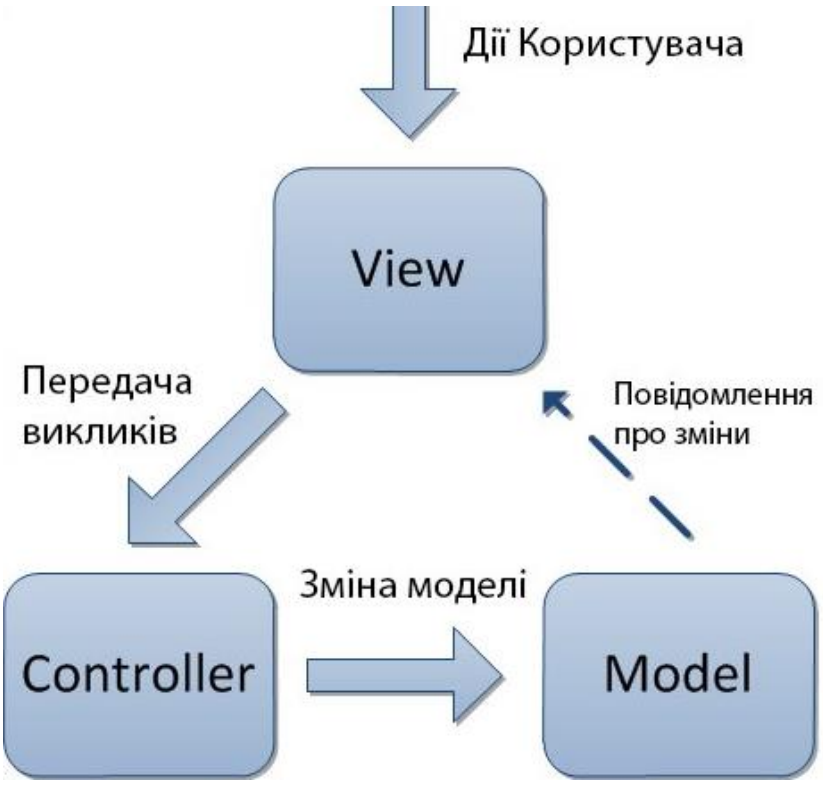

Рисунок 4.1 – Схема роботи архітектури «Модель-Вид-Контролер»

Три важливі компоненти MVC:

а) Модель: Включає в себе всі дані і пов'язану з ними логіку.

б) Подання: Представляє дані користувачеві або обробляє взаємодію з користувачем.

в) Контролер: Інтерфейс між компонентами Model та View.

Розглянемо кожен з цих компонентів більш детально:

– Представлення (View) – це та частина додатку, яка представляє представлення даних. Представлення створюються на основі даних, зібраних з даних моделі. Представлення запитує у моделі інформацію, щоб вона надала користувачеві вихідне представлення.

Представлення також представляє дані з графіків, діаграм і таблиць. Наприклад, будь-яке представлення користувача буде включати всі компоненти інтерфейсу користувача, такі як текстові поля, випадаючі списки і т.д.

– Контролер – це та частина програми, яка обробляє взаємодію з користувачем. Контролер інтерпретує введення миші та клавіатури від

користувача, інформуючи модель та подання, щоб змінити їх відповідно до ситуації. Контролер надсилає команди моделі для оновлення її стану (наприклад, збереження певного документа).

Контролер також надсилає команди пов'язаному з ним представленню для зміни представлення представлення (наприклад, прокручування певного документа).

– Модель – це компонент моделі зберігає дані та пов'язану з ними логіку. Він представляє дані, які передаються між компонентами контролера або будь-якою іншою пов'язаною з ними бізнес-логікою. Наприклад, об'єкт Controller отримує інформацію про клієнта з бази даних. Він маніпулює даними та надсилає їх назад до бази даних або використовує їх для відображення тих самих даних.

Він відповідає на запит від представлень, а також реагує на інструкції від контролера щодо оновлення. Це також найнижчий рівень шаблону, який відповідає за підтримку даних..

API є важливими інструментами для бізнесу - вони спрощують організаційні операції та забезпечують стратегічну цінність, таку як додаткове представлення вашого бренду та збільшення доходів. API - це новий тип екосистеми, що дозволяє компаніям виходити на нові ринки, де вони можуть використовувати їх не лише для реалізації технічних концепцій [17].

API розшифровується як "інтерфейс прикладного програмування", а словник описує його як "набір функцій і процедур, що дозволяють створювати додатки, які отримують доступ до функцій або даних операційної системи, програми або інших служб". У більш простих термінах, API - це програмний посередник, який дозволяє двом різним додаткам спілкуватися один з одним (рис. 4.2).

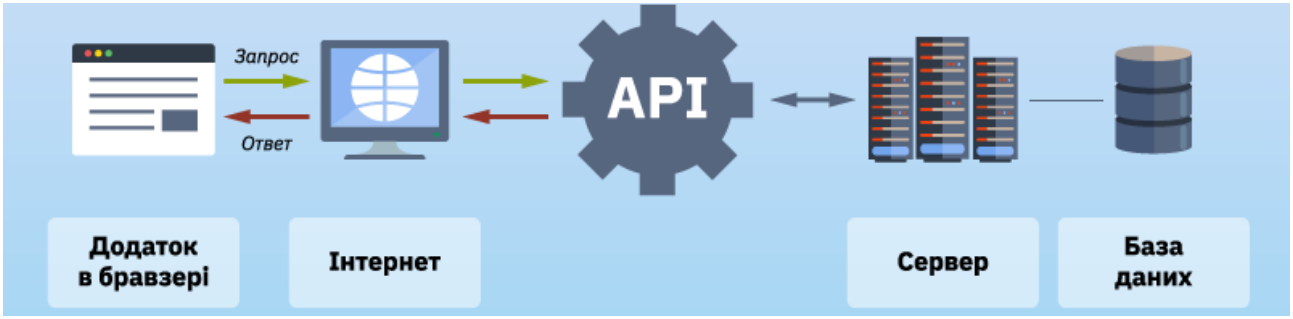

Рисунок 4.2 – Класична структура API

Комерційні веб-сайти часто надають розробникам специфікації або протоколи про те, як запитувати послуги або обмінюватися даними з компанією. Код, яким вони діляться, називається API, а інструменти, які вони виробляють, називаються додатками. Дуже часто великі організації створюють API для своїх клієнтів або для внутрішнього використання, і метою, як правило, є зовнішній обмін інформацією.

Компанії будь-якого розміру можуть використовувати API для аналітики веб-сайтів, інструментів управління проектами та командами, систем онлайн-платежів та багатьох інших операційних рішень.

– Аналітика. Google Analytics є частиною платформи Google Cloud API Platform і пропонує безліч варіантів API, які компанії можуть використовувати для аналітики. Найбільш поширеними API є Core Reporting API та Management API.

– Управління проектами та командами. Такі інструменти, як Jira, Basecamp та Microsoft Teams, забезпечують комунікацію між колегами та обмін даними - і все це відбувається за допомогою API.

– Системи онлайн-платежів. Там, де відбуваються цифрові грошові перекази, існує API, що підтримує цей процес. Багато організацій вирішують використовувати API від таких фірм, як PayPal, щоб забезпечити надійні, безпечні грошові перекази своїм клієнтам.

