ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА СИСТЕМНОГО АНАЛІЗУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «ПРОЄКТНІ РІШЕННЯ ІНФОРМАЦІЙНІ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ РЕСТОРАННОГО БІЗНЕСУ»

на здобуття освітнього ступеня магістра

зі спеціальності 124 Системний аналіз

освітньо-професійної програми Інтелектуальні системи управління

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Ярослав ЯЦИШЕН
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: Здобувач вищої освіт
групи САДМ-61
Ярослав ЯЦИШЕН
Керівник: Ровіл НАФЄЄВ
Рецензент: _____

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙ

Кафедра Системного аналізу

Ступінь вищої освіти Магістр

Спеціальність 124 Системний аналіз

Освітньо-професійна програма Інтелектуальні системи управління

ЗАТВЕРДЖУЮ
Завідувач кафедрою СА
_____Ровіл НАФЄЄВ
«____» _____20___р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Яцишен Ярослав Юрійович

1. Тема кваліфікаційної роботи: Проєктні рішення інформаційні системи підтримки прийняття рішень для ресторанного бізнесу
керівник кваліфікаційної роботи Ровіл НАФЄЄВ к.е.н. ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій
від «___» _____20___р. № ____

2. Строк подання кваліфікаційної роботи «___» _____20___р.

3. Вихідні дані до кваліфікаційної роботи: інформація та дані про становище рестораного бізнусу

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Проєктні рішення інформаційні системи підтримки прийняття рішенньь для ресторанного бізнесу
2. Моделі та методи програзування попиту на продаж продукції ресторанного бізнесу
3. Вибір засобів та розробка програмного забезпеченння

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «___» _____20___р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	23.05.2023-21.06.2023	виконано
2	Аналіз предметної області	22.06.2023-29.06.2023	виконано
3	Постановка задачі	30.06.2023-08.07.2023	виконано
4	Аналітичний огляд методів розв'язання задачі прогнозування	09.07.2023-19.08.2023	виконано
5	Моделі та методи прогнозування попиту на продаж продукції ресторанного бізнесу	21.08.2023-29.09.2023	виконано
6	Вибір технологій реалізації	30.09.2023-09.10.2023	виконано
7	Вибір засобів та розробка програмного забезпечення	10.10.2023-19.11.2023	виконано
8	Особливості програмної реалізації алгоритму прогнозування попиту	20.11.2023-14.12.2023	виконано
9	Попередній захист роботи	15.12.2023	виконано
10	Здача роботи в деканат		виконано

Здобувач вищої освіти _____ Ярослав ЯЦИШЕН
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Ровіл НАФЄЄВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 77 стор., 27 рис., 10 табл., 30 джерел.

Мета роботи – є аналіз бізнес-процесів, пов'язаних з бізнесом обслуговування, як в кав'ярного так і ресторанного секторів, а також розробка програмного забезпечення, спрямованого на підвищення ефективності та збільшення обсягів продажу.

Об'єкт дослідження – є інформаційна система підтримки прийняття рішень для бізнесу обслуговування в контексті кавової та ресторанної галузей.

Предмет дослідження – є методи та засоби управління бізнесом обслуговування, що включають стратегії, технології, процеси та інструменти, які сприяють ефективному управлінню та оптимізації бізнес-процесів в сфері обслуговування в кавових та ресторанних підприємствах.

Короткий зміст роботи: На основі проведених досліджень був розроблений продукт для прогнозування продажів у бізнесі обслуговування, зокрема в кавовому та ресторанному секторах. Основними принципами, що лежали в основі розробки даного продукту, були безкоштовність та його доступність для широкого кола користувачів, оскільки він може бути легко використаний на будь-яких операційних системах родини Windows.

КЛЮЧОВІ СЛОВА:
Ресторан, ресторанний бізнес, розробка, покращення, бази даних, штучний інтелект, штучна нейронна мережа, C#.

ABSTRACT

Text part of the master's qualification work: 77pages, 27 pictures, 10 table, 30 sources.

The purpose of the work is to analyze the business processes related to service businesses in both the coffee and restaurant sectors. Additionally, it involves the development of software aimed at improving efficiency and increasing sales volumes.

Object of research is an information decision support system for service businesses in the context of the coffee and restaurant industries.

Subject of research – methods and means of managing service businesses, including strategies, technologies, processes, and tools that contribute to effective management and optimization of business processes in the service sector of coffee and restaurant enterprises.

Summary of the work: Based on the conducted research, a product for sales forecasting in the service industry, particularly in the coffee and restaurant sectors, was developed. The main principles underlying the development of this product were its cost-effectiveness and accessibility for a wide range of users, as it can be easily utilized on any Windows operating system.

KEYWORDS: Restaurant, restaurant business, development, improvement, databases, artificial intelligence, artificial neural network, C#.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЙ

ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня магістра

Направляється здобувач Яцишен Я. Ю. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 124 Системний аналіз
освітньо-професійної програми Інтелектуальні системи управління
5. на тему: «Проєктні рішення інформаційні системи підтримки прийняття рішень для ресторанного бізнесу ».

Кваліфікаційна робота і рецензія додаються.

Директор ННІТ _____ Владислав КРАВЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Здобувач Яцишен Ярослав Юрійович показав свою інженерну підготовку разом із володінням теоретичного метеріалу із поглибленим розумінням методів збереження, добуття та аналізу даних, адже якістно вміє володіти новими комп'ютерними технологіями та вміє детально користуватися навчальною, довідкиновою, і науково-технічною літературами. Всі поставлені задачі виконувались сумліно та якісно згідно виставленого технічного завдання.

Все це дозволяє оцінити виконану кваліфікаційну роботу здобувача Яцишена Ярослава Юрійовича оцінку «_____» та присвоїти йому(їй) кваліфікацію «Магістр з системного аналізу».

Керівник кваліфікаційної роботи _____ Ровіл НАФЄЄВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

«____» _____20___року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач *Яцишен Я. Ю.* допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедрою Системного аналізу _____ Ровіл НАФЄЄВ
(назва) *(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

«____» _____20___року

ВІДГУК РЕЦЕНЗЕНТА

на кваліфікаційну магістерську роботу

здобувача вищої освіти Яцишен Ярослава Юрійович

на тему «Проєктні рішення інформаційні системи підтримки прийняття рішень для ресторанного бізнесу»

Актуальність.

Дана дипломна робота була спрямована на розробку інформаційної системи для підтримки прийняття рішень в галузі бізнесу обслуговування. Основною метою цієї системи було полегшити роботу менеджерів та аналітиків шляхом надання зручного перегляду даних, можливості додавання, видалення та редагування записів, а також здійснення пошуку та фільтрації за допомогою запитів.

В рамках дипломної роботи було розглянуто основні методи прогнозування економічних процесів та проведено їх порівняльний аналіз. Для реалізації системи підтримки прийняття рішень в галузі бізнесу обслуговування були обрані відповідні засоби розробки, а також обґрунтовано вибір методу прогнозування.

Позитивні сторони.

1. Автором проведено детальний аналіз предметної галузі
2. Проведено аналіз компонентів технології, можливих ризиків переходу до технології системи прийняття рішення
3. Створення робочого програмного забезпечення, яке допоможе в розвитку у сфері ресторанного бізнесу

Недоліки.

1. В роботі реалізовані поставлені задачі ,але це мінімум для проведення в реальних умовах
2. Робота містить незначну кількість стилістичних та синтаксичних помилок

Відзначені зауваження не впливають на загальну позитивну оцінку кваліфікаційної роботи

Висновок: кваліфікаційна робота на здобуття ступеня магістра заслуговує оцінку "_____", а здобувач Яцишен Ярослав Юрійович заслуговує присвоєння кваліфікації: «Магістр з системного аналізу»

Рецензент:

науковий ступінь, вчене звання	*підпис*	*Ім'я, ПРІЗВИЩЕ*

ЗМІСТ

ВСТУП

У сучасному бізнес-середовищі, керівники стикаються з необхідністю прийняття зважених та якісних рішень, що є одним з головних завдань. Ефективність діяльності компаній безпосередньо залежить від якості та своєчасності прийнятих рішень. Незважаючи на це, найпоширеніші методи прийняття рішень не відповідають сучасним вимогам, які характеризуються перенасиченістю інформацією та швидким темпом життя. З метою вирішення цих проблем, доцільно залучити спеціалізовані автоматизовані системи - системи підтримки прийняття рішень (СППР).

Система підтримки прийняття рішень є інформаційною системою, яка пропонує допомогу особам, що займаються прийняттям рішень (ОПР), в умовах складних ситуацій, що характеризуються неповнотою вихідної інформації, багатьма критеріями оцінки та умовами, які впливають на повний та об'єктивний аналіз предметної діяльності. Застосування систем підтримки прийняття рішень, які базуються на різних математичних методах, дозволяє замінити людські ресурси на етапі вибору ефективного рішення.

Системи підтримки прийняття рішень (СППР) є автоматизованими системами, спеціально розробленими для ефективного вирішення завдань повсякденного управління та надання допомоги особам, які приймають рішення. Вони дозволяють здійснювати вибір рішень для неструктурованих і слабо структурованих завдань, зокрема багатокритеріальних. Розробка СППР часто є результатом мультидисциплінарних досліджень, що об'єднують теорію баз даних, штучний інтелект, інтерактивні комп'ютерні системи та методи імітаційного моделювання.

Сучасні умови бізнесу, характеризуються нестабільністю економічного середовища та зростаючою жорсткою конкуренцією, висувають високі вимоги до оперативності та якості прийняття рішень на всіх рівнях управління підприємством або організацією. Підтримка прийняття рішень передбачає наявність актуальної та всебічної інформації про стан та тенденції розвитку

бізнесу, а також застосування методів та засобів інтелектуального аналізу даних. Однак обсяг інформації, який необхідно враховувати для формування оптимальних та обґрунтованих рішень, неперервно зростає.

В сучасних умовах неможливо ефективно управляти компанією без застосування сучасних інформаційних засобів, зокрема методів та засобів автоматизації бізнес-аналізу. Це вимагає використання технологій бізнес-аналітики, які дозволяють організаціям перетворювати накопичені дані в інформацію про бізнес, а потім цю інформацію перетворювати на знання для управління бізнесом. Всі ці методи та засоби об'єднуються під терміном Business Intelligence або BI-рішення.

Один із найбільш перспективних напрямків інформаційних технологій, що застосовується в соціальних та економічних системах для підтримки процесу прийняття рішень, є інтелектуальний аналіз даних, також відомий як Data Mining або розкопка даних. Цей напрямок є міждисциплінарним і включає елементи штучного інтелекту (ШІ), математичної статистики та машинного навчання (МН), які застосовуються для розв'язання задач класифікації, кластеризації та асоціативного аналізу.

У контексті Data Mining (DM), варто відзначити, що він не надає готових шаблонів рішень або ж жорстких алгоритмів для виконання аналітичних завдань. Натомість, DM виступає як методологія організації аналітичної обробки даних, застосування прийомів та методів якої дозволяє видобути найбільш цінні знання з них. У центрі аналітичних технологій DM знаходяться методи машинного навчання (МН), які автоматично відновлюють структури, залежності та закономірності в даних, що інтерпретуються та осмислюються експертами або аналітиками, що дозволяє зробити висновки про особливості стану та розвитку явищ і процесів, а також надати рекомендації щодо більш ефективного управління ними.

Процес впровадження DM-технологій у практичну діяльність підприємств та організацій для розв'язання конкретних завдань підвищення ефективності управління у більшості випадків є складним і ресурсомістким.

Головними проблемами є відсутність чітко поставленого завдання та стратегії пошуку знань, евристичний характер більшості інтелектуальних моделей, висока розмірність і низька якість даних. Тому розробка нових підходів та методів реалізації DM-проектів для розв'язання конкретних завдань підвищення ефективності управління в соціальних та економічних системах є актуальним науково-технічним завданням.

У напрямку штучного інтелекту (ШІ), розвиток методів машинного навчання (МН) був великою мірою пов'язаний з дослідженнями вчених зарубіжних країн, таких як Б. Уідроу, М. Мінськ, П. Дж. Вербос, Дж. Хоп-філд, Д. Румельхарт, С. Пайперт та інших. Впродовж 70-80-х років XX століття в рамках машинного навчання були запропоновані важливі підходи, зокрема дерева рішень, розроблені Дж. Р. Куінленом і Л. Брейманом, асоціативні правила, запропоновані Р. Агравалом і Р. Шрикантом, самоорганізуючіся карти ознак, розроблені Т. Кохоненом та інші.

Формування і розвиток Data Mining (DM) як наукового напряму пов'язане з роботами таких вчених, як Г. П'ятецький-Шапіро, У. Файад, П. Сміт та інші, які внесли значний внесок у розуміння та розвиток методів аналізу даних.

Тема дослідження є актуальною з огляду на сучасний етап розвитку ринкових відносин, де виникає необхідність пошуку оптимальних управлінських рішень та постійного вдосконалення бізнес-процесів обслуговування.

Метою даної роботи є аналіз бізнес-процесів, пов'язаних з бізнесом обслуговування, як в кав’ярного так і ресторанного секторів, а також розробка програмного забезпечення, спрямованого на підвищення ефективності та збільшення обсягів продажу.

З метою досягнення зазначеної мети, було визначено наступні завдання:

– провести дослідження предметної області, щоб отримати глибоке розуміння контексту та особливостей кавового та ресторанного бізнесу;

– на основі аналізу отриманих даних сформулювати постановку завдання для майбутньої системи, враховуючи потреби та вимоги збільшення продажу та підвищення ефективності;

– розробити програмне забезпечення, що відповідає встановленим вимогам та спрямоване на покращення бізнес-процесів обслуговування;

– здійснити тестування розробленої системи для перевірки її функціональності, надійності та відповідності заданим критеріям.

штрішень для бізнесу обслуговування в контексті кавової та ресторанної галузей.

Предметом дослідження є методи та засоби управління бізнесом обслуговування, що включають стратегії, технології, процеси та інструменти, які сприяють ефективному управлінню та оптимізації бізнес-процесів в сфері обслуговування в кавових та ресторанних підприємствах.

Практична цінність. На основі проведених досліджень був розроблений продукт для прогнозування продажів у бізнесі обслуговування, зокрема в кавовому та ресторанному секторах. Основними принципами, що лежали в основі розробки даного продукту, були безкоштовність та його доступність для широкого кола користувачів, оскільки він може бути легко використаний на будь-яких операційних системах родини Windows.

1. ПРОЄКТНІ РІШЕННЯ ІНФОРМАЦІЙНІ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ РЕСТОРАННОГО БІЗНЕСУ

1.1 Аналіз предметної області

Ресторанний та кав'ярний бізнеси представляють собою важливі напрямки, що активно розвиваються в контексті суспільного прогресу. Промисловість громадського харчування тісно пов'язана з ритмом сучасного життя. Швидкість, з якою людина функціонує в сучасному світі, постійно прискорюється, що робить все важчим витрачати час на приготування їжі. У таких обставинах заклади громадського харчування, зокрема ресторани та кафе, можуть бути надзвичайно корисними.

Ресторан (від латинського "Restauro", що означає "відновлення" або "посилення") - це громадське харчування з широким асортиментом страв, що вимагають складного приготування, включаючи рекомендовані та фірмові.

Кафе (від французького "café", що буквально означає "кава") - це заклад громадського харчування та відпочинку, схожий на невеликий ресторан, але з обмеженим асортиментом продукції порівняно з рестораном, і можливо, з принципом самообслуговування.

Після ознайомлення з наведеними визначеннями можна зробити висновок, що ресторани та кафе не лише задовольняють фізіологічні потреби людей у харчуванні, але також є місцем для цікавого проведення вільного часу. Таким чином, можна зробити перший висновок, що якісно приготовлена їжа не є єдиним важливим елементом успішного ресторану або кафе [1].

Останнім часом спостерігається зростаюча популярність франчайзингу в галузі громадського харчування. Цей підхід передбачає відкриття закладів харчування на основі франчайзингових угод, що дозволяє отримати різноманітну підтримку з боку франчайзера та швидше залучити цільову аудиторію.

Згідно з визначенням Міжнародної Асоціації Франчайзингу, франчайзинг є системою постійних взаємовідносин між франчайзором і франчайзі, в результаті яких передаються знання, імідж, успіх, методи

виробництва та маркетингу франчайзі в обмін на взаємне задоволення інтересів. У сутності, це форма бізнес-організації, де потужна компанія (франчайзор) надає право іншій компанії (франчайзі) на продаж своїх товарів або послуг на основі певних умов та взаємовигідних угод.

Франчайзинг вперше почав активно використовуватися в США в середині XIX століття. Великий виробник швейних машин Зінгер розпочав масове виробництво, що дозволило його компанії торгувати за низькими цінами. Однак централізоване технічне обслуговування машин стало економічно неефективним. Це спонукало компанію розглянути франчайзингову систему: вони почали надавати фінансово незалежним фірмам виключні права на продаж та обслуговування швейних машин на конкретній території.

Проте, варто зазначити, що використання франчайзингу не завжди є перевагою, оскільки вимагає додаткового фінансування з боку франчайзі. У сфері ресторанного та кавового бізнесу така модель роботи менш поширена, порівняно з барами, закладами швидкого харчування та іншими. Це пояснюється тим, що багато засновників таких закладів прагнуть мати унікальний продукт, який суттєво відрізнятиметься від конкурентів. Відвідувачі ресторану готові платити за індивідуальність, за те, що їм не запропонують в інших закладах. Введення системи франчайзингу в ресторанному бізнесі може негативно вплинути на імідж компанії як унікального представника на своєму ринку.

Сучасна економічна ситуація відображає необхідність змін в поточному стані представників ресторанного та кавового бізнесу. Часто керівники таких організацій застосовують різні стратегії з метою зниження витрат. Зокрема, це може включати зменшення розміру порцій, виключення з меню страв з низькою націнкою або відмову від певних страв. Однак, окрім зниження витрат, такі зміни можуть викликати негативну реакцію серед відвідувачів, особливо постійних, які мають впевнені уявлення про розмір порцій та асортимент страв, які були раніше доступні.

В книзі "Маркетинг у ресторанному бізнесі" автори Патті Д. Шок та Джон Боуен наводять ряд критеріїв, які часто використовуються споживачами при виборі закладу громадського харчування:

– демографічні характеристики - кількість мешканців або приїжджаючих в цю місцевість (мікрорайон, який знаходиться в радіусі обслуговування закладу);

– середній рівень доходів у цьому населеному районі;

– розвиток або занепад даної місцевості, що впливає на інфраструктурне забезпечення (каналізація, дренаж тощо);

– зручність та доступність з точки зору транспортного сполучення та можливості паркування;

– видимість - наскільки легко помітити та відрізнити ресторан серед інших подібних закладів4

– привабливість - наскільки гостинною здаватиметься установа для перехожих та транзитних осіб;

– розташування - наскільки приємними здаються навколишні споруди [2].

Ця класифікація підтверджує, що в сфері обслуговуючого бізнесу вибір споживача залежить не лише від списку страв, що представлені в меню.

Сучасне суспільство все більше звертає увагу на концепцію ресторану, його унікальність та спрямованість, яка відображається в усіх ключових елементах закладу, від меню до зовнішнього оформлення.

Іншим трендом, що актуальний в наш час, є зростаючий інтерес до здорового харчування. Люди відмовляються від відвідування ресторанів та кафе, оскільки розуміють, що там їх не задовольнять продукти, що відповідають концепції здорового харчування. Цей тренд спричинив появу ряду закладів, спеціалізованих у здоровому харчуванні. В сфері швидкого харчування також починають з'являтися подібні організації.

Зазначені факти дозволяють зробити висновок про необхідність

постійного моніторингу тенденцій ринку для керівників ресторанного та кавового бізнесу. Це обумовлено тим, що такі тенденції можуть допомогти в створенні власної концепції закладу.

Ресторанний бізнес задовольняє кілька потреб людини одночасно: потребу в харчуванні, соціалізації та проведенні дозвілля. Тому засновникам ресторанів необхідно пропонувати рішення, що виходять за межі лише галузі громадського харчування.

Ресторани та кафе можуть бути різних видів, але основним критерієм для їхньої класифікації є їхня унікальність, яка формує образ закладу в свідомості споживача.

У нестабільній економічній ситуації власники ресторанів та кафе часто намагаються зекономити на якості обслуговування (зменшення порцій, заміна на більш економні продукти), проте такі зміни негативно впливають на сприйняття закладу потенційними споживачами.

Франчайзингова система активно використовується в галузі громадського харчування, але в ресторанному та кавовому бізнесі її застосування обмежене через його потребу у збереженні унікальності та індивідуальності закладу.

Враховуючи наведені тенденції, можна зробити висновок, що існування ресторанів та кавових закладів на сучасному ринку є складним процесом, що вимагає особливого підходу. Умови кризи можуть бути подолані лише за умови розробки креативної стратегії, яка відзначатиме особливості та виокремлюватиме заклад серед конкурентів. Вибір креативної стратегії буде визначати рекламну кампанію та просування. Проте важливо, щоб креативна стратегія відображалась не лише у рекламі, а й в самій атмосфері закладу.

Особливість закладу виявляється через його концепцію, що впливає на його позиціонування та сприйняття потенційними споживачами.

Для успішної реалізації обраної стратегії в ресторанному та кавовому бізнесі було встановлено пріоритети його розвитку з урахуванням диференціації ресторанного продукту. Це охоплює включення нових кухонь у

меню ресторану (арабської, грецької, турецької), відродження традиційної купецької кухні, розвиток "дитячої тематики в концепції ресторану", спеціалізацію, використання нетрадиційних методів готування страв, зокрема розвиток молекулярної кулінарії та фьюжн-напрямків.