Сформулюємо такі недоліки API:

– існує багато зручностей і переваг API, але бізнес-лідери також повинні знати про їх недоліки. Як єдина точка входу, API є шлюзом і може стати першочерговою ціллю для хакерів. Як тільки API скомпрометовано, всі інші додатки та системи стають вразливими;

– дев'ять з десяти найбільших вразливостей, перерахованих в OWASP Top 10, тепер згадують API – і оскільки до API можна отримати доступ через Інтернет, вони будуть мати всі ті ж недоліки, що і будь-який інший інтернет-ресурс. API вразливі до атак типу "людина посередині", CSRF-атак, XSS-атак, SQL-ін'єкцій та DDoS-атак.

4.4 Фізичне моделювання даних системи та створення бази даних на платформі СУБД

Відразу від початку моделювання ми зтикнулися з складною задачею, задачею вибору СУБД, але після вибору проблем не поменшало – тепер перел нами вибір структури СУБД. У нас в наявності є дві альтернативи: MyISAM та InnoDB. При порівнянні цих двох схем зберігання даних, ми стикаємось з порівняльною характеристикою двох схем: InnoDB і MyISAM [20]. Основні відмінності між InnoDB та MyISAM ("щодо проектування таблиці або бази даних", про яку ви запитували) - це підтримка "посилальної цілісності" та "транзакцій". Ми обираємо InnoDB, якщо нам потрібно, щоб база даних забезпечувала обмеження зовнішнього ключа або підтримувала транзакції (тобто зміни, внесені двома або більше DML-операціями, які обробляються як одна одиниця роботи, при цьому всі зміни або застосовуються, або всі зміни відміняються). Ці функції не підтримуються механізмом MyISAM.

Це дві найбільші відмінності. Інша велика відмінність - це паралельність. У MyISAM оператор DML отримує ексклюзивне блокування таблиці, і поки це блокування утримується, жоден інший сеанс не може виконати операцію SELECT або DML над таблицею. Ці два конкретні механізми, про які ви

запитували (InnoDB і MyISAM), мають різні цілі проектування. У MySQL також є інші механізми зберігання даних, зі своїми власними цілями проектування.

При виборі між InnoDB і MyISAM першим кроком є визначення того, чи потрібні нам функції, що надаються InnoDB. Якщо ні, то можна розглядати MyISAM. [20].

Після аналізу усіх переваг обох схем звісно ж оберемо InnoDB тому як альтернатива програє в потенціалі до досить частих змін, та в об’ємах цих даних тому що, при зберіганні архіву з минулими цінами він буде досить великий.

Складена модель (рис. 4.2) враховує, що в системі є і функціонує декілька ролей (такі як користувач, зареєстрований користувач, адміністратор, постачальник). А також відомо, що сама база буде складатися з окремих двох схем, кожна з котрих буде мати відношення до різних бізнес-процесів.

Також створене сховище даних яке зберігає у собі вибірку даних для послідуючого прогнозування ціни і вкиористання цих даних для аналізу. Обрана схема дозволяє зберігати велику кількість таких даних у потрібному для обробки форматі. Обираючи MySQL ми одразу вирішили декілька питань з приводу зберігання таких об’ємів даних. Перше – дуже ефективний та надійний спосіб експорту таких даних, друге – вигідний формат експорту .csv, що дозволяє одразу праццювати з даними як з дата-сетом, на відміну від СУБД Oracle який дав би нам лише .xml формат, що приводить до розширення методів нашого додатку тільки за для того щоб провести конвертацію даних з одного формату на більш підходящий для нас.

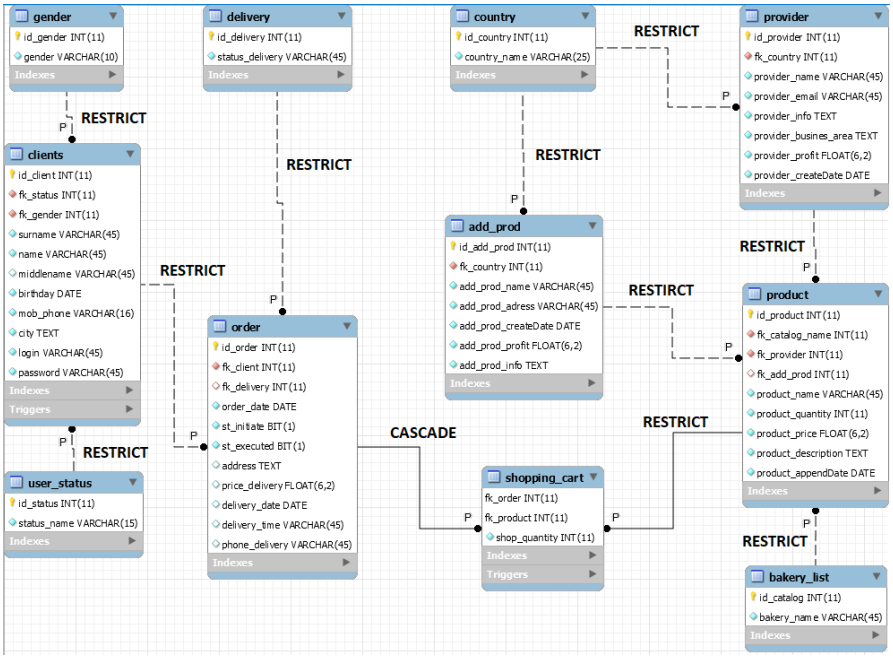

Рисунок 4.2 – Схема бази даних SRM-системи

База даних була зроблена, за допомогою засобу моделювання баз даних вбудований у СУБД MySQL WorkBench, що дозволяє після моделювання відразу перевести цю схему в реальну базу даних [20]. При створенні фізичної моделі бази даних, було враховано, що СУБД MySQL має спільну мову з СУБД MySQL. Тому за відсутності в програмі ERWin можливості створення моделі для СУБД MySQL, модель була створена схема для СУБД MySQL, яку можна використати для створення бази в СУБД MySQL.

4.5 Розробка алгоритму роботи системи

4.5.1 Розробка алгоритму роботи постачальників

Алгоритм - це процедура, яка використовується для розв'язання задачі або виконання обчислень. Алгоритми діють як точний перелік інструкцій, які крок за кроком виконують задані дії в апаратних або програмних процедурах. Алгоритми широко використовуються у всіх сферах IT, а також

використовуються як специфікації для виконання обробки даних і відіграють важливу роль в автоматизованих системах.

Побудований далі алгоритм відображає веб-службу «Робота з постачальниками». Цей модуль буде містити багато блоків що позначають фуункції, які вимагають послідовної виконання. Для представлення такої служби з точки зору постачальника був розроблений алгоритм. Схема алгоритму показана на рисунку 4.3.