Забезпечення ефективної реалізації стратегії розвитку ресторанного та кавового бізнесу у сервісній економіці міст регіону потребує врахування стратегічної взаємодії між кореневими компетенціями підприємств-ресторанів (їх здатність створювати особливу цінність для споживача, складність відтворення конкурентами та використання їх в різних ринкових контекстах) та ключовими компетенціями ринку. Представлено на рису 1.1.

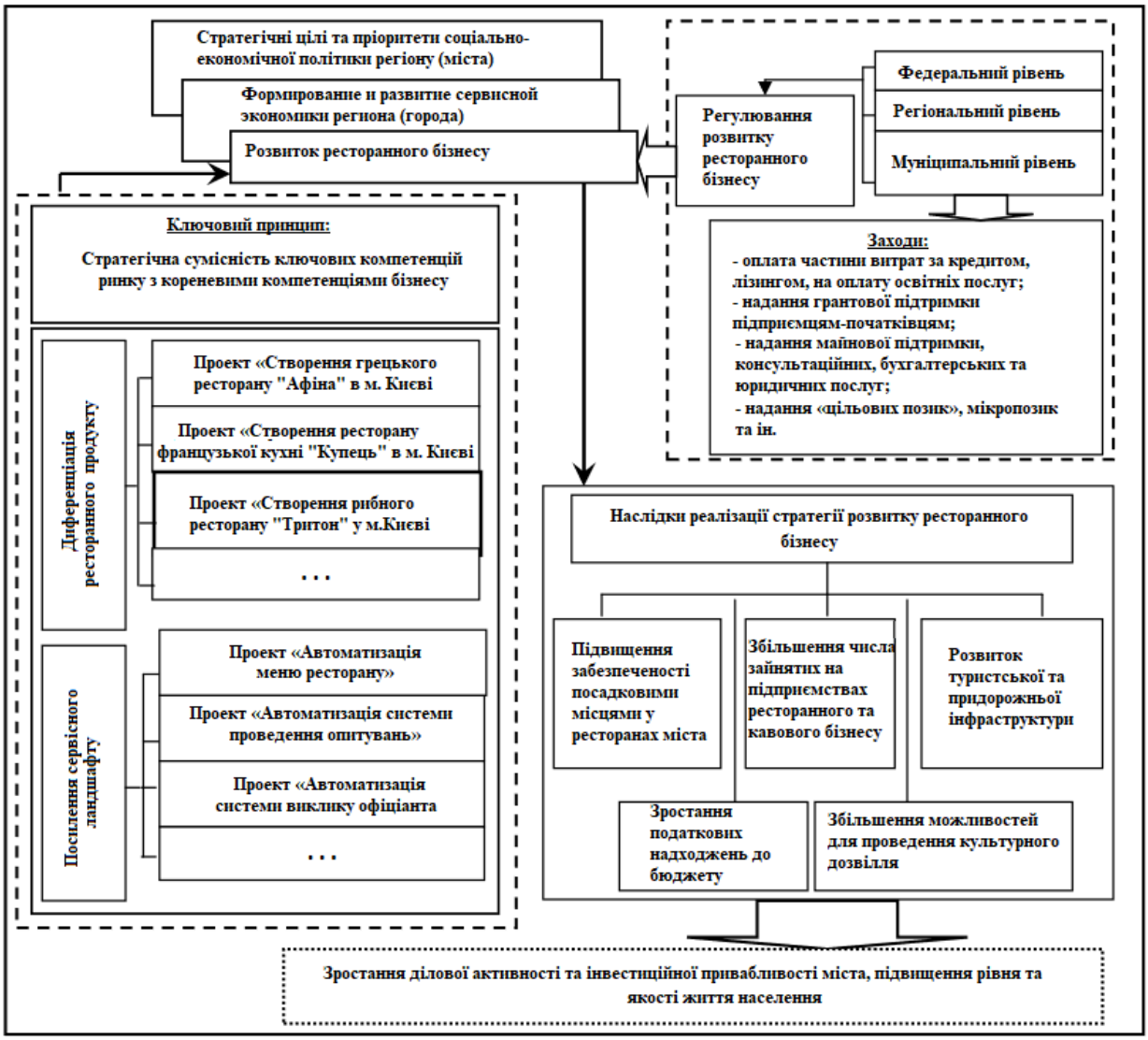


Рисунок 1.1 – Умови ефективної реалізації стратегії розвитку ресторанного бізнесу обслуговування

У сучасних умовах все більшої ваги набуває проектно-цільовий підхід разом із програмно-цільовими методами управління територіальним розвитком. У практичній діяльності проекти об'єднуються в портфель, який для розвитку ресторанного та кавового бізнесу можна розподілити на дві напрямки: проекти диференціації ресторанного продукту та проекти зміцнення сервісного ландшафту, спрямовані на розвиток культури ресторанного обслуговування та інноваційних методів обслуговування клієнтів.

На основі експертно-аналітичної оцінки встановлено, що реалізація комбінованої стратегії розвитку ресторанного та кавового бізнесу з використанням проектного підходу дозволить збільшити частку обороту підприємств даної сфери та досягти ряду позитивних соціально-економічних результатів [3].

1.1.1 Опис процесу діяльності бізнесу обслуговування

Предметним середовищем дослідження є управлінська діяльність персоналу в сфері обслуговування, яка пов'язана з аналізом попиту на асортимент продукції в цьому бізнесі. Основна мета цього дослідження полягає в підвищенні ефективності та рентабельності бізнесу, а також у зменшенні ризиків, з якими цей бізнес може зіткнутися.

На рис. 1.2 у формі діаграми станів представлені всі основні стани системи, які відображають різні аспекти функціонування бізнесу.

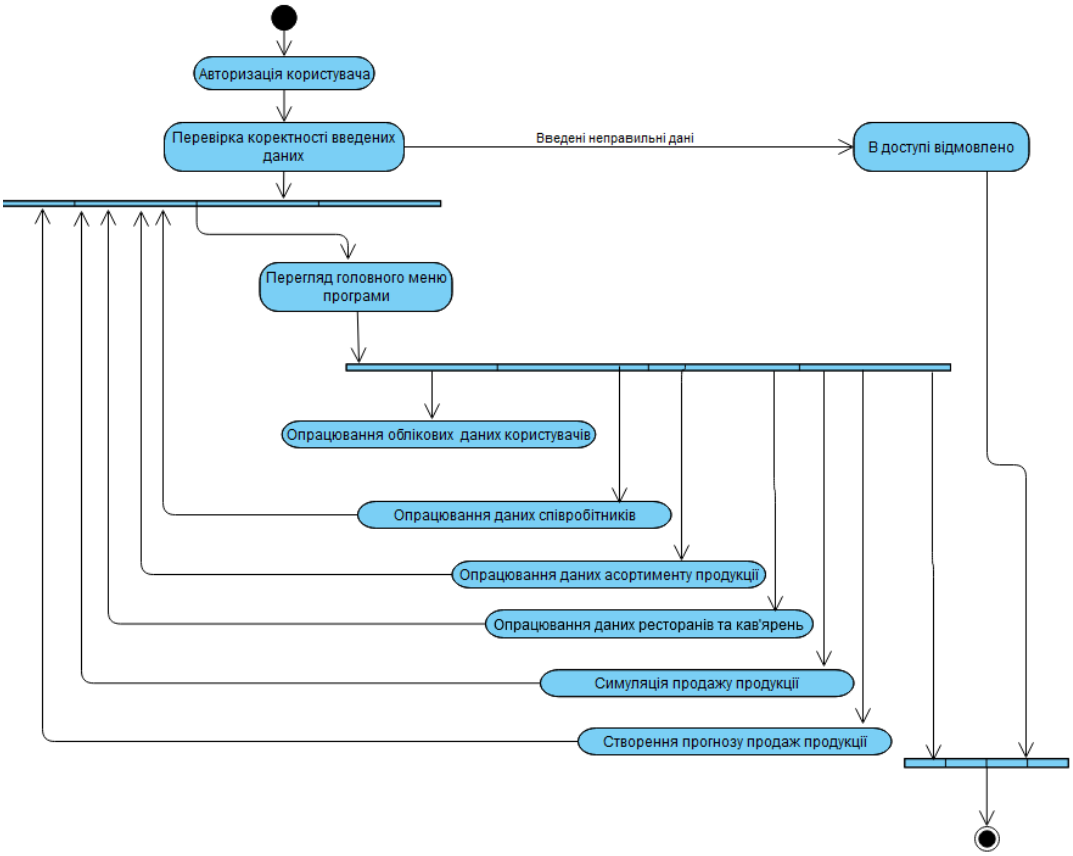


Рисунок 1.2 – Діаграма діяльності станів системи

У разі успішної авторизації користувача відображається головне вікно системи, яке містить меню з усіма доступними діями для користувача. Залежно від ролі користувача, деякі пункти меню можуть бути недоступні.

Адміністратор системи має повний доступ до всіх функцій, зокрема:

– додавання, редагування та видалення облікових даних користувачів;

– додавання, редагування та видалення співробітників ресторанів та кав'ярень;

– додавання, редагування та видалення ресторанів та кав'ярень;

– додавання, редагування та видалення продукції;

– здійснення симуляції продажу для обраного ресторану або кав'ярні;

– проведення навчання математичної моделі;

– створення прогнозів щодо майбутнього попиту на продукцію.

1.1.2 Актори та бізнес-процеси ресторанного бізнесу

Перед розробкою інформаційної системи підтримки прийняття рішень для бізнесу обслуговування необхідно провести аналіз акторів, що взаємодіють з системою, та визначити їхні ролі у цьому контексті.

Прийнято рішення створити дві основні ролі: адміністратора системи та менеджера. Адміністратор системи відповідає за налаштування та керування системою, в той час як менеджер використовує систему для прийняття рішень та управління бізнесом обслуговування.

Для кращого розуміння функціональності розробленої системи рекомендується створення діаграми прецедентів (use-case). Діаграма прецедентів у мові моделювання UML відображає відносини між акторами (користувачами системи) та прецедентами (можливостями, функціональними вимогами), і є важливою складовою моделі прецедентів, яка дозволяє концептуально описати систему [4].

Прецедент представляє собою конкретну можливість або функціональну частину системи, завдяки якій користувач може отримати певний, вимірний та необхідний йому результат. Кожен прецедент описує окремий сервіс або функціональний вимір системи, визначає один з варіантів використання та типовий спосіб взаємодії користувача з системою. Варіанти використання використовуються для специфікації зовнішніх вимог до системи.

Таблиця 1.1 – Функціональні вимоги до додатку

Вимоги	**Опис**
REQ-1	Система має надавати можливість користувачам додавати та редагувати інформацію про облікові записи користувачів.
REQ-2	Система має надавати можливість користувачам додавати та редагувати інформацію про продукцію.
REQ-3	Система має надавати можливість користувачам додавати та редагувати інформацію про співробітників.
REQ-4	Система має надавати можливість користувачам додавати та

	редагувати інформацію про ресторани та кав'ярні.
REQ-5	Система має надавати можливість користувачам здійснювати симуляцію продажу продукції.
REQ-6	Система має надавати можливість користувачам проводити навчання нейронної мережі на основі даних продажу продукції.
REQ-7	Система має надавати можливість користувачам здійснювати прогнозування продажу продукції.
REQ-8	Система має надавати можливість користувачам додавати та редагувати інформацію про облікові записи користувачів.
REQ-9	Система має надавати можливість користувачам додавати та редагувати інформацію про продукцію.

Таблиця 1.2 – Нефункціональні вимоги до додатку

Вимоги	**Опис**
REQ-10	Додаток повинен працювати без перебоїв та збоїв, забезпечуючи стабільну роботу системи.
REQ-11	Додаток повинен мати механізми захисту даних користувачів, забезпечувати конфіденційність та захищати від несанкціонованого доступу.
REQ-12	Додаток повинен працювати швидко та ефективно, забезпечуючи мінімальні часи завантаження та відгуку на запити користувачів.
REQ-13	Додаток повинен мати інтуїтивно зрозумілий та зручний для використання інтерфейс, що сприяє зручності та задоволенню користувачів.
REQ-14	Додаток повинен забезпечувати швидку обробку та відображення даних, що дозволяє користувачам ефективно взаємодіяти з системою.

Таблиця 1.3 – Актори та цілі додатку (користувач із правами адміністратора)

Актори	**Цілі**
Адміністратор	Метою адміністратора є керування обліковими записами, включаючи створення, редагування та видалення облікових даних користувачів.
Менеджер	Метою менеджера є ефективна робота з системою. Це включає в себе використання функціональності додатку для управління ресторанами, кав'ярнями, продукцією.
База даних	Метою бази даних є зберігання інформації, яка використовується в системі. База даних забезпечує надійне збереження даних користувачів та дозволяє ефективну обробку даних у системі.

Таблиця 1.4 – Опис варіантів використання додатку

Варіант використання	**Ім’я**	**Опис**
UC1	Ідентифікація в системі	Надає можливість пройти авторизацію в системі
UC2	Вивід каталогу користувачів	Надає можливість для адміністратора вивести каталог всіх користувачів системи
UC3	Додати користувача	Надає можливість додати нового користувача системи
UC4	Редагувати користувача	Надає можливість редагувати інформацію вибраного із списку користувача
UC5	Видалити користувача	Надає можливість видалити вибраного із списку користувача
UC6	Вивід каталогу продукції	Надає можливість виведення каталогу всієї продукції

UC7	Додати продукцію	Надає можливість додати нову продукцію
UC8	Редагувати продукцію	Надає можливість редагувати інформацію вибраної продукції
UC9	Видалити продукцію	Надає можливість видалити інформацію про вибрану продукцію
UC10	Вивід каталогу співробітників	Надає можливість вивести каталог зареєстрованих у системі співробітників
UC11	Додати співробітника	Надає можливість додати інформацію про нового співробітника
UC12	Редагувати співробітника	Надає можливість і редагувати інформацію вибраного співробітника
UC13	Видалити співробітника	Дозволяє користувачеві видалити вибраного співробітника
UC14	Вивід каталогу ресторанів та кав'ярень	Надає можливість вивести каталог всіх ресторанів та кав'ярень
UC15	Додати ресторан або кав'ярню	Надає можливість додавання інформації про ресторан чи кав'ярню
UC16	Редагувати ресторан або кав'ярню	Надає можливість редагування інформації ресторану або кав'ярні
UC17	Видалити ресторан або кав'ярню	Надає можливість видалити ресторан або кав'ярню
UC18	Симуляція продаж продукції	Надає можливість здійснити симуляцію продажі продукції бізнесу за вибраний період часу
UC19	Прогнозування попиту на продукцію	Надає можливість здійснювати прогнозування попиту на продукцію

UC20	Вивід системних подій	Надає можливість виведення подій, що відбулись в системі

На підставі накопиченої інформації за допомогою візуального інструменту «Paradigm» були створені діаграми варіантів використання (use-case diagrams) для ролей "адміністратор" та "користувач". На рисунку 1.3 наведено діаграму варіантів використання для ролі "адміністратор", а на рисунку 1.4 - для ролі "користувач".

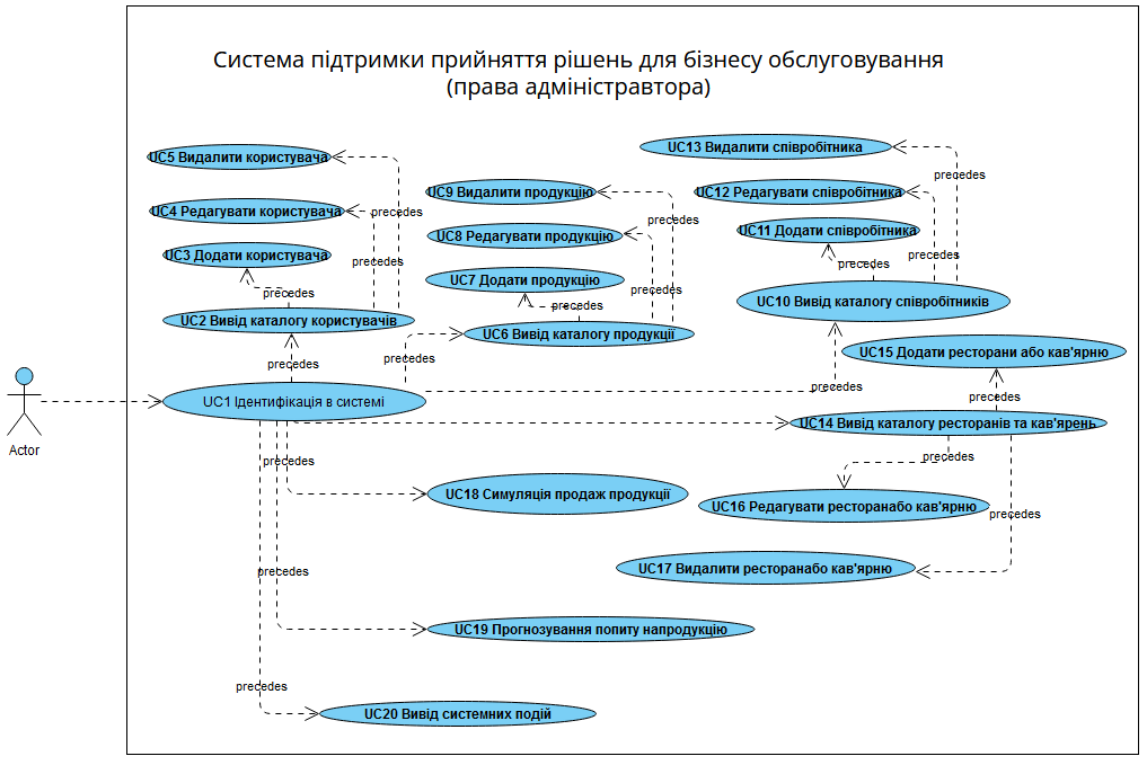


Рисунок 1.3 – Діаграма use-case для ролі «адміністратор»

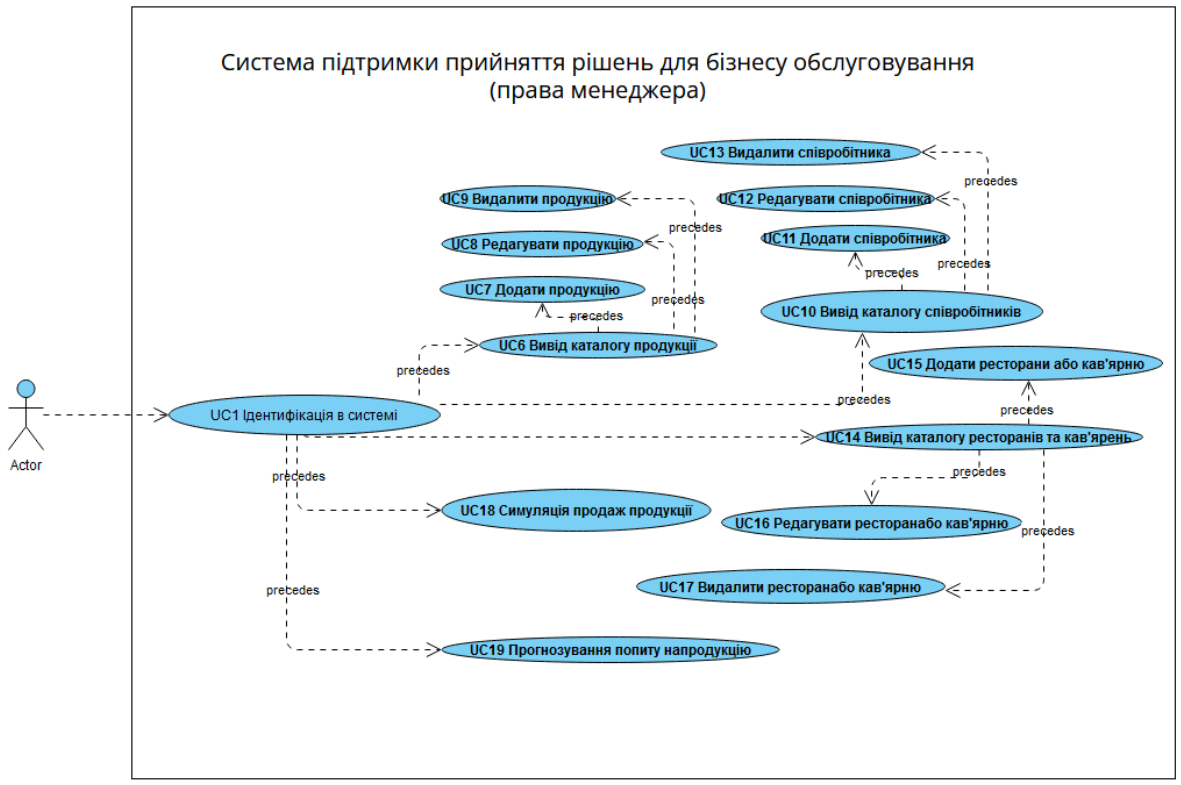


Рисунок 1.4 – Діаграма use-case для ролі «користувача»

Менеджер виконує обробку даних, пов'язаних зі співробітниками, продукцією, ресторанами та кав'ярнями. Він займається симуляцією продажу у ресторанах та кав'ярнях, а також здійснює прогнозування попиту на продукцію у цих закладах.

Адміністратор, на відміну від менеджера, має додаткові функції. Крім роботи з даними співробітників, продукції, ресторанів та кав'ярень, він відповідає за реєстрацію користувачів у системі. Крім того, адміністратор надає користувачам різні рівні доступу відповідно до їх ролей і повноважень.