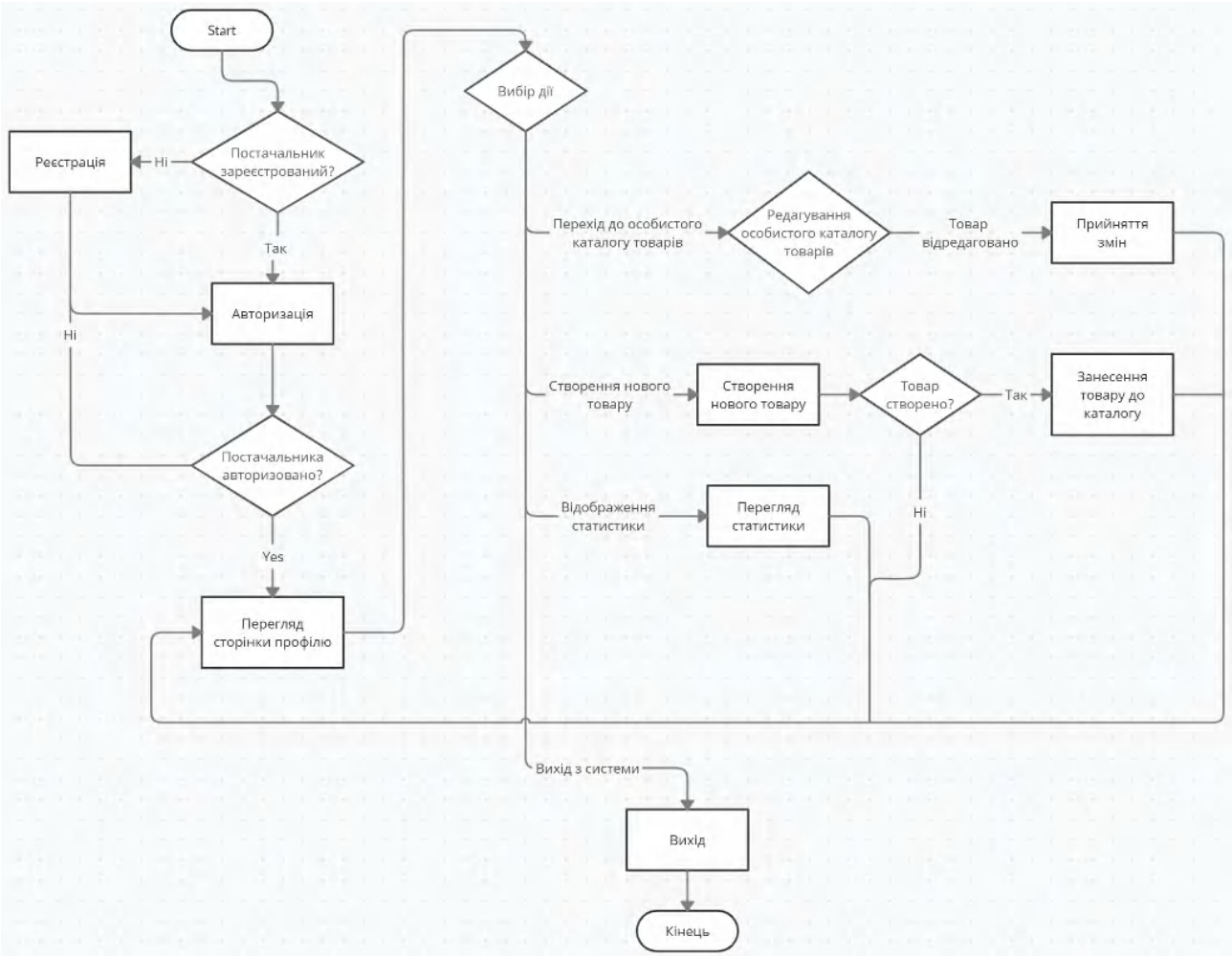

Рисунок 4.3 – Схема алгоритму роботи веб-служби «Робота з постачальниками»

Потік використання компонентів розробленої системи користувача «Постачальник» має наступну послідовність.

1 Блок «Наявність постачальника». Для користування розробленою веб-службою, кожен з постачальників повинен бути зареєстрований. Тому на початку роботи обов’язковим є процес реєстрації. Якщо такого користувача вже зареєстровано, тоді його автоматичне перенесе до авторизації. Для реєстрації постачальник, як рядовий користувач заповнює форму з необхідними даними (логін, пароль і назва компанії).

2 Блок «Авторизація». Після перевірки наявності чи реєстрації користувача потрібно авторизувати його у системі, що й робить цей блок

3 Блок «Перевірка існування користувача». Зазвичай коли користувач авторизується є невелика перевірка корректності вводу даних, якщо вони введені коректно і існують у системі користувач може продовжити роботу з системою як авторизований, інакше – помилка.

4 Блок «Сторінка профілю». Цей блок відповідає за особисту інформацію кожного користувача. У цьому просторі користувач має змогу корегувати свої дані чи просто ознайомитися з ними.

5 Блок «Створення продукту». Цей блок потрібен для обробки нових продуктів що бажає постачати кожен з постачальників. Тут присутня форма введення усіх необхідних даних про товар у певній послідовності, що задається класом продукту.

6 Блок «Редагування існуючого продукту». У будь-який момент переглядуючи товари які постачає постачальник може виявитися помилка в описі чи потреба змінити ціну. Саме цей блок віпводіає за ззміни у продукті.

7 Блок «Відображення статистики». У постачальників є можливість переглядати статистику продажів продукції, що вони постачають. На цій сторінці – детальна інформація, включаючи кількість вже куплених споживачем товарів, загальну вартість конкретного замовлення, тощо.

8 Блок «Вихід». Завершення робочого процесу постачальника на веб-сервісі, а також, збереження усіх корегованих змін.

4.5.2 Розробка споживчого сервісу з модулем аналізу

Далі, потрібно описати веб-сервіс для обслуговування клієнтів. Він має містити функціональні можливості, що потребує кожен користувач у системі, як простий користувач, так само й адміністратор. Для вірної структури функціоналу необхідно створити систему контролю, яка враховує ролі користувачів і, на основі цих даних робить висновок чи слід показати цю сторінку цьому користувачу.

Відобреднння поведінки споживача і адміністратора при роботі з реалізованим веб-сервісом реалізовано в алгоритмі, схема якого продемонстрована на рисунку 4.4.

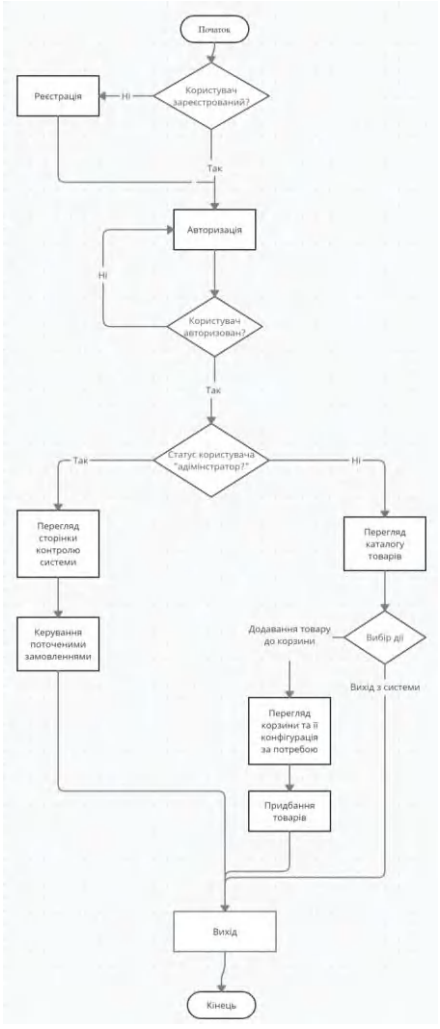

Рисунок 4.4 – Схема алгоритму роботи веб-служби «Обслуговування клієнтів»

Перелік компонентів розроблюваної системи у відповідності до побудованого алгоритму виглядає наступним чином:

1 Блок «Наявність користувача». Для користування розробленою веб-службою, кожен з користувачів повинен бути зареєстрований. Тому на початку роботи обов'язковим є процес реєстрації. Якщо такого користувача вже зареєстровано, тоді його автоматичне перенесе до авторизації. Для реєстрації кожен користувач, заповнює форму з необхідними даними (логін, пароль і назвповне ім'я).