1.2 Функціональна структура бізнесу обслуговування

Рисунок 1.5 демонструє діаграму декомпозиції моделі прогнозування попиту на продукцію, представлену в нотації IDEF0. Ця діаграма відображає структуру задач та взаємозв'язки між елементами системи та її зовнішнім середовищем [5].

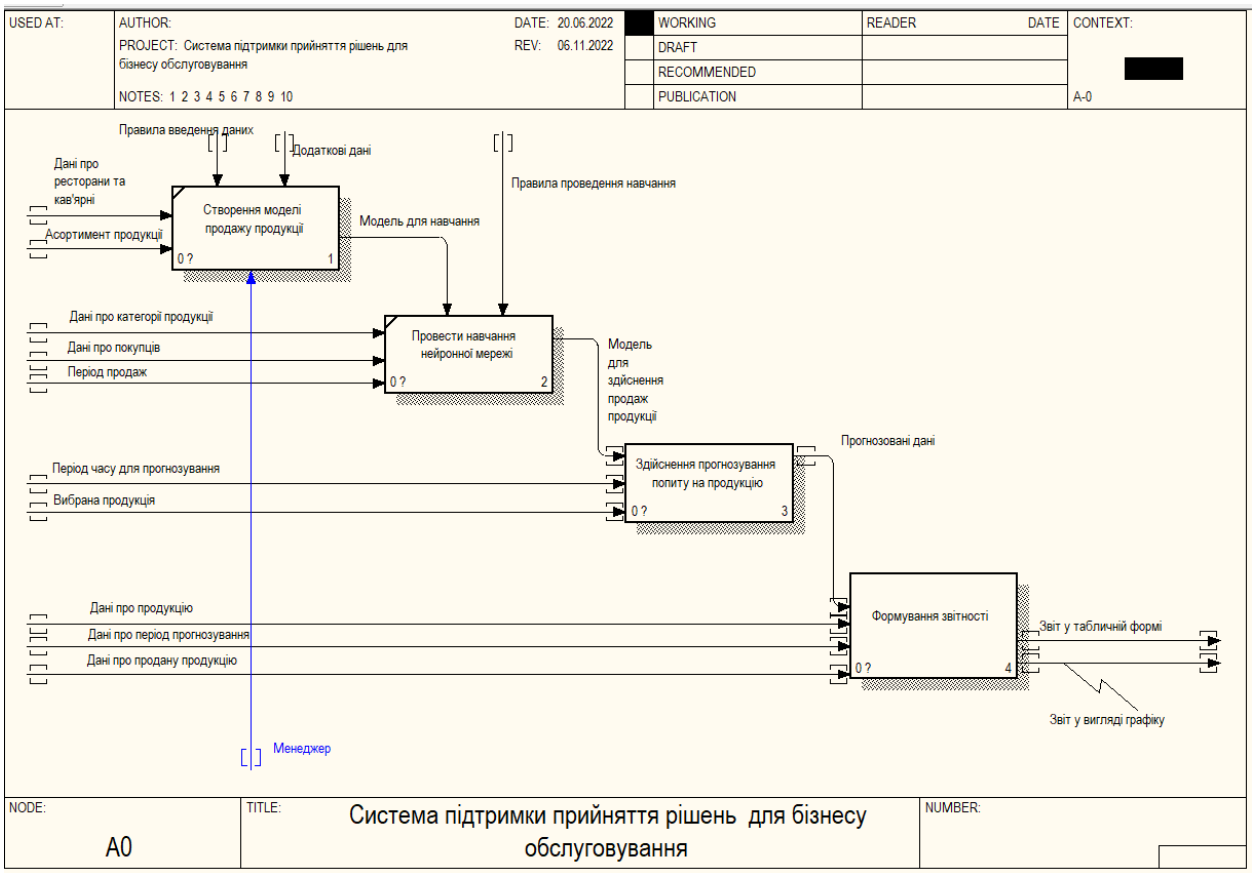


Рисунок 1.5 – Діаграма декомпозиції моделі для прогнозування попиту на продукцію бізнесу обслуговування

1.3 Постановка задачі

Мета створення системи полягає в підвищенні ефективності процесу прогнозування для бізнесу обслуговування (кавового, ресторанного). Головним призначенням системи є моделювання попиту клієнтів на продукцію у ресторанах та кав'ярнях протягом обраних періодів за допомогою нейронної мережі.

Розробка програмного забезпечення передбачає наступні кроки:

1. Розробка структури бази даних.
2. Реалізація програмного додатка, який включає три основних модуля:

– модуль для введення та редагування інформації в базі даних;

– модуль авторизації користувачів системи;

– модуль для прогнозування попиту на продукцію, що пропонується в кав'ярнях та ресторанах.

Реалізація програмного додатка з використанням технологій .NET та мови програмування C#. Результатом роботи є Windows додаток.

Розроблена система повинна мати такі особливості та можливості:

Сутності:

– користувач (інформація про користувачів): прізвище, ім’я, логін, пароль, загальна інформація, ідентифікатор ролі;

– заклад (інформація про кав'ярні та ресторани): назва закладу та опис;

– категорія продукції (інформація про категорію продукції): назва та опис;

– співробітник (інформація про співробітників): ім’я, прізвище, номер телефону, адреса, електронна адреса та посада;

– продукція (інформація про продукцію): назва продукції, ідентифікатор категорії, одиниці виміру, ціна та опис;

– продаж продукції (інформація про продажі продукції): ідентифікатор продукції, дата продажу, кількість, сума продажі;

– логи (інформація про активність користувачів системи) ідентифікатор користувача, подія в системі, дата/час події.

Введення:

– вводиться та редагуються: користувачі системи, заклади бізнесу обслуговування, категорії продукції, продукція та співробітники.

– фіксуються замовлення клієнтів на продукцію у звкладах.

– фіксуються події в системі.

Пошук: замовленої продукції у закладах бізнесу.

Прогнозування: на підставі наявної інформації про продану продукцію, система здійснює прогнозування обсягу продажу продукції в різних закладах. Це дозволяє бізнесу обслуговування (кавовим та ресторанним закладам) зробити обґрунтовані рішення щодо планування виробництва, закупівлі і запасів, а також оптимізації процесів управління та прогнозування попиту на

продукцію.

1.4 Висновок

У процесі створення системи підтримки прийняття рішень (СППР) необхідно провести аналіз даних та бізнес-процесів замовника, а також розробити структури сховища, враховуючи його потреби і технологічні процеси. На сучасному ринку існує багато компаній, які пропонують різноманітні продукти для розв'язання завдань, що виникають під час проектування та експлуатації СППР. Ці продукти включають системи керування базами даних (СУБД), інструменти для завантаження/трансформації/вивантаження даних, аналітичні інструменти для OLAP-аналізу та багато іншого. Аналіз ринку та вивчення цих засобів вимагає значних зусиль і часу з боку фахівців.

Ураховуючи обмежені фінансові та інші ресурси, а також складність та багатоетапність проектів побудови систем підтримки прийняття рішень (СППР), очевидно, висока вартість помилок у проектуванні. Помилки при виборі програмного забезпечення можуть призвести до фінансових витрат та збільшення тривалості проекту. Помилки у проектуванні структури даних можуть призвести до неприйнятних бізнес-рішень та вимагати велику кількість часу на перезавантаження даних, що може займати кілька днів.

У цьому розділі були визначені основні бізнес-процеси, які підлягають автоматизації в предметному середовищі. Була проведена аналітична робота з вивчення діяльності користувачів системи, а також були побудовані діаграми прецедентів для опису їх взаємодії з системою. Встановлено мету та поставлені завдання, які необхідно вирішити для досягнення цілей проекту. Крім того, була визначена вихідна інформація, яка надходить до програми, а також звітна інформація, яку програма генерує в результаті своєї роботи. На основі проведеного аналізу була спроектована ER-діаграма бази даних проекту, що відображає структуру та зв'язки між сутностями системи.

2 МОДЕЛІ ТА МЕТОДИ ПРОГНОЗУВАННЯ ПОПИТУ НА ПРОДАЖ ПРОДУКЦІЇ РЕСТОРАННОГО БІЗНЕСУ

2.1 Суть задачі прогнозування попиту

Прогнозування попиту - це науковий прогноз загальної структури попиту та попиту споживачів на товари та послуги. Вона заснована на дослідженнях, причинах, відносинах, тенденціях та науково обґрунтованих моделях прогнозування ринку.

У будь-якій ринковій системі потреби користувачів товарів та послуг зосереджені на цілі та рушійній силі економічного розвитку. Таким чином, проблема дослідження та прогнозування попиту є першим та найважливішим інструментом для організації маркетингової діяльності всередині компанії, а також для регулювання ринку та товарів, які впливають на економіку держави в цілому.

Прогнозування попиту є науковим процесом, спрямованим на передбачення загальної структури попиту та споживацьких потреб у товарах і послугах. Цей процес базується на проведенні досліджень, аналізі причин, вивченні взаємозв'язків, аналізі тенденцій та використанні науково обґрунтованих моделей для прогнозування ринкових умов.

У будь-якій ринковій системі задоволення потреб користувачів товарів та послуг є ключовою метою та джерелом економічного розвитку. Тому проблема дослідження та прогнозування попиту стає першочерговим та найважливішим інструментом для організації маркетингової діяльності в межах компанії, а також для регулювання ринку і товарів, які мають вплив на економіку держави в цілому.

Прогнозування характеризується наступними особливостями:

– прогноз завжди має певну ступінь помилки, оскільки передбачити майбутні події і точно їх оцінити неможливо;

– з плином часу і оцінкою власних помилок прогнози стають все точнішими і наближаються до реальних значень;

– збільшення обсягу даних та врахування більш широкого спектру чинників призводить до більш точних прогнозів;

– прогнози є більш точними на короткостроковий період, оскільки на довгострокову перспективу вплив багатьох факторів може значно змінитися;

– прогнозування не замінює необхідність робити розрахунки. Виробничі товари, пов'язані з попитом, необхідно розраховувати на основі прогнозування потреби в кінцевому продукті, який можна передбачити.

Прогнозування споживчого попиту є необхідним для розробки стратегій розвитку підприємств у виробничій та торговій сферах. Ефективне прогнозування попиту, як і будь-який інший бізнес-процес, складається з трьох взаємозв'язаних елементів: людей, процесу та інструментів. Попит на прогнозування різних послуг та товарів є дуже специфічним, і, отже, виникає питання, які саме фактори впливають на формування попиту [6].

Під час розробки процесу прогнозування попиту необхідно враховувати наступні фактори:

– як організовані функції маркетингу та продажу в компанії? Важливо визначити, які основні стратегії маркетингу та методи продажу застосовуються, оскільки це може вплинути на попит на товари та послуги;

– хто має можливість впливати на попит в компанії? Ідентифікація осіб або груп, які мають вплив на прийняття рішень щодо попиту, допоможе врахувати їхні впливові фактори при прогнозуванні;

– де знаходиться необхідна для прогнозування інформація? Важливо визначити джерела інформації, яка є необхідною для виконання прогнозування, та організувати її збір та зберігання з урахуванням принципів регулярності та точності.

Ефективний прогноз починається з поліпшення якості вихідної інформації. Організація процесу збору вихідних даних повинна враховувати принципи регулярності та точності, щоб забезпечити надійність та достовірність даних.

Існує багато методів прогнозування попиту, проте для зручності їх можна умовно поділити на дві групи: експертні і статистичні. Використання експертних методів передбачає залучення фахівців та їхніх знань та досвіду для формування прогнозу. Статистичні методи, натомість, базуються на аналізі статистичних даних та використанні математичних моделей для прогнозування [7].

Перший заснований на експертній оцінці та має суб'єктивний характер. Суть у тому, щоб переводити різні експертні думки до формул, які формують прогнози. Експертні методи включають: методи введення в експлуатацію, мозковий штурм, анкетування.

Статистичні методи включають використання статистичних розрахунків у тому, щоб мати картину майбутнього з урахуванням минулого, типовим прикладом є методика розрахунку середнього значення.

Перший тип методів, що базуються на експертній оцінці, має суб'єктивний характер. Ці методи спрямовані на перетворення різних експертних думок в формули, які використовуються для формування прогнозів. Експертні методи включають такі підходи, як методи введення в експлуатацію, мозковий штурм та анкетування.

Статистичні методи, у свою чергу, ґрунтуються на використанні статистичних розрахунків для отримання прогнозу, зважаючи на минулі дані. Один з типових прикладів статистичних методів - це розрахунок середнього значення, який дозволяє отримати уявлення про майбутній стан на основі попередніх даних.

Прогнозування відображає оцінку майбутнього значення змінної або набору змінних. В бізнес-контексті основною метою прогнозування є надання інформації для процесу планування. Прогнозування дозволяє нам приблизно передбачити майбутнє, що в свою чергу допомагає нам прийняти кращі рішення та зайняти більш вигідне становище у майбутньому.

Система прогнозування попиту впроваджує прогнози в плани продажу, враховуючи різноманітні обмеження, що стають основою для фінансового планування, маркетингових стратегій та управління закупівлями товарів. Прогнозування попиту має вирішальне значення для досягнення позитивних фінансових результатів. Якісне прогнозування попиту вимагає наявності в компанії високопрофесійного маркетингового відділу або наявності кваліфікованих маркетологів [8].

2.2 Аналітичний огляд методів розв'язання задачі прогнозування

При плануванні архітектури мережі звичайно випробовуються різні конфігурації з різною кількістю елементів. Оскільки проблема прогнозування є підтипом задачі регресії, можна використовувати наступні типи нейронних мереж: багатошаровий персептрон (MLP), радіально-базисна мережа (RBN), узагальнено-регресійна мережа (GRNN), мережа Воль Ельмана. В даному випадку будемо розглядати три з них - MLP, RBN і GRNN - для подальшого порівняння [9].

2.2.1 Багатошарові персептрони

Багатошарові перцептрони виявляються більш ефективними у вирішенні завдань, подібних до тих, які вирішують одношарові перцептрони, але вони мають значно більшу обчислювальну потужність. Ця властивість дозволяє їм більш точно моделювати багатовимірні залежності з великим ступенем нелінійності, а також враховувати високий рівень перехресного та групового впливу вхідних змінних на вихідні значення.

У разі необхідності застосування мережі складнішої структури до багатошарового перцептрона можна додати додаткові приховані шари або збільшити кількість нейронів у наявних прихованих шарах (див. рис. 2.1).

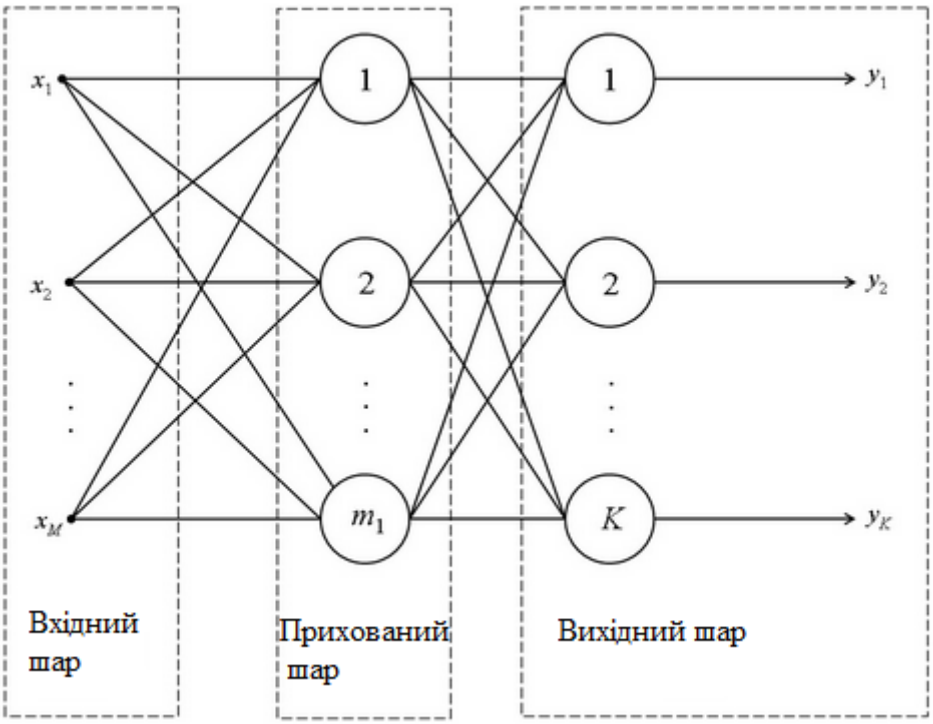


Рисунок 2.1 – Багатошарова штучна нейронна мережа

Для обчислення кількості вагових коефіцієнтів в багатошаровому перцептроні з L прихованими шарами, кожен з яких містить m_l нейронів, можна скористатися наступною формулою:

$$N_w = (M+1)m_l + \sum_{l=2}^{L}(m_{l-1}+1)m_l + (m_L+1)K \qquad (2.1)$$

Кожному нейрону мережі, крім синаптичних зв'язків з елементами вхідного вектора, надається зв'язок з фіктивним одиничним входом, який використовується для керування коефіцієнтом зміщення.

При виборі структури багатошарового перцептрона для більшості завдань, використовується правило "Правила 2–5". Згідно з цим правилом, кількість налаштовуваних вагових коефіцієнтів повинна бути в 2–5 разів меншою, ніж кількість прикладів у навчальній вибірці. Якщо це співвідношення менше 2, мережа втрачає здатність до узагальнення інформації з навчальних прикладів. Коли це співвідношення стає 1 або менше, мережа просто запам'ятовує відповіді для кожного навчального прикладу, не здатна до коректного відгуку на нові випадки. З іншого боку, якщо кількість навчальних прикладів надто велика для обраної структури мережі, модель мережі може усереднювати вихідні значення для різних комбінацій вхідних

векторів, що призводить до втрати точності відповіді для окремих прикладів та збільшення максимальної вибіркової помилки.

При виборі структури багатошарового перцептрону необхідно враховувати, що кількість нейронів у прихованому шарі, що передує вихідному шару, повинна бути не меншою, ніж кількість виходів.

Метод Уідроу-Хоффа, який був розглянутий раніше, не підходить для налаштування вагових коефіцієнтів багатошарових перцептронів. Це зумовлено тим, що цей метод дозволяє змінювати ваги тільки для нейронів вихідного шару, не враховуючи синаптичні коефіцієнти прихованих нейронів, які також потребують коригування.

Завдання навчання багатошарових перцептронів може бути сформульоване як задача оптимізації, в якій цільовою функцією є загальна помилка, що обчислюється за допомогою навчальної вибірки. Таким чином, цю задачу можна розв'язати як задачу багатовимірної оптимізації з використанням методів детермінового, градієнтного або стохастичного пошуку.

2.2.2 Узагальнено регресійні нейронні мережі

Узагальнено-регресійна нейронна мережа (General Regression Neural Network, GRNN) є підвидом радіально-базових мереж, призначених для вирішення задач регресії з використанням ядерної апроксимації. Узагальнено-регресійна штучна нейронна мережа не потребує визначення функціональної залежності між вхідними та вихідними змінними, відмінно від класичного регресійного аналізу.

У GRNN всі вхідні образи розбиваються на кластери. Навіть якщо кількість образів є невеликою, кожен образ може утворювати свій кластер. Структура GRNN складається з чотирьох шарів: вхідних модулів, які мають повний зв'язок зі шаром образів; шару образів; шару підсумовування; вихідного шару. Функція активації використовується у вигляді Гаусової функції.

GRNN навчається за допомогою учителя, тобто навчальний процес включає надання пар вхідних-вихідних даних для тренування мережі [10].

На рисунку 2.2 представлена архітектура нейронної мережі.