2 Блок «Авторизація». Після перевірки наявності чи реєстрації користувача потрібно авторизувати його у системі, що й робить цей блок

3 Блок «Перевірка існування користувача». Зазвичай коли користувач авторизується є невелика перевірка корректності вводу даних, якщо вони введені коректно і існують у системі користувач може продовжити роботу з системою як авторизований, інакше – помилка.

4 Блок «Перевірка статусу». У розробленому веб-сервісі є два зареєстрованих типи користувачів – користувач і адміністратор. Саме для розподілу ролей у системі використовується цей блок.

5 Блок «Керування продуктом». Блок, що доступний тільки якщо статус користувача «Адміністратор». На відображеній сторінці є управління продуктами. Адміністратор бачить всі продукти і детальну інформацію про них.

6 Блок «Керування замовленнями». Сторінка тільки для адміністратору. Дозволяє підтвердити замовлення товарів або видалити будь-яке замовлення.

7 Блок «Каталог продукції». На цій сторінці будь-який користувач може переглянути всі наявні та існуючі позиції з каталогу магазина.

8 Блок «Кошик». Користувач має змогу придбати обрані товари з цієї сторінки безпосередньо. Після успішного виконання буде викликано перехід до каталогу.

9 Блок «Вихід». Користувачч завершують процес роботи над веб-сервісом та більше не мають доступу до певних функцій.

4.6 Розрахунки трудомісткості розробки

COCOMO (Constructive Cost Model) - це регресійна модель, заснована на LOC, тобто кількості рядків коду. Це процедурна модель оцінки вартості програмних проектів і часто використовується як процес надійного прогнозування різних параметрів, пов'язаних із створенням проекту, таких як розмір, зусилля, вартість, час і якість. Вона була запропонована Баррі Бомом у 1981 році і базується на дослідженні 63 проектів, що робить її однією з найбільш задокументованих моделей. Ключовими параметрами, які визначають якість будь-яких програмних продуктів, і які також є результатом Cocomo, є насамперед Effort (зусилля) та Schedule (графік):

Зусилля: Кількість праці, яка буде необхідна для виконання завдання. Вимірюється в одиницях людино-місяців.

Графік: Просто означає кількість часу, необхідного для завершення роботи, яка, звичайно, пропорційна докладеним зусиллям. Вимірюється в таких одиницях часу, як тижні та місяці.

Різні моделі COCOMO були запропоновані для прогнозування оцінки вартості на різних рівнях, виходячи з необхідного рівня точності та правильності. Всі ці моделі можуть бути застосовані до різноманітних проектів, характеристики яких визначають значення константи, що буде використовуватися в подальших розрахунках. Ці характеристики, що відносяться до різних типів систем, згадуються нижче. Визначення Бома щодо органічних, напіввідокремлених та вбудованих систем:

За для оцінки повного обсягу праці по розробці ПЗ (надалі, трудовитрат) у органічному режимі за моделю COCOMO використаємо таку формулу:

$$E = 2{,}4 * (KLOC)^{1{,}05}, \tag{4.1}$$

де E – є трудовитратами розробки ПЗ в межах одного ІТ-проекту, счислюється у людино-місяцях; $KLOC$ – показник тисяч строк коду, котрі потрібно впровадити в процесі розробки ПЗ.

Оцінка тривалості робіт розроблюваного ПЗ (далі, витрат часу) у органічному режимі за моделю COCOMO розраховується за формулою:

$$TDEV = 2{,}5*(E)^{0{,}38}, \tag{4.2}$$

де $TDEV$ – часові витрати (місяці) за для розробки ПЗ в межах одного ІТ-проекту.

Коли ми дізналися дві оцінки обсягів робіт, а також їх тривалості для одного ІТ-проекту, є змогу за допомогою моделі COCOMO обчислити потребу у персоналі і, що досить важливо, продуктивність праці одного залученого співробітника. Для цього використовуються такі формули:

$$SS = E / TDEV, \tag{4.3}$$

$$P = KLOC / E, \tag{4.4}$$

де SS – загальна кількість залучених співробітників одного ІТ-проекту з розробки ПЗ, людей; P – кількістна характеристика (тисяч строк коду), яку один співробітник на ІТ-проекті повинен писати за місяць.

Кількість тисяч строк коду KLOC у проекті може бути порахована з використанням різни моделей та методів. Найпростіший, а тому й ефетивніший, є спрощений метод функціональних точок..

Спрощений метод функціональних точок говорить, що складність одного ІТ-проекту через функціональні точки рахується за формулою:

$$FP = (C_1 + C_2 + C_3)^{2{,}35}. \tag{4.5}$$

Показники C_1, C_2 та C_3 можуть бути знайдені у таблиці 4.1.

Таблиця 4.1 – Класифікатори проекту розроблюваної системи

Масштаб проекту, С1		Користувачі об’єкту проектування, С2		Тип об’єкту проектування, С3	
Бібліотека об'єктів	4	Споживачі shareware ПО	2	ПЗ одноплатної системи, що вбудовується	5
Реалізація концепції	5	Академічне середовище, інженерія	3	База даних	6
Прототип для подальшого еволюційного розвитку	6	Внутрішньокорпоративне, локальне використання	5	Клієнт-серверне ПЗ	8
Програма під замовлення	9	Контрактний проект, цивільний замовник	7	Комунікаційне ПЗ	11
Програма, яка придатна до розширення функціональності в ході життєвого циклу	10	Контрактний проект, замовник - органи місцевої влади	8	ПЗ управління процесами	12
Компонент зовнішньої системи	11	Комерційний проект	9	ПЗ вбудовуваної многоплатної системи	13
Нова масштабна система	12	Контрактний проект, державне фінансування	14	ПЗ для загальнодоступних сервісів	15
Компонентна система	13	Військовий проект	15		

Показник *KLOC* визначається виходячи з показника *FP*, котрий по суті являє собою методику відкату. Слідючи цій методиці показник *KLOC* рахується за наступною формулою:

$$KLOC = k * FP . \quad (4.6)$$

Значення коефіцієнту *k* обирається з табл. 4.2, на основі обраної мови програмування або середовища розробки програмного забезпечення.

Таблиця 4.2 – Кількість строк коду, необхідних для реалізації одного балу функціональності в залежності від мови програмування

Мова	Строк коду на функціональний бал
Асемблер	320
C	148
C++	59
C#	54
Java	53
J2EE	50
JavaScript	54
.NET	60
HTML	43
PL/SQL	40
Python	24
PHP	67

Розразункова частина виглядає наступним чином. Спочатку розрахунок складності нашого IT-проекту – показник *FP*. Для нього визначаємо показники C_1, C_2, C_3:

C_1 = 6 (Прототип для подальшого еволюційного розвитку);

C_2 = 9 (Комерційний проект);

C_3= 8 (Клієнт-серверне ПЗ).