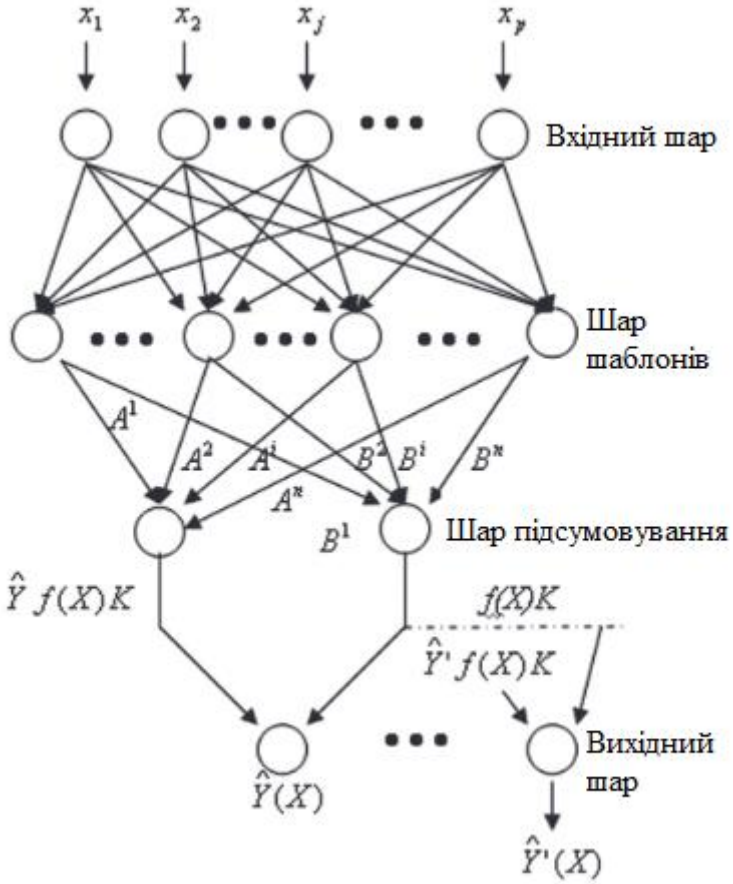


Рисунок 2.2 – Архітектура узагальнено-регресійної нейронної мережі

На вхідний шар нейронної мережі подаються навчальні дані. Кожному вхідному нейрону встановлюються відповідні вагові коефіцієнти, які відображають значення навчальної вибірки. Потім ці дані переходять до шару образів, де вагові коефіцієнти піддаються обробці за допомогою функції активації, представленої такою формулою:

$$\sum_{i=1}^{n} \exp\left(-\frac{D_i^2}{2\sigma^2}\right) \qquad (2.2)$$

де D_i^2 – квадрат Евклідової відстані;

$$D_i^2 = (X - X^i)^T (X - X^i) \qquad (2.3)$$

σ — параметр згладжування, який використовується для досягнення оптимальних результатів мережі, зазвичай обирається в діапазоні від 2 до 6. Перетворені значення, отримані на попередньому етапі, подаються на шар

підсумовування. Цей шар складається з двох нейронів, з яких S-нейрон накопичує суму виходів шару образів, зважених відповідними ваговими коефіцієнтами.

$$\sum_{i=1}^{n} Y^i \exp\left(-\frac{D_i^2}{2\sigma^2}\right) \qquad (2.4)$$

D – нейрон виконує обчислення, яке полягає в сумуванні незважених виходів шару образів. Він приймає вхідні сигнали з шару образів, де вагові коефіцієнти не застосовуються, і обчислює їх суму.

$$\sum_{i=1}^{n} \exp\left(-\frac{D_i^2}{2\sigma^2}\right) \qquad (2.5)$$

У вихідному шарі обчислюється виважене середнє значення за допомогою наступної формули:

$$\hat{Y}(X) = \frac{\sum_{i=1}^{n} Y^i \exp\left(-\frac{D_i^2}{2\sigma^2}\right)}{\sum_{i=1}^{n} \exp\left(-\frac{D_i^2}{2\sigma^2}\right)} \qquad (2.6)$$

2.2.3 Радіально-базисні мережі

Мережі з радіальною базисною функцією (РБФ-мережі) є особливим сімейством нейронних мереж, у яких приховані нейрони виконують функції, що радіально змінюються навколо вибраного центру. Вони приймають значення, відмінні від нуля, тільки в околицях цього центру. Подібні функції, що визначаються у вигляді $\varphi(x) = \varphi(\|x - c\|)$ називаються радіальними базисними функціями.

У мережах радіального типу, приховані нейрони відіграють роль у відображенні радіального простору навколо окремо заданої точки або групи точок, які утворюють кластер. Суперпозиція сигналів від усіх прихованих нейронів, що здійснюється вихідним нейроном, дозволяє отримати відображення багатовимірного простору.

Мережі радіального типу є природним доповненням сигмоїдальних мереж. Типова структура радіальної мережі включає вхідний шар, до якого надходять сигнали, що описуються вхідним вектором x, прихований шар з

нейронами радіального типу і вихідний шар, що зазвичай складається з одного або кількох лінійних нейронів. Функція вихідного нейрона обмежується зваженим підсумовуванням сигналів, що генеруються прихованими нейронами.

Враховуючи практичні обмеження, використання p базисних функцій у розкладанні є неприйнятним, оскільки кількість навчальних вибірок може бути великою, що призводить до високої обчислювальної складності навчального алгоритму. Тому шукається субоптимальне рішення у просторі меншої розмірності, яке достатньо точно апроксимує точне рішення [11].

Якщо обмежитися K базисними функціями, то апроксимацію рішення можна представити у вигляді:

$$F(x) = \sum_{i=1}^{K} w_i \cdot \varphi(\|x - c_i\|), \qquad (2.7)$$

де $K < p$, а c_i (i = 1, 2, …, K) є множиною центрів, що потребують визначення.. Якщо прийняти $K = p$, можна отримати точне рішення $c_i = x_i$.

Задача апроксимації радіальною базисною мережею полягає в підборі відповідної кількості радіальних функцій та їх параметрів, а також в такому підборі ваг (i = 1, 2, ..., K), щоб рішення рівняння (2.7) було найближчим до точного рішення.

Проблема визначення параметрів радіальних функцій та ваг мережі може бути сформульована як задача мінімізації цільової функції, яка може бути записана у наступній формі:

$$E = \sum_{i=1}^{p}\left[\sum_{j=1}^{K} w_j \cdot \varphi(\|x_i - c_i\|) - t_1\right]^2 \qquad (2.8)$$

У цьому рівнянні K представляє кількість радіальних нейронів, а p – кількість пар (x, t), де x – це вхідний вектор, а t – відповідна йому очікувана величина.

Найчастіше як радіальна функція застосовується функція Гауса. При розміщенні її центру в точці сі вона може бути визначена як

$$\varphi(x) = \varphi(\|x - c_i\|) = exp\left[-\frac{\|x-c_i\|^2}{2\sigma_i^2}\right] \qquad (2.9)$$

У цьому виразі σ_i– параметр, від якого залежить ширина розмаху

функції.

Архітектура радіальних мереж має структуру, подібну до багатошарової структури сигмоїдальних мереж з одним прихованим шаром. (рис. 2.3).

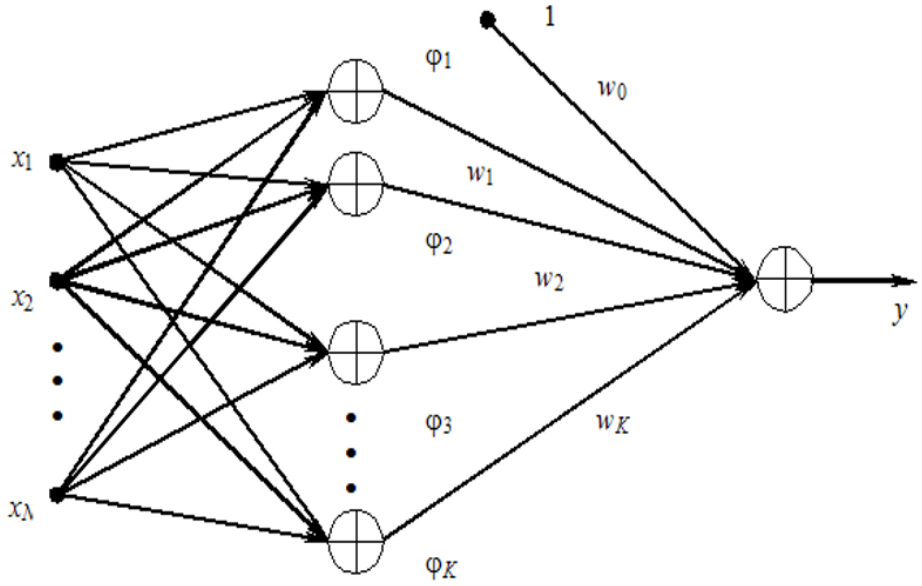


Рисунок 2.3 – Архітектура радіальних мереж

Радіальні мережі мають фіксовану структуру з одним прихованим шаром та лінійними вихідними нейронами, тоді як сигмоїдальні мережі можуть мати різну кількість шарів і вихідні нейрони можуть бути як лінійними, так і нелінійними.

Один з найпростіших, хоча не найефективніших, способів визначення параметрів базисних функцій - це випадковий вибір. В цьому випадку центри (сі) базисних функцій вибираються випадковим чином за рівномірним розподілом. Цей підхід є припустимим для класичних радіальних мереж, якщо рівномірний розподіл навчальних даних добре відповідає характеру завдання.

Мережі з радіальними базисними функціями знаходять застосування в задачах класифікації, апроксимації функцій багатьох змінних та прогнозування, як і сигмоїдальні мережі. Однак, вони реалізують інші методи обробки даних, пов'язані з локальними відображеннями, що спрощує та прискорює процес навчання.

2.3 Загально стуктура нейроної мережі

Для визначення структури нейронної мережі потрібно специфіку вхідних та вихідних даних.

2.3.1 Вхідні дані

Для ефективного навчання моделі прогнозування попиту необхідно враховувати низку параметрів, які наведені у таблиці 2.1.

Таблиця 2.1 – Параметри проведення навчання моделі прогнозування

Назва параметру	*Тип даних*	*Розмір*
Назва моделі	string	Рядок символів Unicode
К-сть епох навчання	int	32 біт
К-сть покупців	int	32 біт
Продукція	int	32 біт
Ціна	double	64 біт
Знижка	double	64 біт
Категорія	int	32 біт
Період продажу продукції	DateTime	Структура

2.3.2 Вихідні дані

Система прогнозування продажу продукції забезпечує формування результатів у вигляді звітності, яка включає таблиці та графіки з прогнозованими даними про кількість покупців, обсяг придбаних товарів, дати продажу, загальні витрати на товари та прогнозований прибуток.

Результати формування звітності щодо прогнозування продажу продукції представлені в табличному вигляді у таблиці 2.2.

Таблиця 2.2 – Вихідні дані прогнозування

Назва параметру	*Тип даних*	*Розмір*

Період продажі	DateTime	Структура
Продукція	int	32 біт
К-сть проданої продукції	int	32 біт
К-сть продавців	int	32 біт
Кількість покупців	int	32 біт
Загальна сума виручки	double	64 біт
Прибуток	double	64 біт
Заклад	int	32 біт

Враховуючи особливості навчання нейронної мережі, вона може бути налаштована для прогнозування відповідних параметрів обсягу продажу вибраної продукції в закладі (ресторані або кав'ярні) протягом заданого періоду часу.

2.4 Розробка методу навчання нейронної мережі

Алгоритм зворотного розповсюдження помилки є широко використовуваним методом навчання багатошарових персептронів - нейронних мереж з прямим поширенням. Він належить до категорії методів навчання з учителем, оскільки вимагає наявності цільових значень у навчальних прикладах. Алгоритм зворотного розповсюдження помилки також є одним з найвідоміших алгоритмів машинного навчання.

На основі цього алгоритму використовується вихідна помилка нейронної мережі як основний принцип. Ідея полягає у використанні градієнтного спуску для налаштування вагових коефіцієнтів мережі, зменшуючи вихідну помилку. Процес навчання передбачає поширення помилки від виходу мережі назад до входу, коригуючи ваги на кожному шарі. Цей ітеративний процес триває до досягнення заданої точності або зменшення помилки мережі [12].

В основі ідеї алгоритму лежить використання вихідної помилки нейронної мережі:

$$E = \frac{1}{2}\sum_{i=1}^{k}(y - y')^2 \qquad (2.9)$$

де k - число вихідних нейронів мережі, y - цільове значення, y' - фактичне вихідне значення. Алгоритм зворотного розповсюдження помилки є ітеративним методом навчання, що використовує принцип навчання "по кроках" або в режимі "онлайн". Це означає, що після подання одного навчального прикладу на вхід мережі, вагові коефіцієнти нейронів коригуються негайно.

В кожній ітерації здійснюється 2 проходи моделі - прямий та зворотний. Прямий прохід включає поширення вхідних даних від початку до кінця моделі, що формує вихідні значення, відповідні поточним вагам. Потім обчислюється помилка моделі, яка представляє розбіжність між фактичними значеннями та очікуваними значеннями. На зворотному проході ця помилка розповсюджується від виходу до входу моделі, і змінюються ваги відповідно до визначеного правила оновлення:

$$\Delta w_{j,i}(n) = -\eta \frac{\partial E_{av}}{\partial w_{ij}} \qquad (2.10)$$

де w_{ij} – вага i-го зв'язку та j-го нейрона, η - параметр швидкості навчання, який дозволяє додатково керувати величиною кроку корекції $\Delta w_{j,i}$ з метою більш точного налаштування на мінімум помилки і підбирається експериментально в процесі навчання (змінюється в інтервалі від 0 до 1).

Враховуючи, що вихідна сума j-го нейрона дорівнює:

$$S_j = \sum_{i=1}^{n} w_{i,j} x_i \qquad (2.11)$$

можна показати, що:

$$\frac{\partial \mathrm{E}}{\partial w_{i,j}} = \frac{\partial \mathrm{E}}{\partial S_j}\frac{\partial S_j}{\partial w_{i,j}} = x_i \frac{\partial \mathrm{E}}{\partial S_j} \qquad (2.12)$$

Згідно останнього виразу, диференціал ∂S_j активаційної функції *f(s)* нейронів мережі повинен існувати та не дорівнювати нулю в будь-якій точці, що означає, що активаційна функція повинна бути диференційованою на всій числовій осі. Тому для використання методу зворотного поширення найчастіше використовуються сигмоїдні активаційні функції, наприклад,

логістичну або гіперболічний тангенс.

Таким чином, алгоритм зворотного поширення помилки використовує стохастичний градієнтний спуск, що дозволяє «рухатися» в багатовимірному просторі ваг в напрямку антиградієнта з метою досягнення мінімуму функції помилки.

Цей алгоритм був обраний для навчання нейронної мережі через його простоту реалізації та стійкість до аномалій та викидів у вихідних даних. Він відомий своєю ефективністю та широким застосуванням у багатьох задачах навчання.

2.5 Розробка алгоритму процесу аналізу та прогнозування

З метою досягнення високої точності прогнозування попиту на асортимент продукції в сфері обслуговування, використання нейронних мереж вимагає проведення ефективного процесу навчання.

Для реалізації алгоритму навчання необхідно спочатку отримати дані для навчання з бази даних. У цьому файлі містяться параметри, які впливають на попит, а також фактичні значення попиту на вибрану продукцію за визначений період, отримані шляхом моделювання продажів.

На рисунку 2.4 представлений алгоритм аналізу та прогнозування на основі часових рядів з підтримкою прийняття рішень. Цей алгоритм слугує основою для використання нейронних мереж у процесі прогнозування попиту.

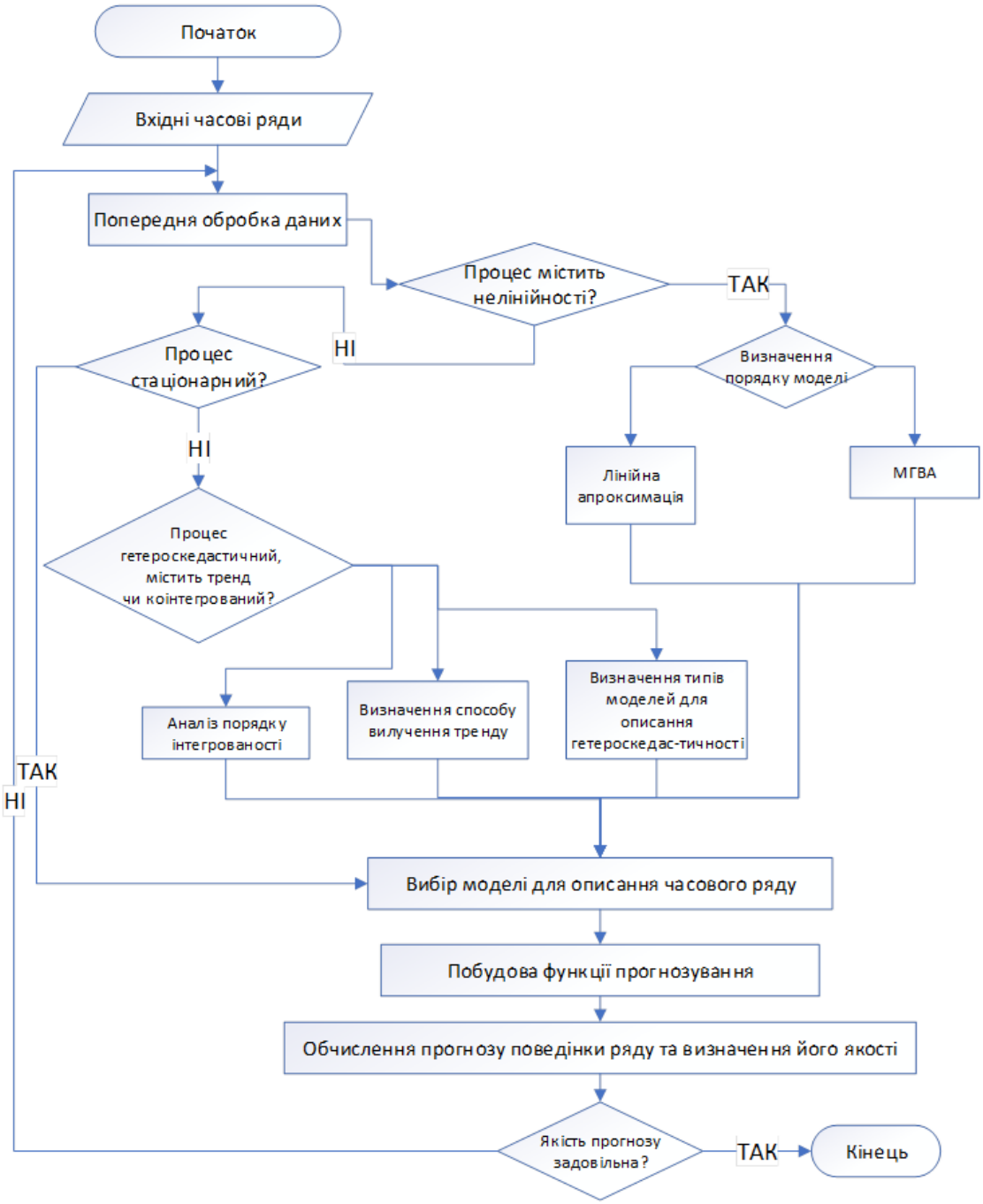


Рисунок 2.4 – Алгоритм процесу аналізу та прогнозування

Алгоритм складається із такої послідовності кроків:

1. Попередня обробка та аналіз даних.
2. Перевірка наявності нелінійностей.
3. Визначення порядку нелінійності. Якщо порядок не лінійності визначений, переходимо до кроку 9.
4. Перевірка стаціонарності процесу. Якщо процес стаціонарний, переходимо до кроку 9, інакше переходимо до кроку 5.

5. Перевірка наявності гетероскедастичності. Якщо процес має гетероскедастичну структуру, переходимо до кроку 6. Якщо присутній тренд, переходимо до кроку 7, в іншому випадку до кроку 8.

6. Вибір моделі для опису гетероскедастичності. Після вибору відповідної моделі переходимо до кроку 9.

7. Визначення способу вилучення або моделювання тренду. Переходимо до кроку 9.

8. Для коінтегрованих процесів будується модель корекції похибок. Переходимо до кроку 10.

9. Будується модель часового ряду, обчислюються критерії адекватності моделі. Переходимо до кроку 10.

10. Будується функція прогнозування на основі обраної моделі. Обчислюється прогноз поведінки ряду та оцінки точності прогнозу.

11. Якщо точність прогнозу не задовольняє вимоги приймаючої особи, уточнюються вхідні дані і повертаємося до кроку 1. Інакше, процес завершується.

2.6 Вибір методу навчання нейромережі згідно досліджень

Для навчання моделей прогнозування попиту на продукцію бізнесу обслуговування були використані дані з ресурсу "kaggle.com" [13]. Даний ресурс надає доступ до якісних навчальних даних щодо попиту на продукцію у кав'ярнях, які є придатними для тренування нейронних мереж.

Розглянемо створені нейронні мережі. В якості кандидатів були обрані наступні типи нейронних мереж: багатошаровий персептрон, узагальнена регресійна нейронна мережа та мережа з радіальною базисною функцією (РБФ). Усі вони були реалізовані за допомогою мови програмування C# та бібліотеки Accord.NET.

Accord.NET - це платформа для наукових обчислень у середовищі .NET. Вихідний код проекту доступний відповідно до умов Суспільної ліцензії обмеженого застосування GNU, версія 2.1.

Платформа Accord.NET включає в себе набір бібліотек, доступних у вихідному коді, а також через установщики та пакети NuGet. Вона охоплює різні області, такі як чисельна лінійна алгебра, чисельна оптимізація, статистика, машинне навчання, штучні нейронні мережі, обробка сигналів і зображень, а також надає додаткові функціональні можливості, такі як побудова графіків та візуалізація даних.

Початково проект Accord.NET був розроблений з метою розширення можливостей AForge.NET Framework, але згодом він став самостійною платформою, включаючи в себе функціональні можливості AForge.NET. Під загальною назвою Accord.NET були об'єднані обидві платформи [14].

Accord.NET Framework отримав визнання у галузі машинного навчання і був згаданий у декількох книгах, зокрема "Mastering.NET Machine Learning", опублікованій PACKT, а також "F# для програм машинного навчання", представленої на конференції QCON San Francisco [15].

Для порівняння побудованих мереж будуть використовуватися метрики середньоквадратичної похибки (MSE), квадратного кореня середньоквадратичної похибки (RMSE) та квадрату відносної середньоквадратичної похибки (RMSPE).

Почнемо з розгляду радіальної базисної мережі (RBF-мережі). Було створено екземпляр RBF-мережі з чотирма вхідними вузлами, які відповідають кожному вхідному значенню даних, п'ятьма прихованими вузлами та трьома вихідними вузлами, відповідними кожному вихідному значенню даних. Кількість прихованих вузлів була визначена переважно методом проб і помилок, і в даному випадку було вибрано п'ять прихованих вузлів.

Навчання RBF-мережі включає три етапи. На першому етапі визначаються центроїди. Ці центроїди можна розглядати як представники значень x, вибраних з навчальних даних. RBF-мережа вимагає наявності одного центроїда для кожного прихованого вузла, отже, у даному випадку потрібно п'ять центроїдів. Алгоритм навчання вибирає значення x з

навчальних даних для визначення цих центроїді [16].

На другому етапі проводиться визначення інтервалів ширини, які представляють відстань між центроїдами. Для кожного прихованого вузла необхідне одне значення ширини. Замість обчислення п'яти окремих значень ширини, було обчислено єдиний загальний інтервал ширини зі значенням 3.3318 для всіх п'яти прихованих вузлів.

На третьому етапі були визначені вагові значення і зміщення. Ці значення є числовими константами. У випадку RBF-мережі з NI вхідними вузлами, NH прихованими вузлами та NO вихідними вузлами, необхідно мати NH*NO вагових значень та NO зсувів. Оскільки в нашій RBF-мережі архітектура «4-5-3», нам потрібно 5 * 3 = 15 вагових значень плюс три зміщення, загалом отримується 18 вагових значень і зміщень.

Для наступної мережі було використано багатошаровий перцептрон (MLP). Було прийняте рішення побудувати мережу з трьома прихованими шарами, додавши шар вибування (dropout) після кожного з цих шарів. Шар вибування використовується для додаткового контролю перенавчання мережі. Головним параметром цього шару є рівень вибування (dropout rate), який приймає значення від 0 до 1. Цей рівень визначає ймовірність кожного виходу мережі бути вимкненим, тоді як усі інші виходи будуть помножені на значення 1/(1 – rate), щоб зберегти загальну суму виходу приблизно незмінною.

Останнім вибраним варіантом мережі була нейронна мережа з узагальненою регресією (GRNN).

GRNN є варіацією радіальної базисної функційної мережі (RBF), яка має сильну здатність до нелінійного моделювання та швидкого навчання. Вона має численні переваги порівняно з RBF. GRNN здатна досягти оптимальної регресії з великим кластером навчальних даних, навіть при наявності обмеженої вибірки даних. Вона також ефективно працює з нестабільними даними. Хоча може здатися, що GRNN не має точності радіальної базисної функції, насправді вона демонструє значні переваги в класифікації та адаптації, особливо коли точність вихідних даних невисока.

GRNN представляє собою покращену версію RBF з аналогічною структурою. Вона включає додатковий рівень підсумовування, а зв'язки ваг між прихованим шаром і вихідним шаром (здійснюється за допомогою методу найменших квадратів гаусівських ваг) виключаються [17].

Навчання нейронної мережі складається з таких етапів:

– вхідний шар. Цей шар представляє собою вектор вхідних даних з розмірністю m та кількістю вибірок n. Лінійна функція, яка використовується, є передавальною функцією, що передає вхідні дані без змін;

– прихований шар. Цей шар повністю зв'язаний з вхідним шаром, але не містить жодних зв'язків усередині себе. Кількість нейронів у прихованому шарі дорівнює кількості вибірок, тобто n. Для передачі сигналу у прихований шар використовується радіальна базисна функція;

– додатковий шар. Цей шар містить два вузли. Перший вузол представляє вихідну суму кожного нейрона у прихованому шарі, а другий вузол представляє зважену суму очікуваного результату та кожного нейрона у прихованому шарі;

– вихідний шар. Вихідним значенням мережі є результат, який отримується шляхом ділення значення другого вузла вихідного шару на значення першого вузла.

Кількість епох навчання для всіх мереж була встановлена на 400 циклів. Однак, навчання буде припинено, якщо значення функції втрат для тестових даних не покращується протягом 10 епох.

Отримані результати навчання нейронних мереж наведені у таблиці 2.3.

Таблиця 2.3 – Результати проведеного тренування нейронних мереж

Мережа	*К-сть епох навчання*	*MSE*	*RMSE*	*RMSPE*
Радіально -базисна мережа	153	0,2211	0,4861	2,3832
Багатошаровий перцептрон	326	0,0957	0,3103	1,6298
НМ з узагальненою регресією	128	0,0869	0,2501	1,3299

За даними таблиці 2.3, видно, що модель нейронної мережі з узагальненою регресією виявилася найефективнішою під час процесу навчання. Це може бути пояснено її сильними нелінійними властивостями та високою швидкістю навчання, а також перевагами, які перевищують інші моделі, що були протестовані.

Проте, слід зазначити, що більш складна структура цієї мережі призвела до значного збільшення часу та обчислювальних ресурсів, необхідних для навчання. Незважаючи на це, модель нейронної мережі з узагальненою регресією була збережена для подальшого використання у системі підтримки прийняття рішень.

Результати прогнозування, отримані за допомогою моделі нейронної мережі з узагальненою регресією, можна ознайомитися у таблиці 2.4.

Таблиця 2.4 – Прогнозування за допомогою НМ з GRNN

Реальні значення	5425	8538	6520	5319	7389	4073	9154
Прогнозовані	5612	8457	6217	5451	7182	3980	9253

Незважаючи на наявну похибку, прогнозні результати можна вважати прийнятними, оскільки вони достатньо добре відображають ситуацію щодо майбутнього попиту на продукцію бізнесу обслуговування. Поява певної

похибки може бути пояснена обмеженою кількістю даних, які були використані для навчання моделі. Недостатня кількість даних може призвести до певної непевності в прогнозуванні та обмеженої точності результатів. Проте, не зважаючи на це, отримані результати все ж демонструють прийнятну якість та корисність моделі у прогнозуванні попиту на продукцію бізнесу обслуговування.

2.7 Висновок

В цьому розділі розглянуто поняття штучних нейронних мереж та базові концепції, пов'язані з ними. Було описано декілька типів нейронних мереж, які можуть бути використані для прогнозування попиту на продукцію в сфері бізнесу обслуговування. Була розглянута їх структура та принципи відбору параметрів. Також був описаний алгоритм аналізу та прогнозування на основі часових рядів з підтримкою прийняття рішень.

Розділ завершується оглядом методології, яка використовується для побудови та оцінки кандидатських моделей з метою визначення найбільш адекватної нейронної мережі, що описує модельований процес. Згідно з результатами проведених експериментів, модель НМ з узагальненою регресією можна вважати фаворитом навчання. Цю модель було використано для розробки інформаційної системи підтримки прийняття рішень для бізнесу обслуговування.

3 ВИБІР ЗАСОБІВ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір технологій реалізації

Вибір мови програмування є критичним етапом у процесі розробки програмного забезпечення.

Python - це високорівнева, інтерпретована мова програмування, яка була розроблена Гвідо ван Россумом у 1991 році. Ця мова відрізняється своїм чистим синтаксисом та зрозумілістю коду, що робить її популярною серед початківців. Python знаходить широке застосування у веб-розробці, автоматизації, наукових дослідженнях, аналізі даних, штучному інтелекті та машинному навчанні [18].

Java є високорівневою, об'єктно-орієнтованою мовою програмування, яку створила компанія Sun Microsystems у 1995 році (в даний час вона належить компанії Oracle) [19]. Розроблена Джеймсом Гослінгом, Java вирізняється принципом "Напиши одного разу, запускай будь-де" завдяки використанню віртуальної машини Java (JVM), що дозволяє виконувати код на різних платформах. Java широко використовується в розробці різноманітних корпоративних додатків, веб-додатків, мобільних додатків на платформі Android, систем вбудованих пристроїв та Інтернету речей [20].

C# (вимовляється як "С-шарп") є об'єктно-орієнтованою мовою програмування, що була створена компанією Microsoft у 2000 році як частина платформи .NET. Розроблена Андерсом Хейльсбергом, C# поєднує найкращі характеристики мов програмування C++ та Java. Вона застосовується для розробки різноманітних додатків, включаючи веб-додатки, мобільні додатки, додатки для платформи Windows, ігри (зокрема з використанням движка Unity), корпоративні програмні рішення та хмарні сервіси. C# працює в середовищі .NET, що надає широкий набір бібліотек та фреймворків для швидкої та ефективної розробки програмного забезпечення [21].

В табл. 3.1 був здійснений аналіз можливостей кожної з мов програмування з метою порівняння.

Таблиця 1.3 – Порівняльний аналіз мов програмування

Характеристика	Python	Java	C#
Типізація	Динамічна	Статична	Статична
Стиль програмування	Об'єктно-орієнтований, процедурний	Об'єктно-орієнтований	Об'єктно-орієнтований, компонентний
Виконання коду	Інтерпретована	Компіляція у байт-код, JVM	Компіляція у проміжний код, CLR
Типізація	Динамічна	Статична	Статична
Синтаксис	Легкий, відступи для блоків	Схожий на C++, більш формальний	Схожий на C++, Java, більш формальний
Веб-розробка	Django, Flask, FastAPI	Spring, JavaEE, Servlets, JSP	ASP.NET, Blazor
Мобільна розробка	Kivy, BeeWare, Chaquopy (Android)	Android SDK	Xamarin, .NET MAUI
Наукові дослідження	NumPy, SciPy, Pandas, TensorFlow	Weka, Java-ML, Deeplearning4j	Math.NET, CNTK, ML.NET
Ігрова розробка	Pygame, Panda3D, Godot (через GDNative)	LibGDX, LWJGL, jMonkeyEngine	Unity, MonoGame, Xenko
Крос-платформність	Висока	Висока	Висока (завдяки .NET)
Характеристика	Python	Java	C#

Вибір мови програмування C# для розробки системи обумовлений його продуктивністю, крос-платформністю та активною підтримкою з боку

компанії Microsoft. C# забезпечує модульність та масштабованість, що дозволяє створювати стабільні та гнучкі рішення. Завдяки багатому набору бібліотек та фреймворків, розробка інформаційної системи підтримки прийняття рішень для бізнесу обслуговування стає швидкою та ефективною.

3.2 Вибір архітектурного шаблону

Розробка програмного забезпечення вимагає створення ефективної, надійної та масштабованої системи, яка задовольняє потреби користувачів. Вибір вірної архітектури є ключовим етапом у розробці будь-якої програмної системи, оскільки він може впливати на всі аспекти програми, від швидкодії та масштабованості до легкості розширення та підтримки.

Трирівнева архітектура — це архітектурний шаблон, який розбиває програму на три основні рівні: презентаційний рівень (інтерфейс користувача), рівень бізнес-логіки (процеси обробки даних) та рівень даних (зберігання та управління даними). Цей шаблон забезпечує незалежність компонентів та спрощує розробку та підтримку програмного забезпечення.

Архітектурний шаблон MVC (Model-View-Controller) розбиває програму на три основні компоненти: модель (Model), представлення (View) та контролер (Controller). Модель відповідає за управління даними та бізнес-логіку, представлення відображає дані користувачеві, а контролер забезпечує зв'язок між моделлю та представленням, оброблюючи взаємодію користувача. Цей шаблон розділяє відповідальності між компонентами, забезпечуючи модульність, розширюваність та полегшуючи тестування та підтримку програмного забезпечення [22].

Сервісно-орієнтована архітектура (SOA) є архітектурним шаблоном, який включає розбиття програмного забезпечення на незалежні сервіси, що взаємодіють між собою через стандартизовані протоколи та інтерфейси. Кожен сервіс виконує конкретну функціональність та може бути розроблений, розгорнутий та масштабований незалежно від інших сервісів, що забезпечує системі гнучкість та надійність. SOA сприяє розподіленій розробці, забезпечує

повторне використання коду та прискорює процес розгортання та оновлення системи [23].

В табл. 3.2 наведено порівняльний аналіз основних архітектурних шаблонів, використованих у розробці програмного забезпечення.

Таблиця 3.2 – Порівняльний аналіз архітектурних шаблонів

Характеристика	Три-ярусна архітектура	MVC	Сервісно-орієнтована архітектура
Розділення відповідальності	Розділяє програму на 3 рівні	Розділяє програму на 3 компоненти	Розділяє програму на незалежні сервіси
Супроводження та розширення	Середня складність	Відносно легке розширення та супроводження	Легке розширення та супроводження
Гнучкість	Середня гнучкість	Висока гнучкість	Висока гнучкість
Масштабованість	Обмежена масштабованість	Обмежена масштабованість	Висока масштабованість
Інтеграція	Легка інтеграція компонентів	Легка інтеграція компонентів	Складніша інтеграція через сервіси
Тестування	Середня складність тестування	Легше тестування окремих компонентів	Легше тестування окремих сервісів
Взаємодія компонентів	Відносно проста взаємодія	Стандартизована взаємодія	Взаємодія через стандартизовані протоколи

Вибір трирівневої архітектури для розробки базується на потребі чіткого

розділення відповідальності між рівнями презентації, бізнес-логіки та даних. Цей архітектурний підхід сприяє створенню стабільної, надійної та легко супроводжуваної системи, здатної ефективно виконувати функції управління складом логістичних компаній.

Трирівнева архітектура дозволяє відокремити презентаційний рівень, що відповідає за інтерфейс користувача, від бізнес-логіки, яка оброблює логічні процеси та бізнес-правила, а також від рівня даних, який забезпечує зберігання та управління інформацією. Це розділення дозволяє розробникам працювати над кожним рівнем незалежно, спрощує розробку, тестування та супроводження системи.

Також трирівнева архітектура забезпечує гнучкість системи, оскільки різні компоненти можуть бути легко інтегровані та замінені без значних змін на інших рівнях. Це компаніям адаптувати систему до змін у бізнес-процесах, впроваджувати нові функціональності та розширювати можливості.

3.2.1 Особливості розробки бази даних

ER-діаграма (схема "сутність-зв'язок" або ERD) є специфічним типом блок-схеми, яка служить для візуалізації та моделювання взаємозв'язків між різними "сутностями" або елементами всередині певної системи. Сутності можуть представляти реальні або концептуальні об'єкти, людей, місця, події тощо. ER-діаграми широко використовуються в процесі проектування та розробки реляційних баз даних, а також у сферах освіти, наукових досліджень та розробки програмного забезпечення для бізнесу.

Вони базуються на стандартному наборі символів, які включають прямокутники, ромби, овали та лінії зв'язку. Ці символи використовуються для відображення сутностей, атрибутів та зв'язків між ними. Кожна сутність має свої атрибути, які відображають характеристики цієї сутності. Зв'язки між сутностями показують, як вони взаємодіють або пов'язані одна з одною. ER-діаграми конструюються за аналогією з граматичними структурами, де сутності виступають в ролі іменників, а зв'язки виконують роль дієслів.

ER-діаграми, також відомі як діаграми "сутність-зв'язок" (або ERD), є різновидом схем структури даних (DSD). У цих діаграмах відображаються відношення між елементами всередині сутностей, замість зв'язків між самими сутностями. ER-діаграми часто використовуються в поєднанні з діаграмами потоку даних (DFD), які ілюструють рух інформації в системі або процесі.

У моделях даних та ER-моделях зазвичай виділяються три рівні деталізації. Перший рівень - це концептуальна модель даних, яка представляє загальну структуру моделі та архітектуру системи з мінімальною кількістю деталей. Для менш складних систем можна обійтися без цієї моделі та перейти відразу до наступного рівня [24].

Другий рівень - це логічна модель даних, яка включає більш детальну інформацію, ніж концептуальна модель. На цьому рівні визначаються операційні та транзакційні сутності більш детально. Логічна модель залежить від конкретної технології, у якій буде застосовуватися.

Третій рівень - це фізична модель даних, яка базується на логічній моделі. Фізична модель містить достатньо технічних деталей для розробки та впровадження фактичної бази даних. Зазвичай на цьому рівні створюють одну або дві фізичні моделі відповідно до потреб реалізації бази даних.

Використання цих рівнів деталізації дозволяє розробляти систему з урахуванням її загальної структури, операційних потреб та технічних аспектів.

Має сенс звернути увагу на те, що існують альтернативні підходи до подання структури даних, які можуть мати схожі рівні масштабу та деталізації. Наприклад, у діаграмах потоку даних (DFD) також використовуються рівні, але з іншою інтерпретацією та розподілом інформації. Також варто зазначити, що деякі розробники можуть додавати додаткові рівні та ієрархії до ER-діаграм, якщо це необхідно для певного дизайну бази даних. Наприклад, можуть бути використані суперкласи та підкласи для розширення моделі.

Проте важливо зрозуміти, що ER-діаграми призначені для відображення зв'язків та відношень між елементами, зокрема в контексті реляційних даних. Вони не враховують інші типи даних або їх структури, тому можуть бути

обмежені для використання з нереляційними або частково структурованими даними. Крім того, інтеграція ER-моделей з існуючою базою даних може бути складною через відмінності в архітектурі.

Ці обмеження варто враховувати при застосуванні ER-діаграм для моделювання та розробки баз даних, особливо в контексті більш складних та розношерстних систем. Інші підходи та інструменти можуть бути використані для врахування специфічних потреб та характеристик даних [25].

За допомогою Системи Управління Базами Даних (СУБД) MS Access була розроблена сутнісно-зв'язкова (ER) діаграма бази даних, яка є фундаментом для системи підтримки прийняття рішень в галузі бізнесу обслуговування (див. рис. 3.1). Ця діаграма відображає ключові сутності, їх взаємозв'язки та атрибути, необхідні для зберігання та організації важливої інформації, яка використовується під час процесу прийняття рішень. Вона допомагає створити структуровану та систематизовану базу даних, яка відображає логіку та зв'язки між різними аспектами бізнесу обслуговування, що забезпечує зручний доступ та обробку інформації для аналізу та вирішення задач прийняття рішень [26].

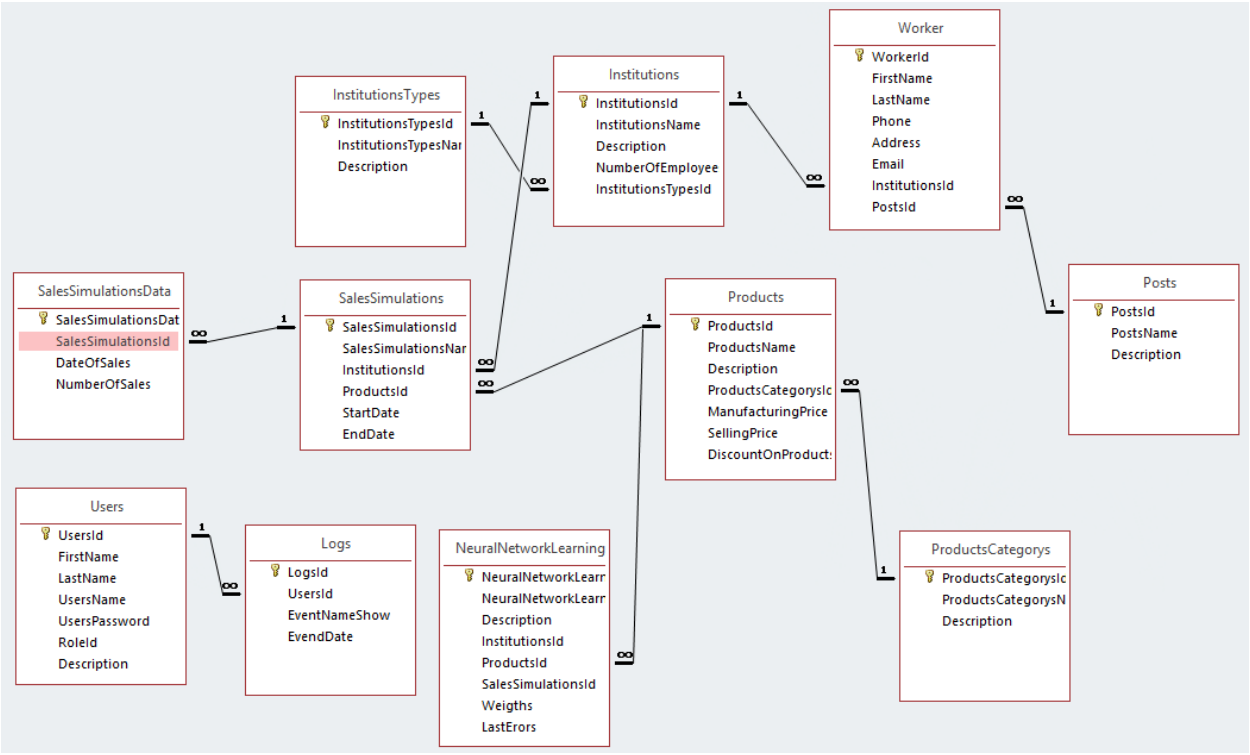


Рисунок 3.1 – Діаграма бази даних

На основі представленої на рисунку 1.6 діаграми бази даних, можна встановити, що база даних проекту складається з 11 сутностей, які детально досліджують предметну область. Ось опис цих сутностей:

1. Сутність "Institutions" - зберігає інформацію про всі заклади бізнесу обслуговування, такі як ресторани та кав'ярні.
2. Сутність "InstitutionsTypes" - містить інформацію про типи закладів.
3. Сутність "Logs" - зберігає дані про активність користувачів системи та їхні дії.
4. Сутність "NeuralNetworkLearning" - містить інформацію про процес навчання нейронної мережі.
5. Сутність "Posts" - зберігає інформацію про посади працівників.
6. Сутність "Products" - містить дані про продукцію.
7. Сутність "ProductsCategorys" - зберігає інформацію про категорії продукції.
8. Сутність "Users" - містить дані про всіх користувачів системи та їхні облікові записи.
9. Сутність "Worker" - зберігає інформацію про працівників.
10. Сутність "SalesSimulations" - містить загальні дані щодо симуляції продажу у закладах.
11. Сутність "SalesSimulationsData" - зберігає результати симуляції продажу продукції у закладах.

3.2.2 Особливості розробки бізнес-логіки

Бізнес-логіка в контексті розробки інформаційних систем охоплює набір правил, принципів та залежностей, що визначають поведінку об'єктів в предметній галузі, яку система підтримує. Вона може бути також описана як реалізація правил та обмежень, які підлягають автоматизації. Термін "логіка предметної області" (англ. domain logic) використовується як синонім до бізнес-логіки. Бізнес-логіка визначає правила, якими керуються дані в

предметній галузі.