Тепер, коли показники визначені, розрахуємо *FP:*

$$FP = (5 + 5 + 8)^{2,35} = 1585.$$

Визначивши показник *FP*, можемо прорахувати показник *KLOC*, заздалегідь обравши коефіцієнт *k*. Цей проект буде реалізований за допомогою мови C#. З цього виходить, що коефіцієнт *k* = 54, а надалі слідує що:

$$KLOC = 54 * 1585 = 85\,596.$$

Знаючи цей показник (*KLOC*) можемо оцінити обсяг трудовитрат (кількість робіт за проектом) у органічному режимі за формулою 3.1:

$$E = 2{,}4 * (85\,596)^{1{,}05} = 362\,483.$$

Оцінка витрат часу у органічному режимі виконується за формулою 3.2, але тепер знаючи *E* ми спроможні його розрахувати*:*

$$TDEV = 2{,}5 * (362\,483)^{0{,}38} = 324.$$

Отримавши усі ці показники вираховуємо потребу в персоналі одного IT-проекту та продуктивність праці одного залученого програміста. Для цього скористаємося формулою 3.3:

$$SS = \frac{362\,483}{324} = 1119,$$

$$P = \frac{85\,596}{362\,483} = 0.24.$$

Отримані результати після усіх розрахунків такі: рівень складності IT-проекту – 1585, трудовитрати з розробки ПЗ – 362 483, тривалість робіт – 324, кількість залучених співробітників IT-проекту – 1119, кількість тисяч строк коду, які один програміст IT-проекту повинен писати за місяць – 0.24.

4.7 Розробка компонентів системи

4.7.1 Розробка компоненту зв’язку додатку з базою даних

Для доступу додатку до бази даних MySQL було реалізовано декілька класів, що у свою чергу реалізують патерн розробки «Репозиторій».

Клас перший «ClassSetupProgram.cs» (рис. 4.5): клас який зберігає налаштування для підключення БД.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Configuration;

namespace ShopASP.Models
{
    0 references
    public class ClassSetupProgram
    {
        0 references
        public string connectionToDBString = "Server=127.0.0.1;Database={};Username={};Password={};CharSet=utf8;";
        0 references
        public ClassSetupProgram() { }
    }
}
```

Рисунок 4.5 – Лістинг коду класу ClassSetupProgram.cs

Клас «ClassDataBase.cs» (рис. 4.6): містить методи виконання SQL-запитів від додатку до БД.

```
public class ClassDataBase
{

    #region ExecuteNonQuery ···

    #region Execute

    1 reference
    protected T GetObject<T>(params object[] args)
    {
        return (T)Activator.CreateInstance(typeof(T), args);
    }

    0 references
    public void Execute<T>(ref ClassSetupProgram setupProgram, string sSql, ref List<T> listResult)
    {
        string result = "";
        try
        {
            string databaseName = setupProgram.connectionToDBString;
            MySqlConnection con = new MySqlConnection(setupProgram.connectionToDBString);
            con.Open();

            MySqlCommand command = new MySqlCommand(sSql, con);
            MySqlDataReader dataReader = command.ExecuteReader();

            if (dataReader.HasRows)
            {
                while (dataReader.Read())
                {
                    result = "";
                    for (int i = 0; i < dataReader.FieldCount; i++)
                    {
                        result += dataReader.GetString(i) + "!";
                    }
                    if (result.Count() > 2) result = result.Remove(result.Count() - 1);
                    if (result != "") listResult.Add(GetObject<T>(result));
                }
            }
            con.Close();
        }
        catch (Exception ex)
        {

        }
    }
    #endregion
}
```

Рисунок 4.6 – Лістинг коду класу ClassDataBase.cs

Клас «EFDbContext.cs» (рис 4.7): клас у якому перелічені усі функції що викликаються у базі даних.

```
public class EFDbContext
{
    16 references
    public ClassDataBase db = new ClassDataBase();
    16 references
    public ClassSetupProgram stp = new ClassSetupProgram();
    10 references
    public List<Cake> Cakes = new List<Cake>();
    9 references
    public List<Order> Orders = new List<Order>();
    9 references
    public List<User.UserList> UserList = new List<User.UserList>();
    3 references
    public List<Flag> FilterFlag = new List<Flag>();

    0 references
    public string getFlag()
    {
        FilterFlag.Clear();
        db.Execute<Flag>(ref stp, "SELECT * FROM cake.testblob;", ref FilterFlag);
        return FilterFlag[0].FlagStatus;
    }

    0 references
    public void SetFlag(string value)
    {
        db.ExecuteNonQuery(ref stp, "UPDATE `cake`.`testblob` SET `Content` = '"+ value +"' WHERE (`ID_Image` = '1');");
    }

    2 references
    public List<Cake> getCakes() {

        Cakes.Clear();

        db.Execute<Cake>(ref stp, "SELECT * FROM `cake`.`cakes` where Quantity > 0;", ref Cakes);

        return Cakes;
    }

    0 references
    public List<User.UserList> getUsers() {

        UserList.Clear();

        db.Execute<User.UserList>(ref stp, "SELECT * FROM cake.users;", ref UserList);

        return UserList;
    }
```

Рисунок 4.7 – Лістинг коду класу EFDbContext.cs

Клас «Repository.cs» (рис 4.8): клас-реалізація патерну «Репозиторій», тобто зберігання кожного «звернення» від EFDbContext до бази даних.

```
public class Repository
{
    9 references
    private EFDbContext context = new EFDbContext();


    0 references
    public string getFlag
    {
        get { return context.getFlag(); }
    }

    0 references
    public void setFlag(string value)
    {
        context.SetFlag(value);
    }


    0 references
    public List<Cake> Cakes
    {
        get { return context.getCakes(); }
    }

    0 references
    public IEnumerable<Order> Orders
    {
        get
        {
            return context.getOrders();
        }
    }

    0 references
    public IEnumerable<User.UserList> Users
    {
        get
        {
            return context.getUsers();
        }
    }

    0 references
    public void SaveOrder(Order order)
    {
        context.insertOrder(order);
    }
```

Рисунок 4.8 – Лістинг коду класу Repository.cs

4.7.2 Розробка компоненту системи, що реалізує аналіз попиту на товари

Для реалізації корзини товарів у додатку було розроблено клас «Analize.py» (рис 4.9): цей клас є контейнером зовнішнім Python скриптом, зо приймає та надсилає дані за допомогою API за мікросервісної архітектури REST.