Проще кажучи, бізнес-логіка є реалізацією предметної галузі в інформаційній системі. Вона включає в себе розрахункові формули для щомісячних виплат з позик (у фінансовій індустрії), автоматизовану відправку повідомлень електронної пошти керівнику проекту після завершення певних частин завдання всіма підлеглими (у системах управління проектами), а також відмову від готелю при скасуванні рейсу авіакомпанією (у туристичній галузі) та інші подібні випадки.

Під час фази бізнес-моделювання та розробки вимог, бізнес-логіка може бути описана наступними способами:

– у вигляді тексту, де формулюються правила та умови, що регулюють функціонування системи;

– застосуванням концептуальних аналітичних моделей предметної галузі, таких як онтології, які описують сутність та зв'язки між різними елементами;

– визначенням бізнес-правил, які встановлюють поведінку системи та обмеження на операції;

– використанням різноманітних алгоритмів для опису обчислювальних процесів та логічних операцій;

– представленням у вигляді діаграм діяльності, які ілюструють послідовність кроків та взаємодію об'єктів;

– використанням графів та діаграм переходу станів для опису поведінки системи у різних станах та переходів між ними;

– моделюванням бізнес-процесів, що включають послідовність дій та взаємодію між різними учасниками процесу.

У процесі аналізу та проектування системи, бізнес-логіка може бути представлена за допомогою різних діаграм мови UML або подібних до неї. Під час фази програмування, бізнес-логіка реалізується в коді класів та їх методів, коли застосовуються об'єктно-орієнтовані мови програмування, або в

процедурах та функціях, коли використовуються процедурні мови.

Термін "бізнес-логіка" також використовується для позначення програмних модулів, що реалізують цю логіку, а також рівня системи, на якому знаходяться ці модулі (відомий як "business logic layer" або BLL). У багаторівневих інформаційних системах, цей рівень взаємодіє з нижчим рівнем інфраструктурних сервісів, таких як рівень доступу до даних або рівень сервісів додатка, а також з рівнем користувальницького інтерфейсу та зовнішніми системами [27].

BLL відповідає за обробку справ, які стосуються бізнес-сфери, а не бази даних або користувальницького інтерфейсу. У даному контексті, шар BLL містить клас "NeuronsBLL", який виконує логіку формування звітів та пошуку. Діаграма цього класу представлена на рисунку 3.2.

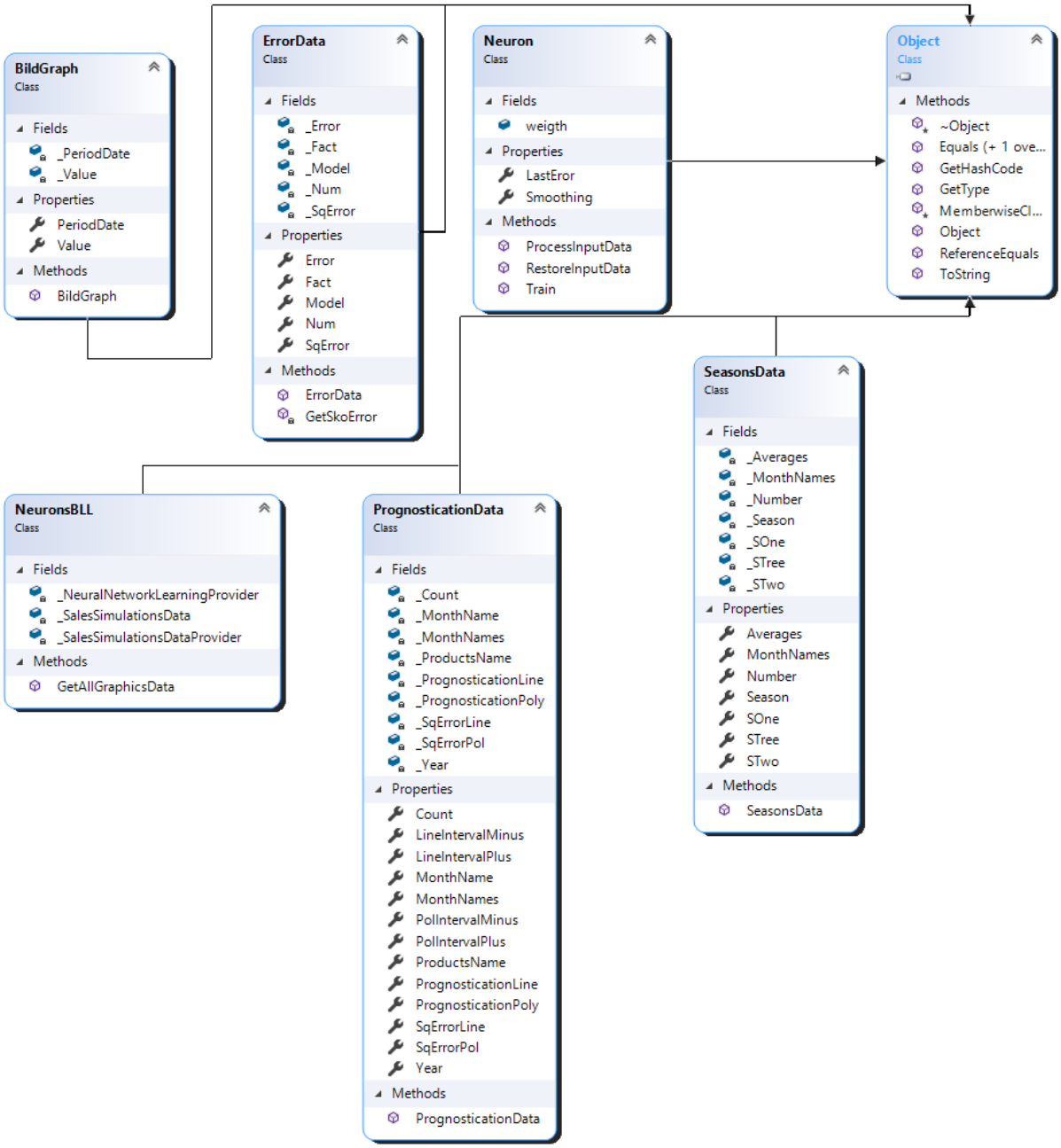


Рисунок 3.2 – Діаграма класів бізнес-логіки

У класі "NeuronsBLL" присутній публічний метод "GetAllGraphicsData", який має на меті повернути дані, необхідні для побудови графіка прогнозованих даних.

3.2.3 Особливості розробки рівня UI

Елементи інтерфейсу користувача (UI) є складовими частинами, які використовуються дизайнерами для створення програм та веб-сайтів. Вони додають взаємодію та інтерактивність до інтерфейсу користувача, надаючи

користувачам точки взаємодії під час навігації. Прикладами таких елементів є кнопки, смуги прокрутки, пункти меню та чекбокси.

Інтерфейс користувача (UI) використовує ці елементи для створення візуального дизайну та забезпечення консистентності продукту, зробивши його зручним для використання та простим у навігації без особливих зусиль з боку користувача.

UI елементи в інтерфейсі користувача (UI) зазвичай розподіляються на чотири основні групи:

– елементи керування введенням (Input Controls): ці елементи дозволяють користувачам вводити інформацію до системи. Вони включають такі компоненти, як текстові поля, кнопки, чекбокси, радіокнопки, випадаючі списки та інші елементи, які сприяють взаємодії з користувачем;

– компоненти навігації (Navigation Components): ці компоненти допомагають користувачам переміщатися по продукту або веб-сайту. Вони включають панелі вкладок, головне меню програми, кнопки навігації, посилання та інші елементи, які допомагають управляти навігацією користувача;

– інформаційні компоненти (Informational Components): ці компоненти використовуються для відображення інформації користувачу. Вони можуть містити текстові поля, мітки, іконки, графіки, таблиці та інші елементи, що передають важливу інформацію користувачу;

– контейнери (Containers): ці елементи використовуються для групування та організації інших елементів. Вони дозволяють структурувати вміст інтерфейсу, забезпечуючи логічний розподіл компонентів [28].

У даному проекті для розробки користувацького інтерфейсу було використано Windows Forms - набір інструментів, який надає засоби для створення графічного інтерфейсу у програмах на платформі Windows [29].

На рисунку 3.3 наведена діаграма програми з відображенням прав адміністратора системи.

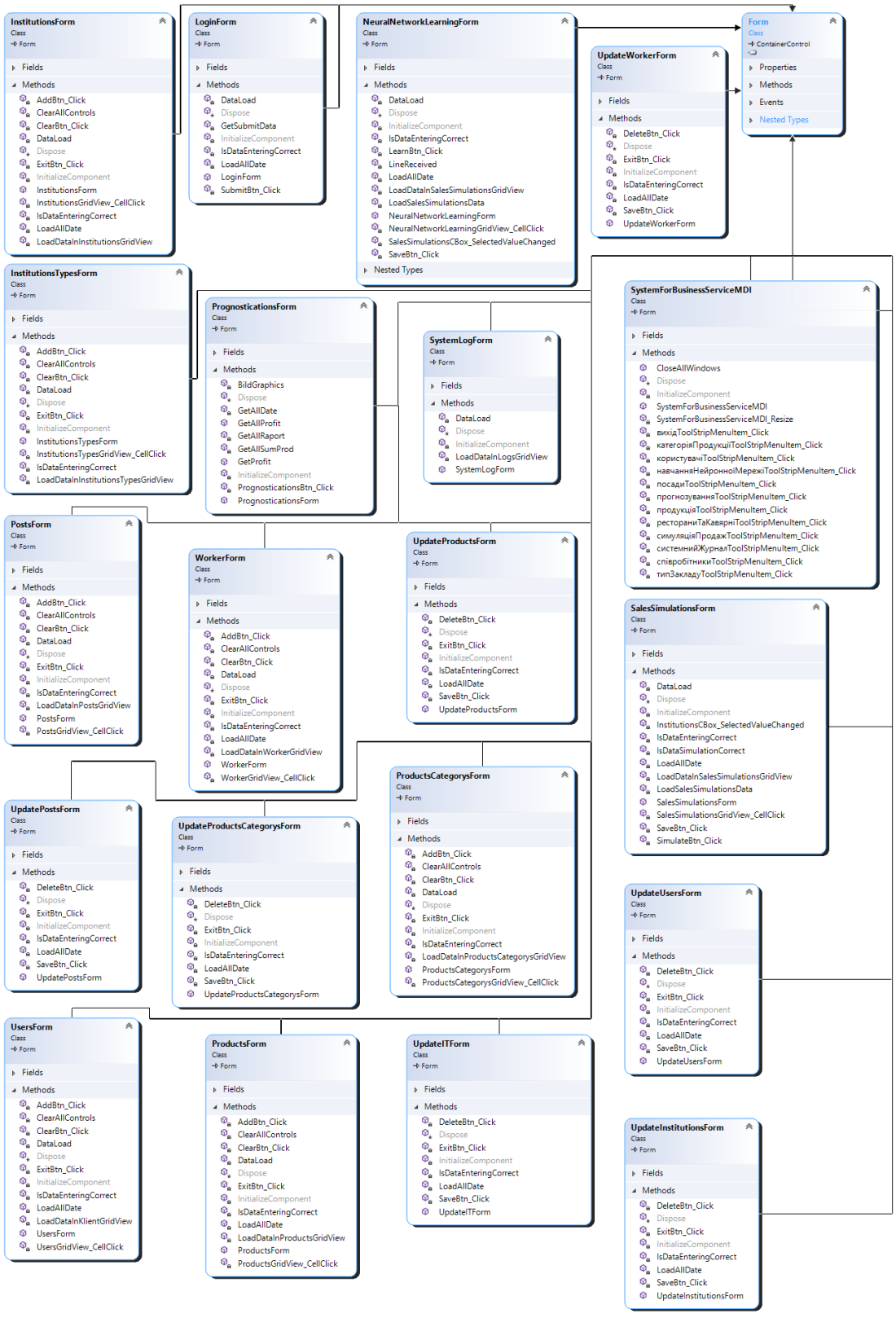


Рисунок 3.3 – Діаграма класів рівня користувацького інтерфейсу

Програма складається з двадцяти класів, які належать до рівня UI та є похідними від класу Form, що дозволяє їм мати графічний інтерфейс.

Один з основних класів програми - «SystemForBusinessServiceMDI» -

виступає в ролі головного вікна. Головне меню, розташоване в цьому вікні, є найважливішим елементом керування, оскільки воно надає можливість відкривати інші форми та виходити з програми.

Клас - «InstitutionsTypesForm», використовується для ведення інформації про типи закладів.

Клас «UpdateInstitutionsTypesForm» має за мету редагування інформації про обраний тип закладу.

Клас «InstitutionsForm» використовується для ведення інформації про заклади.

Клас "UpdateInstitutionsForm" має на меті редагування інформації про вибраний заклад.

Клас "PostsForm" використовується для ведення інформації про посади працівників.

Клас "UpdatePostsForm" служить для редагування інформації про посади.

Клас "ProductsCategorysForm" використовується для ведення інформації про категорії продукції.

Клас "UpdateProductsCategorysForm" призначений для редагування інформації про вибрану категорію продукції.

Клас "WorkerForm" використовується для ведення інформації про працівників.

Клас "UpdateWorkerForm" призначений для редагування інформації про вибраного працівника.

Клас "ProductsForm" використовується для збереження інформації про продукцію.

Клас "UpdateProductsForm" призначений для внесення змін до інформації про вибрану продукцію.

Клас "NeuralNetworkLearningForm" використовується для проведення процесу навчання нейронної мережі.

Клас "PrognosticationsForm" служить для проведення прогнозування

попиту на продукцію.

Клас "SalesSimulationsForm" призначений для проведення симуляції продажу продукції.

Клас "UpdateGoodsForm" використовується для внесення змін до інформації про вибраний товар.

Клас "LoginForm" призначений для аутентифікації користувачів у системі.

Клас "SystemLogForm" використовується для перегляду системних подій.

Клас "UsersForm" служить для обробки даних про користувачів системи.

Клас "UpdateUsersForm" використовується для редагування інформації про користувачів системи.

Створення об'єктів і виклик методів інших класів відбувається під час взаємодії користувача з елементами графічного інтерфейсу програми.

3.2.4 Особливості розробки DAL

Шар доступу до даних (Data Access Layer - DAL) в програмному забезпеченні є компонентом, який забезпечує зручний доступ до даних, збережених у постійному сховищі, такому як реляційна база даних. Цей термін часто використовується в контексті оточення Microsoft.NET.

DAL може повертати об'єкти з їх атрибутами, замість простих рядків і полів з таблиці бази даних. Це дозволяє створювати модулі клієнта (або користувацькі модулі) з вищим рівнем абстракції. Така модель може бути реалізована за допомогою класу, який має методи доступу до даних, що безпосередньо викликають відповідні процедури бази даних. Інша реалізація може включати отримання або запис даних в файлову систему. DAL приховує складність сховища даних, що лежить в основі системи.

Замість використання прямих команд, таких як "створити", "видалити" або "оновити" в певній таблиці бази даних, можна створити клас та декілька

збережених процедур у базі даних. Ці процедури можуть бути викликані методом всередині класу, що повертає об'єкт з необхідними значеннями. Або команди створення, видалення та оновлення можуть бути виконані за допомогою простих збережених функцій, розташованих у шарі доступу до даних [30].

Також методи бізнес-логіки програми можуть бути пов'язані з шаром доступу до даних.

Для роботи з базою даних було реалізовано 7 класів. Кожен клас отримав назву, що відповідає відповідній таблиці бази даних, та має приставку "Provider".

На рис. 3.4 наведена діаграма класів з методами для роботи з базою даних.

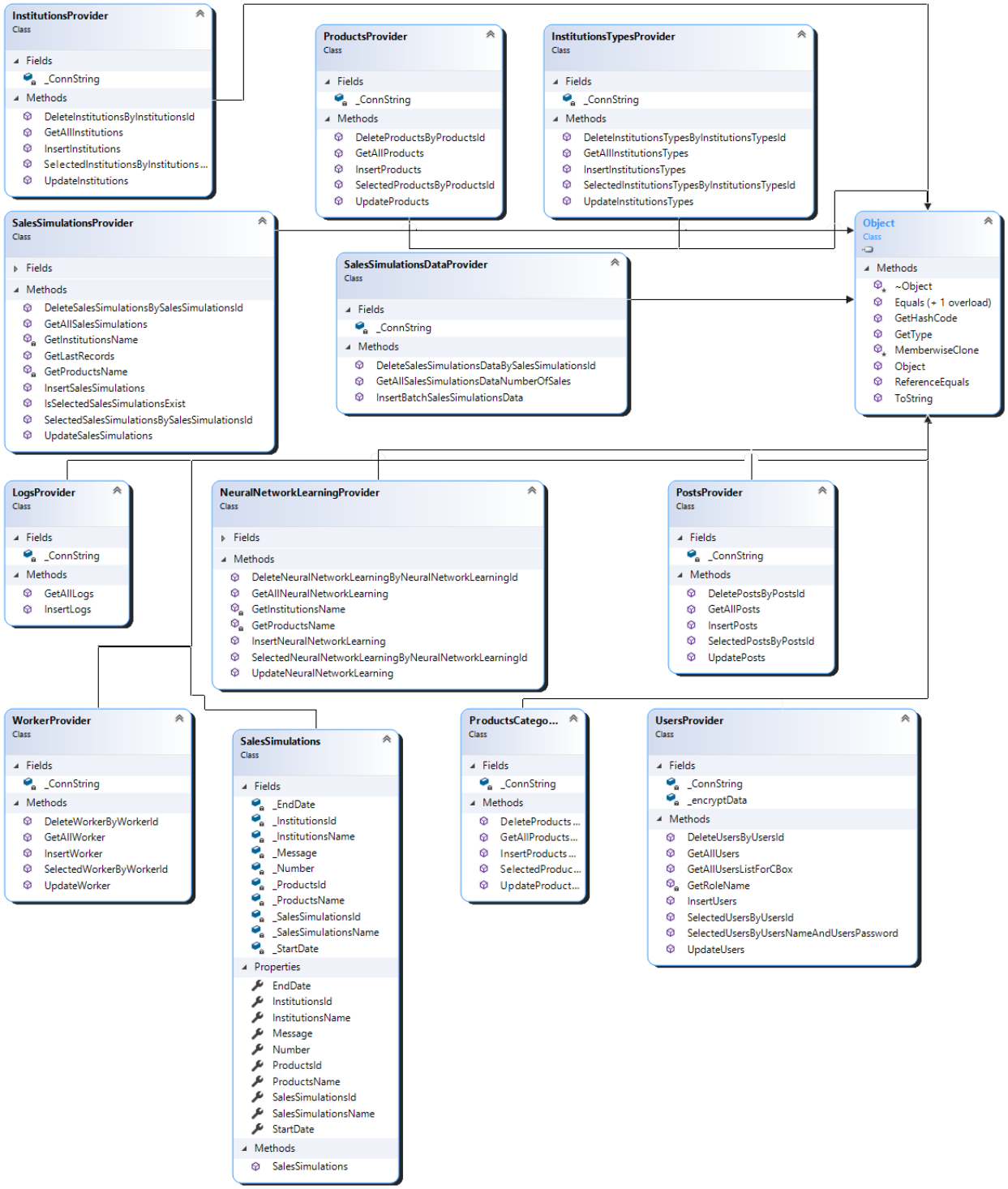


Рисунок 3.4 – Діаграма класів з методами для роботи з базою даних

3.3 Інструкція користувача

На початку необхідно запустити додаток "Система підтримки ресторанного бізнесу". Після запуску програми користувачу буде відображено

вікно, в якому буде запропоновано ввести ім'я користувача та пароль. Зображення цього вікна представлено на рис. 3.5.

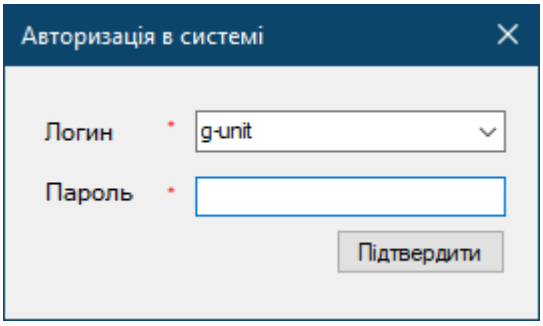


Рисунок 3.5 – Авторизація користувача

У випадку введення некоректних даних, програма надсилатиме користувачеві відповідне повідомлення, яке попереджатиме про помилку. Зображення цього повідомлення можна побачити на рис. 3.6.

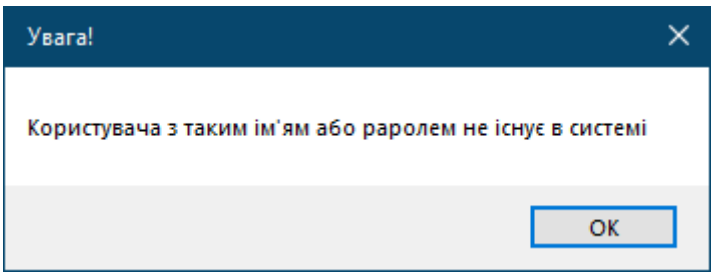


Рисунок 3.6 – Вікно попередження

Для забезпечення швидкого пошуку, користувач може обрати ім'я з випадаючого списку або вводити дані в поле вводу. Програма впорядкує список за алфавітом, і під час введення даних буде відображати вгорі списку те ім'я, яке відповідає введеним символам з клавіатури.

Після успішної ідентифікації користувача, система відкриє головне вікно програми з основним меню (рис. 3.7). Це вікно надасть користувачеві доступ до ключових функцій та опцій програми для подальшої роботи та прийняття рішень.

Рисунок 3.7 – Головне вікно

Для здійснення прогнозування попиту на продукцію в сфері бізнесу обслуговування необхідно наповнити інформаційну систему відповідною інформацією. Ця інформація включає категорії продукції, конкретні продукти, заклади обслуговування, типи закладів, співробітників та їхні посади.

Програма надає можливість редагувати та видаляти введені дані щодо категорій продукції, конкретних продуктів, закладів обслуговування, типів закладів, співробітників та їхніх посад.

Для початку процесу додавання інформації про типи закладів бізнесу обслуговування необхідно виконати наступні кроки: перейти до меню програми «Довідники» -> «Типи закладу». Після цього відкриється вікно, яке відображає вміст інформації про типи закладів, як показано на рис. 3.8.

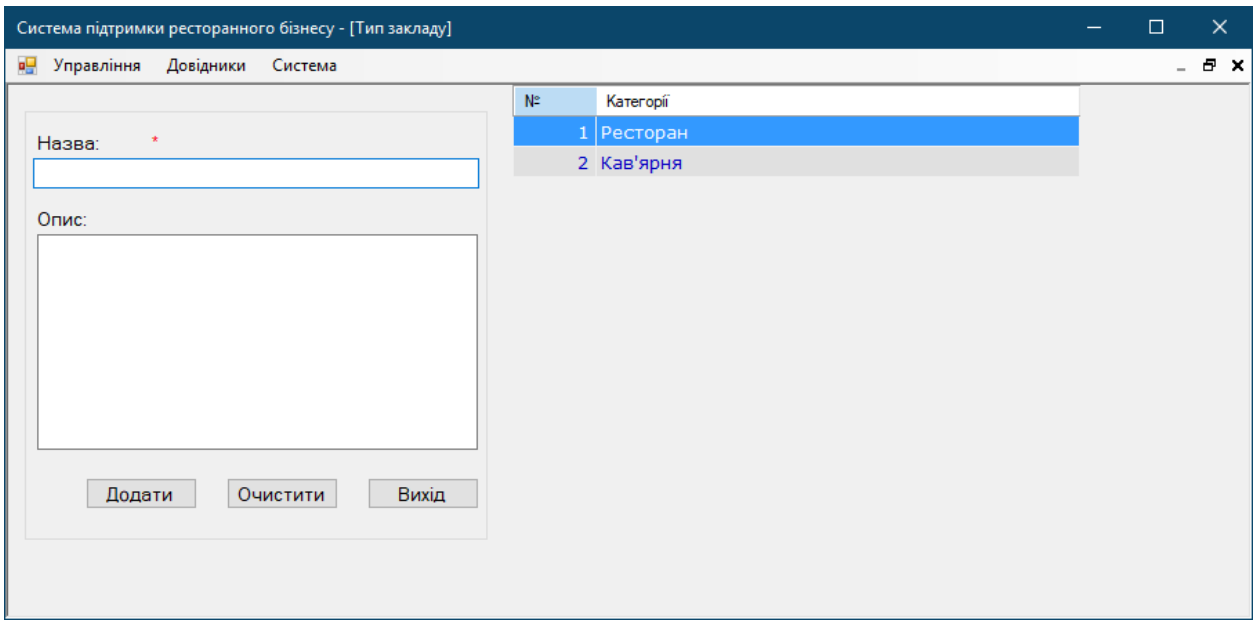


Рисунок 3.8 – Типи закладу бізнесу

У програмі також передбачена можливість редагування та видалення даних у разі потреби. Для здійснення редагування даних про співробітників використовується відповідне вікно, яке можна побачити на рис. 3.9. Це вікно надає користувачу можливість змінювати існуючі дані про співробітників та видаляти непотрібні записи з бази даних. Завдяки цій функціональності, користувач може здійснювати потрібні зміни у відомостях про співробітників, що є важливим для забезпечення актуальності та точності інформації в системі.

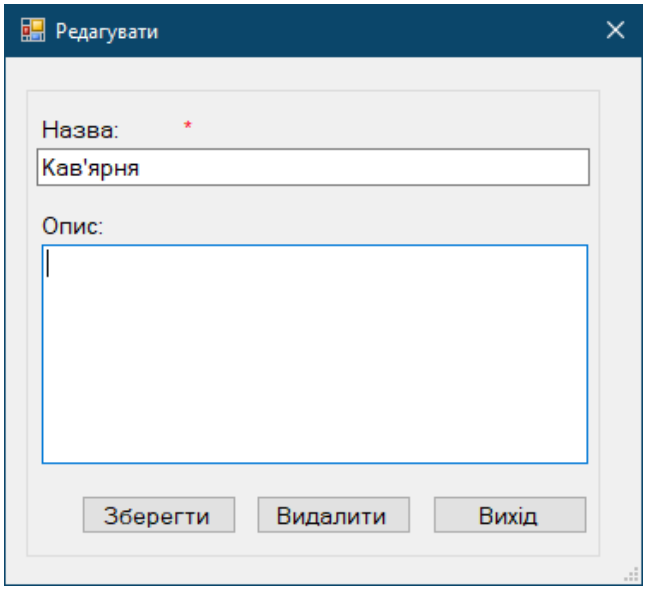


Рисунок 3.9 – Вікно редагування даних типів закладу

Для внесення інформації про заклади бізнесу обслуговування в систему

передбачено можливість, доступна через меню програми. Конкретно, користувач може перейти до розділу "Довідники" і обрати пункт "Заклади обслуговування". Це відображено на рис. 3.10, де показано відповідне вікно. У цьому вікні користувач може ввести необхідну інформацію про заклади бізнесу обслуговування, таку як їх назви, адреси, контактні дані тощо. Ця функціональність дозволяє системі мати повну та актуальну інформацію про заклади, що є важливим для подальшого проведення прогнозування попиту та прийняття рішень в сфері бізнесу обслуговування.

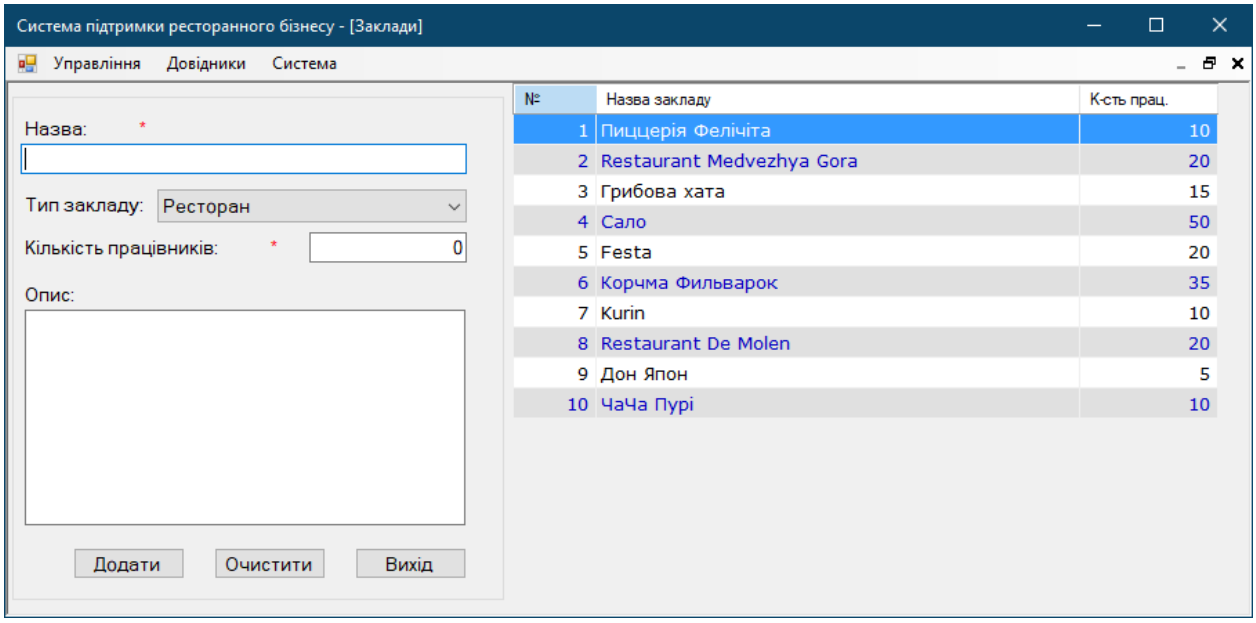


Рисунок 3.10 – Вікно опрацювання закладів

Для додавання інформації про співробітника, користувач повинен скористатися меню програми, конкретно перейти до розділу "Довідники" і обрати пункт "Співробітники". У відповідному вікні, зображеному на рис. 3.11, користувач повинен ввести всі необхідні дані про співробітника, такі як його прізвище, ім'я, посаду, контактні дані тощо. Після заповнення відповідних полів, користувач може натиснути кнопку "Додати", тоді новий співробітник буде доданий до списку всіх співробітників, що відображається у правій частині екрану. Цей процес дозволяє системі мати повну та актуальну інформацію про співробітників, що є важливим для подальшої роботи та прийняття рішень в контексті системи підтримки прийняття рішень для бізнесу обслуговування.

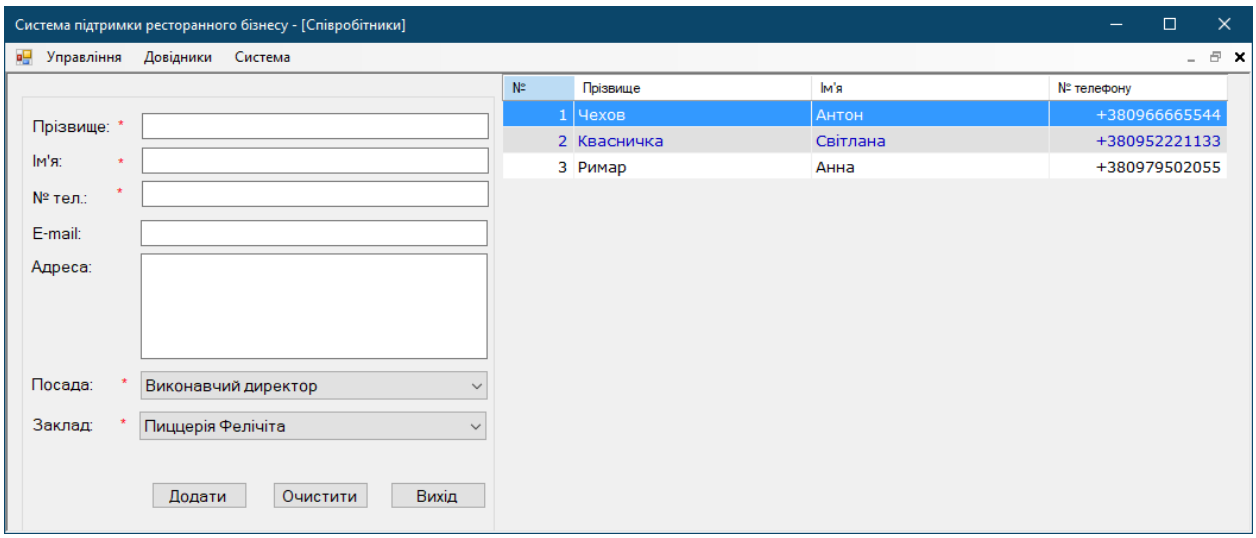


Рисунок 3.11 – Вікно опрацювання даних співробітників закладів

У системі передбачена можливість редагування та видалення даних про співробітників в разі потреби. Для цього доступне відповідне вікно редагування, яке можна відкрити за допомогою програмного інтерфейсу. На рис. 3.12 зображено зразок такого вікна для редагування даних про співробітників. Це вікно надає користувачеві можливість внести зміни в раніше введені дані про співробітника. Він може відредагувати існуючі значення атрибутів, такі як ім'я, прізвище, посада, контактні дані тощо. Крім того, користувач має можливість видалити запис про співробітника у разі потреби. Цей механізм редагування та видалення даних дозволяє забезпечити актуальність та точність інформації про співробітників у системі.

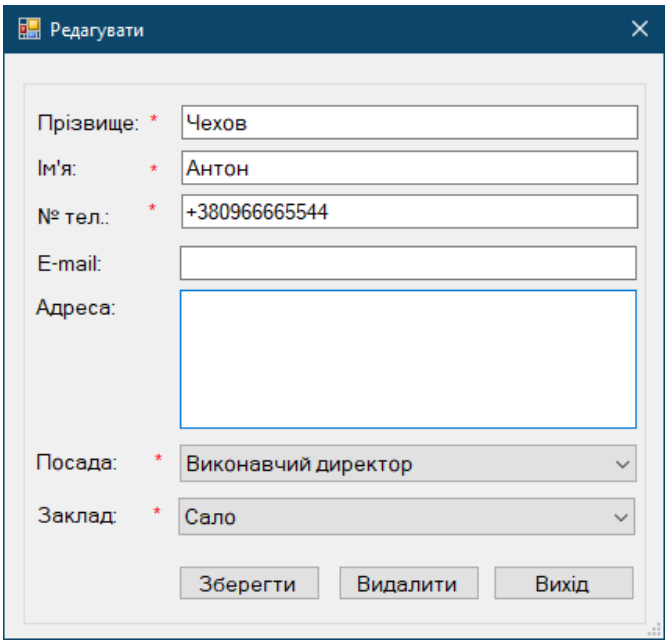


Рисунок 3.12 – Вікно редагування даних співробітників закладів

Методи обробки інформації про посади співробітників, категорії продукції та саму продукцію виконуються за аналогічним принципом. Після успішного заповнення довідників програми, користувач може використовувати її основні функціональні можливості. Зокрема, це включає проведення симуляції продажу продукції, навчання нейронної мережі на основі згенерованих даних та здійснення прогнозування.

Для проведення симуляції продажу, користувач повинен перейти до меню програми "Управління" -> "Симуляція продаж". У результаті цього відкриється вікно, зображене на рис. 3.13. Користувачу необхідно встановити параметри симуляції, такі як обрані заклад, продукція та період продажу. Після введення параметрів користувач повинен натиснути кнопку "Симулювати". Програма здійснить симуляцію продажу обраної продукції в обраному закладі на вибраний період.

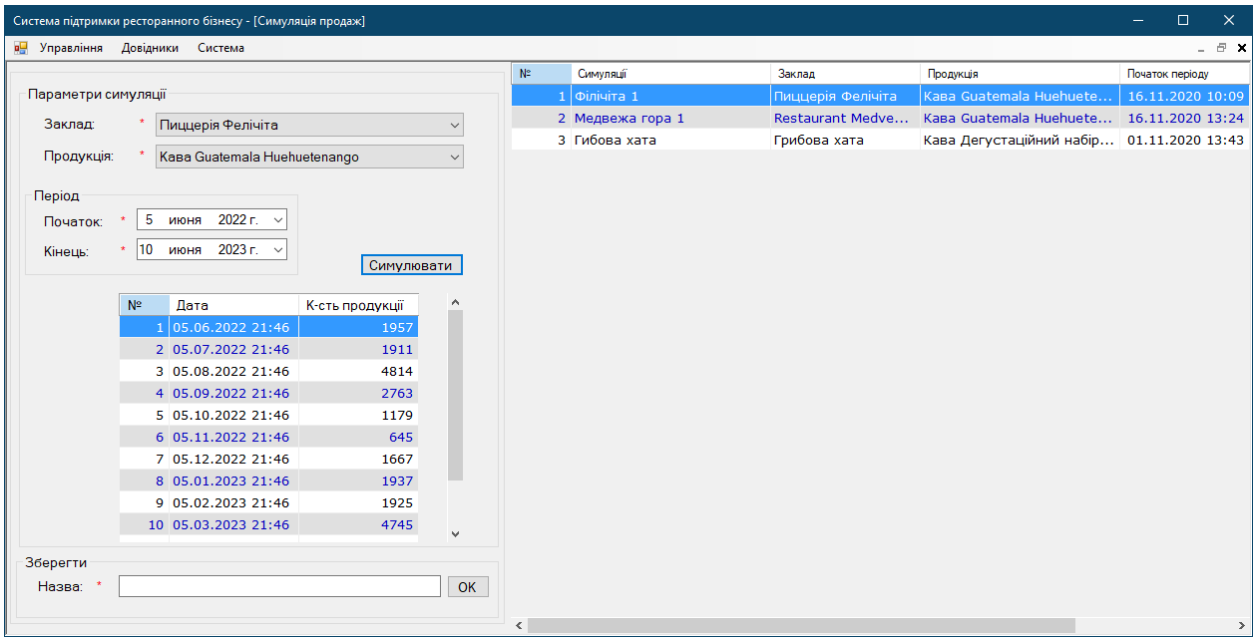


Рисунок 3.13 – Вікно для проведення симуляції пордаж

Після завершення прогнозування, користувач має можливість зберегти дані шляхом введення назви у відповідне поле для збереження.

Наступним кроком є проведення навчання нейронної мережі. Для цього користувач повинен перейти до меню програми "Управління" -> "Навчання нейронної мережі". В результаті цього дії відкриється вікно, яке зображено на

рис. 3.14. Після цього користувач повинен обрати відповідну симуляцію та натиснути кнопку "Навчати", що запустить процес навчання нейронної мережі.

Інформація про прогрес навчання нейронної мережі буде відображена відповідному вікні програми.

Користувач має можливість зберегти дані навчання нейронної мережі у базі даних. Для цього необхідно ввести назву проведеного навчання та натиснути кнопку "Ок". Після цього програма збереже всі дані, пов'язані з навчанням, у системі.

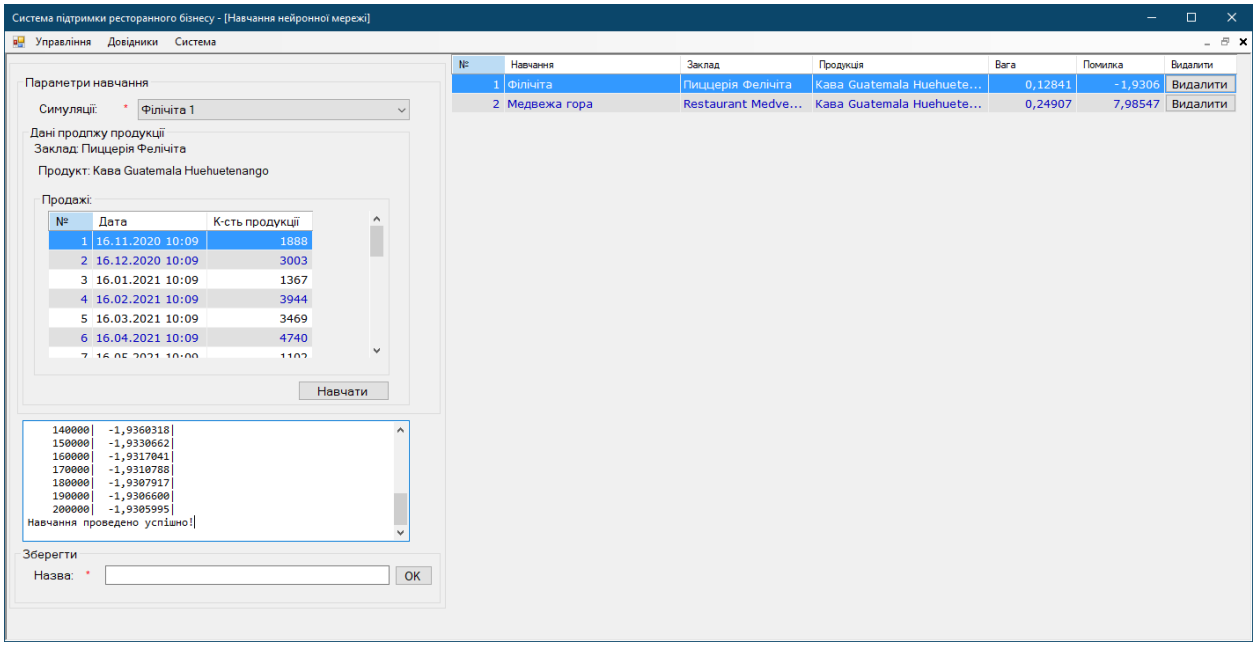


Рисунок 3.14 – Вікно навчання нейронної мережі

Після успішного збереження даних, користувач має можливість провести прогнозування попиту на певну продукцію. Це можна зробити, перейшовши до меню програми "Управління" -> "Прогнозування". В результаті цієї дії відкриється вікно, яке зображено на рис. 3.15. Для проведення прогнозування користувач повинен вибрати відповідну нейронну мережу та встановити період прогнозування.

За допомогою вибраної нейронної мережі та встановленого періоду прогнозування програма здійснить прогноз попиту на продукцію.