```
price_range = np.arange(0.5, 3.5, 0.01)
df_price_range = pd.DataFrame({'Mean_Price': price_range})

y_proba = lr.predict_proba(df_price_range)
purchase_proba = y_proba[:][:, 1]

df_purchase_proba = pd.DataFrame({'Purchase_Proba': purchase_proba})
df_price_proba = pd.concat([df_price_range, df_purchase_proba], axis=1)
# df_price_proba.head(50)

fig, ax = plt.subplots(figsize=(12, 8))

sns.lineplot(data=df_price_proba, x='Mean_Price', y='Purchase_Proba', ax=ax)
ax.fill_between(data=df_price_proba, x='Mean_Price', y1='Purchase_Proba', alpha=0.2)

plt.show()

pe = lr.coef_[:, 0] * price_range * (1 - purchase_proba)

df_pe = pd.DataFrame({'Mean_Price_Elasticity': pe})
df_price_elasticity = pd.concat([df_price_proba, df_pe], axis=1)
df_price_elasticity

fig, ax = plt.subplots(figsize=(16, 8))

sns.lineplot(data=df_price_elasticity, x='Mean_Price', y='Mean_Price_Elasticity', ax=ax)

plt.arrow(1.25, -2.5, 0, 1.3, head_width=0.025, head_length=0.1, color='#FF8400')

plt.plot([0.5, 1.22], [-1, -1], color='green')
plt.annotate("Increase Price", xy=(0.7, -1.4), color='green', fontsize=15)
plt.plot([1.28, 3.5], [-1, -1], color='red')
plt.annotate("Decrease Price", xy=(2.0, -1.4), color='red', fontsize=15)

plt.show()

seg_list = ['standard', 'career-focused', 'fewer-opportunities', 'well-off']

for seg in seg_list:
    df_data = df_purchase.copy()
    df_data = df_data[df_data['Segment'] == seg]

    X = pd.DataFrame()
    X['Mean_Price'] = df_data['Mean_Price']
    y = df_data['Incidence']

    lr = LogisticRegression(solver='sag')
    lr.fit(X, y)
```

Рисунок 4.9 – Лістинг коду класу Analize.py

А також на самому представлені була реалізована функція відображення результатів. Цей клас має назву «Visualize.py», його лістинг наведено нижче на рисунку 4.10, а реалізація на рисунку 4.11

```
df_segment = df_purchase[['ID', 'Segment']].drop_duplicates()
segments = df_segment['Segment'].value_counts()

segments.plot.pie(figsize=(6, 6), autopct='%1.0f%%', label='') # startangle=0
p = plt.gcf()
p.gca().add_artist(plt.Circle((0,0), 0.5, color='white'))

plt.show()

plt.figure(figsize=(8, 6))
s = sns.countplot(data=df_purchase, x='Segment', hue='Incidence')
plt.xlabel('Segment')
plt.ylabel('Incidence')
plt.show()

df_segment_incidence = df_purchase[df_purchase['Incidence'] == 1].copy()

fig, axs = plt.subplots(1, 2, figsize=(16, 8))

sns.boxplot(x='Segment', y='Mean_Price', data=df_segment_incidence, ax=axs[0])
sns.boxplot(x='Segment', y='Action_Price', data=df_segment_incidence, ax=axs[1])

plt.show()

X = pd.DataFrame()
X['Mean_Price'] = df_purchase['Mean_Price']
y = df_purchase['Incidence']

lr = LogisticRegression(solver='sag')
lr.fit(X, y)

print(lr.coef_)
```

Рисунок 4.10 – Лістинг коду класу «Visualize.py»

Рисунок 4.11 – Уявлення «Візуалізація результатів аналізу»

4.7.3 Розробка компонентів для виводу основної інформації

Проаналізувавши та провевши моделювання структура MVC була найбільш оптимальною, де видимі частини додатку є представленнями, тобто «View», а їх код досить сильно нагадує один одного.

Приклад представлення «Listing.aspx» (рис 4.12):

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Listing.aspx.cs" Inherits="ShopASP.Pages.Listing"
    MasterPageFile="~/Pages/Cakes.Master" %>
<%@ Import Namespace="System.Web.Routing" %>

<asp:Content ContentPlaceHolderID="bodyContent" runat="server">
    <div id="filters">
        Фильтры:   
        <span>Цена
            от
            <input id="bottom_input_filter" name="bottom_input_filter" type="number" min="0" max="9999" runat="server"/>
            до
            <input id="upper_input_filter" name="upper_input_filter" type="number" min="0" max="9999" runat="server"/>
            <button name="confirm_filter" type="submit">
                        Применить
            </button>
              
            Сортировать цену по:
            <select id="sorting" name="storing" runat="server">
              <option>Увеличению</option>
              <option>Уменьшению</option>
            </select>
            <button name="confirm_sorting" type="submit" value = "4">
                        Применить
            </button>
            <button name="cancel_filter" type="submit" value = "0">
                        Сбросить фильтры
            </button>
        </span>

    </div>
    <div id="content">
            <asp:Repeater ItemType="ShopASP.Models.Cake"
                SelectMethod="GetCakes" runat="server">
                <ItemTemplate>
                    <div class="item">
                        <img src="/Content/<%# GetImagePath(Item.CakeId) %>.jpg" alt="this_cake" style="float: left; width: 75px; height: 75px; margin-right: 1.5%;"/>
                        <h3><%# Item.Name %></h3>
                        <%# Item.Description %>
                        <h4>₴ <%# Item.Price.ToString() %></h4>
                        <button name="add" type="submit" value="<%# Item.CakeId %>">
                            Добавить в корзину
                        </button>
                    </div>
                </ItemTemplate>
            </asp:Repeater>
     </div>
    <div class="pager">
        <%
            for (int i = 1; i <= MaxPage; i++)
```

Рисунок 4.12 – Лістинг представлення Listing.aspx

5 ТЕСТУВАННЯ (ВЕРИФІКАЦІЯ) РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Навантажувальне тестування

На той час, коли будь-який проект з розробки програмного забезпечення наближається до завершення, він, ймовірно, пройде через численні тести, особливо в середовищі Agile-тестування, де тестування і розробка відбуваються одночасно. Але незалежно від того, скільки тестів ви провели, після того, як ваша програма майже завершена, є тільки один спосіб дізнатися, чи може ваше програмне забезпечення впоратися з реальними вимогами, які незабаром будуть пред'являти до нього армія кінцевих користувачів. Це називається навантажувальним тестуванням, і ви можете використовувати такий інструмент, як Load Testing Tool, щоб виконати цю роботу. Навантажувальне тестування - це процес створення імітованого попиту на програмне забезпечення, додаток або веб-сайт таким чином, щоб перевірити або продемонструвати його поведінку в різних умовах[23].

Навантажувальне тестування – полягає у створенні виробничих симуляцій в додатку або системі, які максимально наближені до готового продукту, готового до розгортання і підпорядкованого масам. Використовуючи спеціалізоване програмне забезпечення для тестування, навантажувальне тестування дозволяє командам розробників відповісти на такі питання, як "Чи робить моя система те, що я очікую в цих умовах?" і "Чи достатньо хороша її продуктивність?" Як зазначено в керівництві Microsoft "Керівництво з тестування продуктивності веб-додатків":

«…навантажувальний тест дозволяє виміряти час відгуку, пропускну здатність і рівень використання ресурсів, а також визначити точку зламу вашого додатка, припускаючи, що точка зламу відбувається нижче пікового навантаження…»[24]

Тут слова "нижче пікового навантаження" просто передбачають, знову ж таки, методологію тестування, яка підпадає під параметри навантажувального тесту, на відміну від стрес-тесту (який, за визначенням, тестує систему при піковому навантаженні і за його межами). Навантажувальне тестування може виявити відставання системи, проблеми з завантаженням сторінок і все інше, що може піти не так, коли кілька користувачів отримують доступ до програми або бомбардують систему раптовим трафіком - речі, які можна легко не помітити в середовищі розробки і тестування, де код часто перевіряється з урахуванням індивідуальних користувачів. Однак додайте до цього сотню або тисячу людей, які намагаються отримати доступ до програмного забезпечення або віддавати команди більш-менш одночасно, і проблеми, які могли б не бути виявлені у випадках індивідуального використання, можуть раптово проявитися у всій їхній помилковій красі.

Таблиця 5.1 – Результати ручного тестування

Тест	Результат тестування
Завантаження веб-служб користувачів, з урахуванням ролей	Завантажується сервісу без помилок
Перевірка корректності роботи зовнішньої бази даних	Запуск сервісу та успішне підключення до зовнішньої бази даних.
Перевірка відповідності прописаним вимогам	Веб-сервіс оброблює GET та POST запити успішно та виводить усі отримані дані
Перевірка коректності роботи панелі користувача	Натискаючи на будь-яке посилання сервіс перемикає сторінки коректно та без затримок

Продовження таблиці 5.1

Тест	Результат тестування
Перевірка аутентифікаційного модулю	Усі помилки при вводі даних користувача фіксуються і повідомляються користувачеві
Коректність текстових повідомлень та їх відображень як реація на невірні дії користувача	При користуванні додатком кожна помилка чи успіх фіксуються та повідомляються відповідним повідомленням
Коректність створення продукції у каталогу	При усіх введених коректних даних продукт додано до каталогу і відображену в загальному переліку, інакше – помилка
Функція корегування продукції	Після зміни будь-яких даних продукту і коректному вводі нових даних – продукт оновлено.
Коректність обліку товарів у базі даних	Якщо товар був куплений (тобто на нього оформлено замовлення) його кількість на складі зменьшилась

Ручне тестування проведено успішно, усі тести виконані та отриманий результат є коректним. Система відповідає розробленому функціоналу та очікуванням на кожну з функцій, тому може використовуватися.

Також, на початкових етапах проектування було прийнято рішення щодо проведення тестування додатку на продуктивність. Тестування продуктивності - це метод тестування, який визначає швидкість, масштабованість і стабільність роботи програми під заданим робочим навантаженням. Воно допомагає забезпечити якість програмного забезпечення і робить додаток готовим до випуску на ринок [23].

– Швидкість – швидкість, з якою додаток реагує на запити.

– Масштабованість – максимальне користувацьке навантаження, з яким може впоратися додаток.

– Стабільність – стан додатку при різних навантаженнях.

Щоб зрозуміти, як додаток буде працювати після запуску, існує багато різних типів тестів продуктивності, які проводяться на основі багатьох факторів. Ось деякі з найбільш поширених.

а) Об'ємне тестування - основною метою об'ємного тестування є перевірка продуктивності програми в різних об'ємах бази даних. Поведінка програми контролюється шляхом заповнення різних обсягів даних в базу даних.

б) Навантажувальне тестування - основною метою навантажувального тестування є визначення основної точки зламу програмного додатку. Це робиться шляхом тестування програми під екстремальними робочими навантаженнями, щоб оцінити її продуктивність в умовах високого трафіку або обробки даних.

в) Спік-тестування - основною метою спік-тестування є перевірка реакції програми при раптовому великому стрибку навантаження (згенерованому користувачами).

г) Тестування масштабованості - основною метою тестування масштабованості є визначення того, чи може додаток ефективно масштабуватися в разі перевантаження користувачів. Це тестування також допомагає планувати додавання потужностей до вашого додатку на майбутнє.

г) Навантажувальне тестування - основною метою навантажувального тестування є виявлення вузьких місць в продуктивності або здатності програми працювати при очікуваному користувацькому навантаженні.

д) Тестування на витривалість - тестування на витривалість проводиться для того, щоб переконатися, що програмне забезпечення може впоратися з очікуваним навантаженням протягом тривалого періоду часу. [23]

Тож, розглянемо один з таких видів тестування, та взагалі принцип тестування продуктивності. Будемо це робити на прикладі навантажувального тестування. Навантажувальне тестування (load testing) – це тестування дозволяє зробити оцінку поведінки системи коли навантаження на будь-яку її компоненту зростає. За мету навантажувального тестування прийнято вважати також визначення того яке максимальне навантаження може витримати система (рис 5.1).

Рисунок 5.1 – Результат навантажувального тестування на SRM-систему

По проведеному тесту зрозуміло, що наша система працює достатньо швидко та має високу продуктивність. Адже при наявності продуктивності більше, ніж 90, сторінка в браузері завантажуватимуться майже миттєво.

5.2 Забезпечення захисту інформації, що циркулює в системі

Оцінка безпеки додатків - це процес тестування додатків для виявлення загроз і визначення заходів, які необхідно вжити для захисту від них. За допомогою процесу оцінки організації можуть оцінити поточний стан безпеки своїх додатків і визначити наступні кроки для подальшого захисту свого

програмного забезпечення від майбутніх атак. Більшість організацій проводять оцінку безпеки додатків на регулярній основі, щоб забезпечити актуальність і ефективність своїх заходів безпеки [25].

Ретельна оцінка безпеки додатків може дозволити організаціям виявити потенційні загрози для свого програмного забезпечення і додатків до того, як вони стануть проблемою. Інциденти безпеки є суттєвим ризиком для сучасного бізнес-середовища, що базується на програмному забезпеченні, оскільки вони можуть мати негативний вплив на репутацію та доходи компанії. У багатьох галузях промисловості оцінка безпеки додатків може бути навіть необхідною для дотримання законів і нормативних актів у сфері кібербезпеки. Наприклад, стандарти PCI передбачають дотримання принципів OWASP Top 10 [25].

– Визначення потенційних суб'єктів загрози. Першим кроком при проведенні оцінки безпеки додатку є визначення того, хто з найбільшою ймовірністю може становити загрозу для вашого додатку.

– Визначте конфіденційні дані, які варто захистити. Після того, як ви визначили, хто може атакувати ваш додаток, важливо визначити, що варто захищати.

– Складіть «road-map» атаки додатку. Сучасні хмарні додатки складаються з багатьох компонентів, таких як користувацький код, залежності з відкритим кодом, контейнери, інфраструктура у вигляді коду тощо.

– Оцініть больові точки процесу забезпечення безпеки додатку. Після того, як ви зрозуміли ризики для додатків, корисно визначити, чому ці ризики існують, оцінивши ваш поточний процес AppSec.

– Створіть «road-map» безпеки. Після того, як ви провели ретельний аналіз зловмисників, і потенційних шляхів атаки, корисно створити дорожню карту для усунення слабких місць в ваших процесах AppSec[25].

Проаналізувавши та впровадивши більшу частину перелічених пунктів можемо сміливо казати, що наш додаток розроблений з урахуванням останніх стандартів та є достатньо захищеним від втручання ззовні.

ВИСНОВКИ

Під час розробки кваліфікаційної роботи було розроблено SRM-систему для фабрики кондитерських виробів, а також додано універсальний модуль аналізу попиту ринку. Система являє собою клієнт-серверний додаток з базою даних. Спроектований і розроблений модуль може бути використаний для аналізу попиту та збільшення прибутковості підприємства в подальшому.

При виконанні проектування до кваліфікаційної роботи було проведено функціональне моделювання. Були проаналізовані усі доступні методи аналізу попиту. Для багатокритеріального аналізу були обрані найкращі фактори впливу.

В результаті аналізу виявлено, що найбільш вигідним і найбільш ефективним буде впровадження в систему комбінованого методу аналізу попиту кондитерської продукції, а саме комбінування методів лінійної регрессії, корегування помилок, а також методів цінової еластичності попиту. Порівнявши кожен з цих методів та їх аналогів було виведено, що методи найбільш оптимальні та можуть використовуватися один з одним підвищуючи прибутковсість підприємтсва досить швидко.

Результатом дослідження стала SRM-система для кондитерського виробництва що дозволяє напрочут ефективно вести торгівельний бізнес, ураховувати тенденції на ринку та корегувати ціни, не забуваючи про складський обіг продукції та роботу з поставниками, що є одними з найважливіших функцій такої системи.

Удосконалення розробленої аналітичної SRM-системи включає в себе такі напрями:

– додавання нових функцій. Для комфортного користування системою потрібно впроваджувати нові можливості, що покращать взаємодію з системою. Інтеграція прямою взаємодії модулю аналізу замість стороннього сервісу.

– покращення безпеки. Впровадження та використання інформаційної системи в бізнесі передбачає провеведнння роботи пошуку вразливостей у системі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки до організації виконання та захисту кваліфікаційної роботи на здобуття вищої освіти ступеня магістра спеціальності 122 Комп'ютерні науки, освітньо-професійна програма «Інформаційні технології проектування» / Упорядники: І.В. Гребеннік, В.Г. Іванов, Н.І. Калита, А.І. Коваленко, Д.Е.Ситніков, І.А. Урняєва. Харків: ХНУРЕ, 2021. 52 с.

2. ДСТУ 3008:2015. «Звіти у сфері науки і техніки. Структура та правила оформлення» - Український інститут науково-технічної і економічної інформації; Технічний комітет стандартизації «Інформація і документація» (ТК 144) В. Земцева; Ю. Поліщук, канд. фіз.-мат. наук; Р. Санченко, канд. техн. наук; Л. Шрамко; А. Ямчук (науковий керівник) від 22 червня 2015 р. 20 с.

3. Промо-сайт кондирської фабрики «Солодкий світ» URL http://sladkiymir.com.ua/ (дата звернення 19.10.22)

4. Промо-сайт кондирської фабрики «Світоч» URL https://svitoch.ua/ (дата звернення 19.10.22)

5. Промо-сайт кондирської фабрики «АВК» URL https://avk.ua/ua/uk (дата звернення 19.10.22)

6. Синіцин С. В., Налютин Н. Ю. Верификация программного обеспечения. М.: БИНОМ, 2008. 368 с.

7. Савченко С. С., Морозова А. І. Впровадження SRM-систем у реальне виробництво // Topical issues of modern science, society and education. Proceedings of the 6th International scientific and practical conference. SPC "Sci-conf.com.ua". Kharkiv, Ukraine. 2021. Pp. 468-469.

8. Просветов Г. И. Экономический анализ: задачи и решения: учебно-практическое пособие / Г.И. Просветов. – Москва: Издательство «Альфа-Прес», 2008. – 640 с.

9. Промо-сайт кондирської фабрики «Бісквіт-шоколад» URL https://biscuit.com.ua/ (дата звернення 20.10.22)

10. Управління закупівлями та поставками / М.Линдерс, Ф.Джонсон, А.Флинн, Г.Фирон : 2013. 752 с.

11. Коннолли Т., Бегг К., Страчан А. База даних. Проектування, реалізація та супровід: Теорія і практика: Видавничий дім «Вільямс», 2003. – 1436 с.

12. Калянов Г.Н. CASE - технології: Консалтинг в автоматизації бізнес-процесів. - 3-є видання. - М.: Гаряча лінія-Телеком : 2008. – 320 с.

13. Мацяшек Л. Аналіз вимог і проектування систем. Розробка інформаційних систем з використанням UML / Пер. з англ. - М.: Видавничий дім «Вільямс», 2002. 432 с..

14. Савченко С.С., Колесник Л.В. Наслідки підвищення цін на кондитерські вироби та автоматизація розрахунку цінової еластичності попиту // Modern science: innovation and prospects. Proceedings of XVI International Scientific and Practical Conference, 11-13 December 2022. Stockholm, Sweden. Pp. 176-182.

15. Основы маркетинга: 5-е европейское издание / [Котлер Ф., Армстронг Г., Вонг В., Сондерс Д.]. – Москва: Издательский дом «Вильямс». – 2013. – 752 с.

16. Шкварчук Л. Дослідження особливостей формування попиту на продовольчому ринку України / Л. Шкварчук // Маркетинг в Україні. — 2009. – № 6. – с. 44–48

17. Албахари Джозеф, Албахари Бен. C# 7.0. Справочник. Полное описание языка, Пер. с англ. — СПб.: Альфа-книга, 2018. — 1026 с/

18. Макконнелл С. Совершенный код. Мастер класс 2-е издание. — СПб.: Питер; М.: Русская редакция, 2010. — 889 с.

19. Лутц, Марк. Изучаем Python, том 1, 5-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 832 с

20. Коннолли, Т., Бегг, К., Страчан, А. Базы данных. Проектирование, реализация и сопровождение: Теория и практика : Пер. с англ. / Т. Коннолли, К. Бегг, А. Страчан. – М.: Издательский дом «Вильямс», 2003. – 1436 с.

21. Савченко С., Колесник Л.В. Застосування E-SRM в електронній комерції // Інформаційні технології в соціокультурній сфері, освіті та економіці: матеріали VI Міжнар. наук.-практ. конф. студентів і молодих учених, м. Київ, 19-20 квітня 2022 р. Київ: Видивничій. центр КНУКіМ, 2022. С.132–133.

22. ДСТУ 34.601-90 - Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення : Україна, 1992. 6 с.

23. Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем. СПб.: Питер, 2004. 320 с.

24. Калбертсон Роберт, Браун Крис, Кобб Гэри. Быстрое тестирование. М.: «Вильямс», 2002. 374 с.

25. Damien V. Puyvelde, Aaron F. Brantly. Cybersecurity: politics, governance and conflict in cyberspace. Polity Press, Cambridge, UK, 2019. 212 c.

26. E. Petrov, N. Brynza, L. Kolesnik, O. Pisklakova. Methods and models of decision-making under conditions of multi-criteria and uncertainty // Grin DS, Kherson, 2014. T.2. C.54.

27. Andreyeva, Tatiana et al. “The Impact of Food Prices on Consumption: a Systematic Review of Research on the Price Elasticity of Demand for Food.” American journal of public health 100.2 (2010): 216–222.

28. ISO 690:2010 Information and documentation — Guidelines for bibliographic references and citations to information resources. – 3th ed 2010-06-15 (Інформація та документація. Настанови щодо бібліографічних посилань і цитування інформаційних ресурсів – 3-тє вид. 2010-06-15).