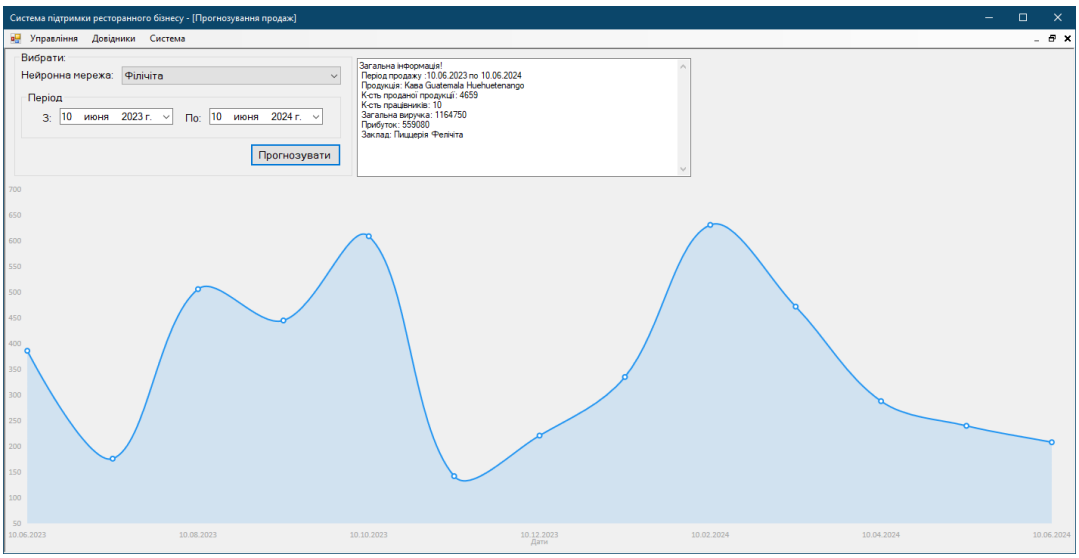


Рисунок 3.15 – Вікно для здійснення прогнозування

Після проведення прогнозування програма надає користувачу додаткову інформацію, яка включає наступне:

– період продажу, в якому здійснювався прогноз;

– назва продукції, на яку було зроблено прогноз;

– інформація про кількість проданої продукції в цьому періоді;

– інформація про кількість працівників у закладі, пов'язаному з продукцією;

– загальний обсяг виручки, отриманої в результаті продажу даної продукції;

– прогнозований прибуток, який можна очікувати в результаті продажу продукції;

– назва закладу, пов'язаного з продукцією, на який стосується прогноз.

Слід відзначити, що користувач з роллю "Системний адміністратор" має особливі привілегії щодо управління обліковими записами в системі. Він має змогу виконувати дії, такі як створення, редагування та видалення облікових записів, що підтверджено на рис. 3.16. Це надає системному адміністратору повний контроль над управлінням обліковими записами та забезпечує можливість керувати доступом користувачів до системи згідно їхніх ролей та повноважень.

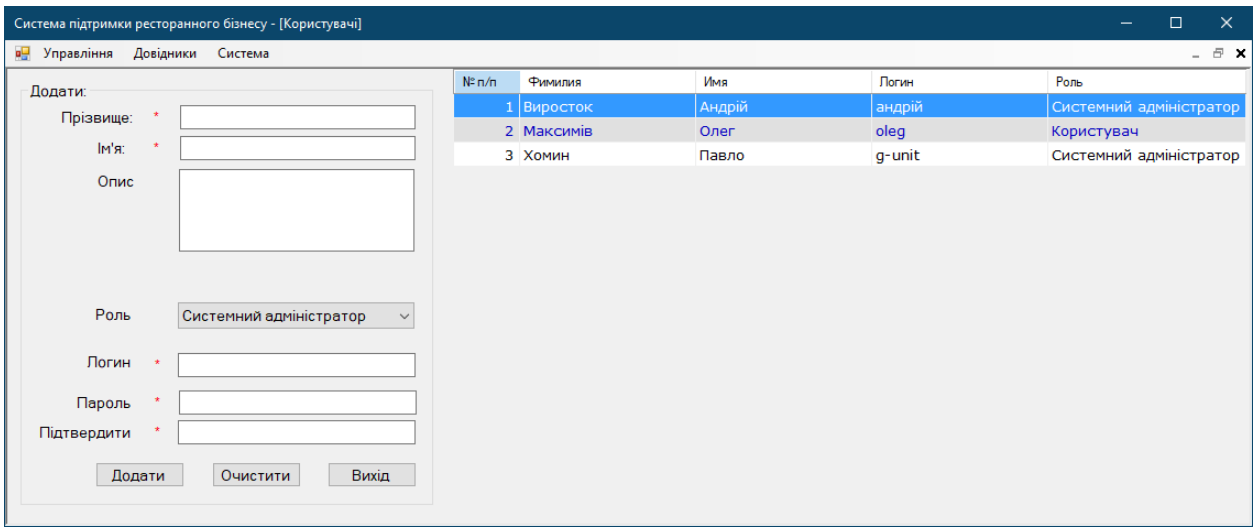


Рисунок 3.16 – Вікно для керування обліковими записами

У випадку потреби, користувачеві надається можливість змінити свої дані та пароль (див. рис. 3.17). Це дає змогу користувачам актуалізувати свою інформацію та забезпечити безпеку свого облікового запису. Зміна даних та пароля дозволяє користувачам зберігати актуальну та захищену інформацію у системі.

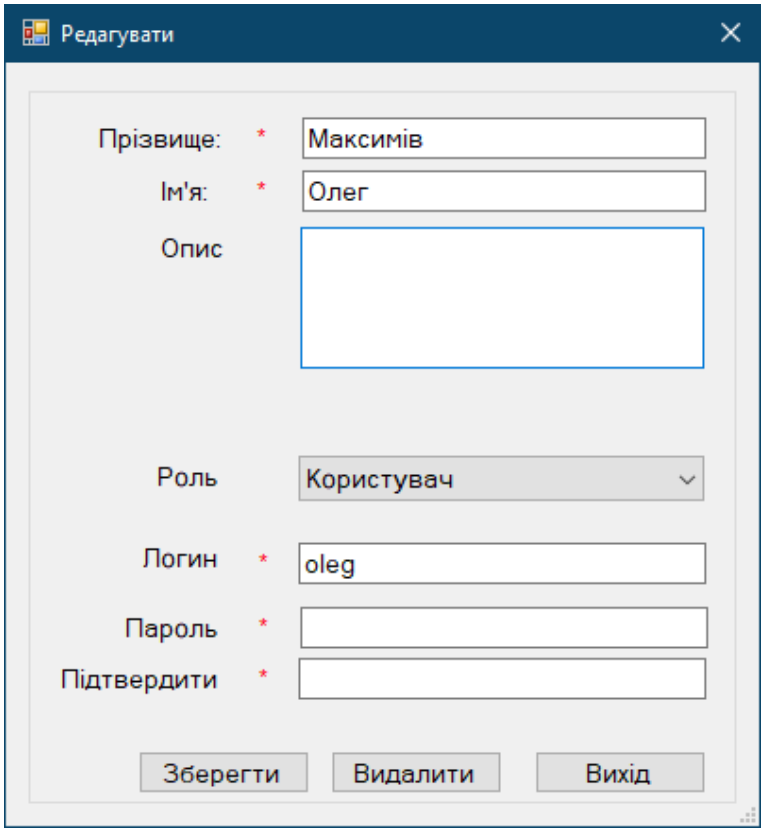
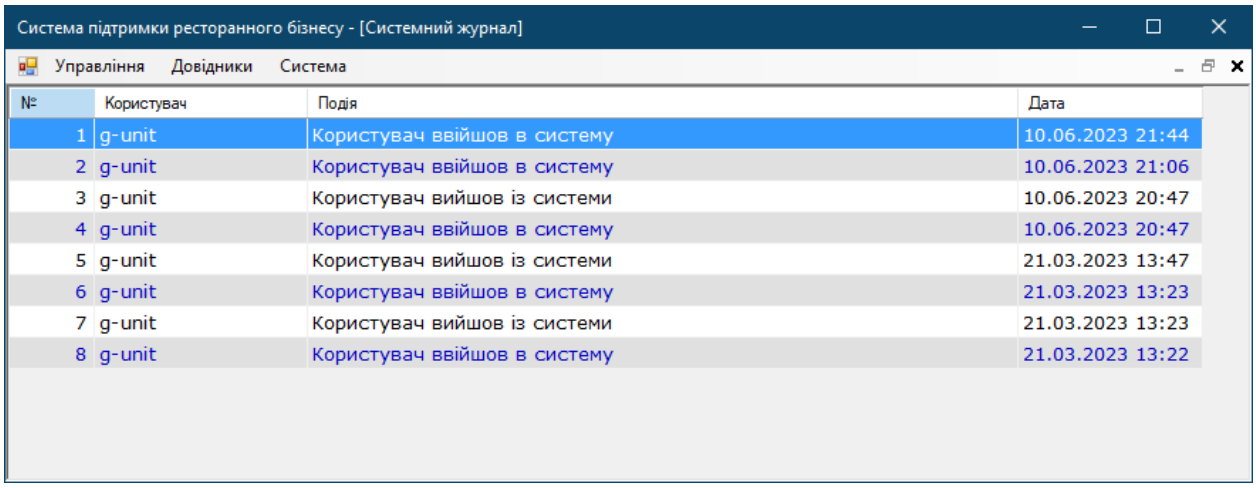


Рисунок 3.17 – Редагування даних облікового запису

Система забезпечує можливість переглядати активність користувачів та їх дії в системі. Ця функціональність доступна через обліковий запис з правами системного адміністратора. Для отримання такої інформації, користувач повинен перейти до меню "Система" та обрати опцію "Системний журнал". В цьому вікні відображаються всі події, що сталися в системі (див. рис. 3.19). Це дозволяє системним адміністраторам слідкувати за активністю користувачів та здійснювати контроль за діями, які вони виконують у системі.



№	Користувач	Подія	Дата
1	g-unit	Користувач ввійшов в систему	10.06.2023 21:44
2	g-unit	Користувач ввійшов в систему	10.06.2023 21:06
3	g-unit	Користувач вийшов із системи	10.06.2023 20:47
4	g-unit	Користувач ввійшов в систему	10.06.2023 20:47
5	g-unit	Користувач вийшов із системи	21.03.2023 13:47
6	g-unit	Користувач ввійшов в систему	21.03.2023 13:23
7	g-unit	Користувач вийшов із системи	21.03.2023 13:23
8	g-unit	Користувач ввійшов в систему	21.03.2023 13:22

Рисунок 3.18 – Вікно «Системний журнал»

Тестування програми успішно проведено та не було виявлено жодних помилок системи.

3.4 Особливості програмної реалізації алгоритму прогнозування попиту

На рис. 3.19 наведена діаграма класів, які використовуються для реалізації алгоритму прогнозування. Ця діаграма відображає взаємозв'язки та взаємодію між різними класами, необхідними для виконання процесу прогнозування. Кожен клас на діаграмі представляє окрему функціональну одиницю, а зв'язки між класами вказують на залежності та взаємодію між ними. Ця діаграма надає візуальне уявлення про структуру та організацію класів, що становлять основу алгоритму прогнозування.

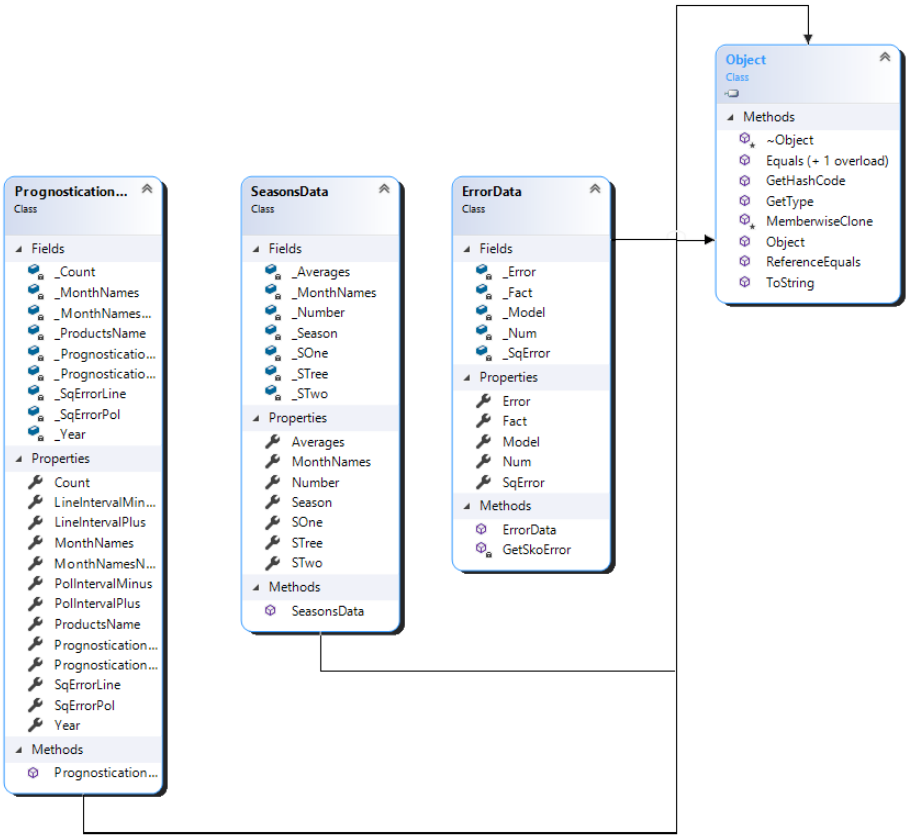


Рисунок 3.20 – Діаграма класів роботи алгоритму прогнозування

Клас ErrorData використовується для обчислення та зберігання даних з помилками прогнозування. Цей клас містить наступні публічні властивості:

– Num (тип int): Номер періоду.

– Fact (тип double): Фактичне значення продажів для даного періоду.

– Model (тип double): Розрахункове значення моделі для даного періоду.

– Error (тип double): Помилка моделі (різниця між фактичним значенням та значенням моделі).

– SqError (тип double): Середньоквадратична помилка.

Обчислення значення середньоквадратичної помилки виконується за допомогою методу, код якого наведений у лістингу 3.1. Даний метод використовує значення помилок для кожного періоду та обчислює середньоквадратичне значення на основі цих даних.

Лістинг 3.1. Код методу для розрахунку средньоквадратичної помилки для периоду

```
private double GetSkoError()
{return Math.Pow(Error, 2) / Math.Pow(Model, 2); }
```

У даному методі використовуються значення помилки (Error) та значення моделі (Model). За допомогою математичних операцій відбувається обчислення середньоквадратичної помилки шляхом піднесення значення помилки до квадрату та поділу його на квадрат значення моделі. Результат обчислення повертається як значення методу.

У класі "PrognosticationData" зберігаються результати прогнозування, включаючи різноманітну інформацію, таку як назва місяця, номер місяця, рік, назва продукції, фактична кількість продажів, прогнозовані значення за лінійною та поліноміальною моделями, а також квадратичні помилки цих моделей.

Кожна публічна властивість має свою функцію:

– "MonthName" - назва місяця;

– "MonthNumber" - номер місяця;

– "Year" – рік;

– "ProductName" - назва продукції;

– "SalesCount" - фактична кількість продажів;

– "LinearModelForecast" - прогнозоване значення за лінійною моделлю;

– "LinearModelSquaredError" - квадратична помилка лінійної моделі (статична властивість, загальна для всіх об'єктів класу);

– "PolynomialModelForecast" - прогнозоване значення за поліноміальною моделлю;

– "PolynomialModelSquaredError" - квадратична помилка поліноміальної моделі (статична властивість, загальна для всіх об'єктів класу).

У класі "PrognosticationData" доступні автоматичні властивості для розрахунку довірчих інтервалів, код яких наведений у лістингу 3.2.

Лістинг 3.2. Автовластивості класу PrognosticationData

public double LineIntervalPlus { get { return PrognosticationLine * (1 + SqErrorLine); } }

public double LineIntervalMinus { get { return PrognosticationLine * (1 - SqErrorLine); } }

public double PolIntervalPlus { get { return PrognosticationPoly * (1 + SqErrorPol); } }

public double PolIntervalMinus { get { return PrognosticationPoly * (1 - SqErrorPol); } }

Клас SeasonsData містить дані про сезонну компоненту і реалізує наступні властивості:

– Number - номер періоду;
– MonthNames - назва місяця;
– SOne - сезонна компонента першого року;
– STwo - сезонна компонента другого року;
– STree - сезонна компонента третього року;
– Averages - середнє значення сезонної компоненти за період;
– Season - обчислене значення сезонної компоненти.

Для правильного функціонування алгоритму прогнозування необхідні дані з попередніх періодів. Під час вибору товару, дані з бази даних вибираються за допомогою запиту. Результатом запиту є список сумарних продажів, згрупованих за роком та місяцем продажу.

На наступному кроці алгоритму обчислюються дані для розрахунку помилки моделі. Метод отримує список даних про продажі (salesData) та список даних про сезонну компоненту (seasonsData). Використовуючи рівняння тренда та сезонну компоненту за поточний період, обчислюються місячні значення продажів (modelSales).

Потім, за допомогою методу LINQ, розраховується середньоквадратична помилка кожної моделі. Для цього обчислюється різниця між фактичними значеннями продажу та прогнозованими значеннями

кожної моделі, підноситься до квадрата, а потім обчислюється середнє значення помилки. Результати помилок моделей зберігаються у відповідних властивостях класу.

Лістинг 3.3. Розрахунок помилки моделі

```
public void CalculateModelErrors(List<SalesData> salesData, List<SeasonsData> seasonsData) {
    // Отримання місячних значень продажів за допомогою рівняння тренда та сезонної компоненти
    var modelSales = salesData.Select(s => s.Trend + seasonsData.FirstOrDefault(se => se.MonthNames == s.MonthNames).Season).ToList();
    // Розрахунок помилки кожної моделі
    LineError = Math.Sqrt(modelSales.Select((s, i) => Math.Pow(s - PrognosticationData[i].PrognosticationLine, 2)).Average());
    PolyError = Math.Sqrt(modelSales.Select((s, i) => Math.Pow(s - PrognosticationData[i].PrognosticationPoly, 2)).Average());
    }
```

Після обчислення помилки, алгоритм реалізує прогнозування значень продажів на наступний рік. У цьому кроці, за допомогою рівняння тренда та сезонної компоненти, обчислюються прогнозні значення для кожного періоду в класі PrognosticationData. Сезонна компонента для кожного місяця отримується зі відповідного об'єкта SeasonsData.

Далі, на основі значень середньоквадратичної помилки, обчислюються довірчі інтервали для кожної моделі. Довірчі інтервали обчислюються шляхом збільшення та зменшення прогнозних значень на відсоток, що відповідає помилці моделі. Результати прогнозування та довірчі інтервали зберігаються в відповідних властивостях класу PrognosticationData.

Лістинг 3.6. Результат прогнозування

```
public void ForecastSales(List<SeasonsData> seasonsData) {
```

```
// Обчислення прогнозних значень за рівнянням тренда та сезонною компонентою
foreach (var data in PrognosticationData)     {
   var season = seasonsData.FirstOrDefault(s => s.MonthNames == data.MonthNames)?.Season ?? 0;
   data.PrognosticationLine = Trend + season;
   data.PrognosticationPoly = Polynomial + season;
}
// Обчислення довірчих інтервалів для кожної моделі
foreach (var data in PrognosticationData)     {
   data.LineIntervalPlus = data.PrognosticationLine * (1 + LineError);
   data.LineIntervalMinus = data.PrognosticationLine * (1 - LineError);
   data.PolIntervalPlus = data.PrognosticationPoly * (1 + PolyError);
   data.PolIntervalMinus = data.PrognosticationPoly * (1 - PolyError);
}}
```

3.6 Висновки до розділу

У ході розробки додатку для підтримки прийняття бізнесу обслуговування на мові C# у середовищі Visual Studio 2019 були реалізовані наступні функціональні можливості:

– керування обліковими записами користувачів, зокрема: додавання, редагування та видалення облікових записів користувачів;

– керування даними про співробітників, зокрема: додавання, редагування та видалення даних про співробітників;

– керування даними про посади співробітників, зокрема: Додавання, редагування та видалення даних про посади співробітників;

– керування даними про категорії продукції, зокрема: додавання, редагування та видалення даних про категорії продукції;

– керування даними про продукцію, зокрема: додавання, редагування та видалення даних про продукцію;

– керування даними про типи закладів, зокрема: додавання, редагування та видалення даних про типи закладів;

– керування даними про заклади, зокрема: додавання, редагування та видалення даних про заклади;

– проведення симуляції продажу для закладів бізнесу обслуговування;

– проведення навчання нейронної мережі та збереження даних проведеного навчання;

– проведення прогнозування попиту на продукцію за обраний період часу;

– фіксування активності користувачів системи та документування подій у системний журнал.

Ці функціональні вимоги були успішно реалізовані в додатку, надаючи користувачам широкий спектр можливостей для керування даними та проведення аналітичних операцій.

ВИСНОВКИ

Дана дипломна робота була спрямована на розробку інформаційної системи для підтримки прийняття рішень в галузі бізнесу обслуговування. Розробка даної системи була виконана у середовищі Microsoft Visual Studio 2019 з використанням мови програмування C# та СУБД MS Access. Основною метою цієї системи було полегшити роботу менеджерів та аналітиків шляхом надання зручного перегляду даних, можливості додавання, видалення та редагування записів, а також здійснення пошуку та фільтрації за допомогою запитів.

В рамках дипломної роботи було розглянуто основні методи прогнозування економічних процесів та проведено їх порівняльний аналіз. Для реалізації системи підтримки прийняття рішень в галузі бізнесу обслуговування були обрані відповідні засоби розробки, а також обґрунтовано вибір методу прогнозування.

На основі обраного методу був розроблений алгоритм для побудови прогнозу та оцінки його помилки. Цей алгоритм був успішно реалізований та впроваджений в інформаційну систему бізнесу обслуговування. Програма показала задовільну точність прогнозування та повністю відповідає вимогам, поставленим перед нею.

Розроблена система відповідає вимогам розробки, має інтерактивні елементи управління та інтуїтивно зрозумілий інтерфейс. Графічна оболонка має просту навігаційну структуру, що полегшує доступ до необхідної інформації.

У подальших дослідженнях можна розглянути врахування рідкісних подій, які час від часу відбуваються у світі, на рівні ймовірностей (наприклад, передсвяткові дні, введення карантину тощо). Це може значно змінити прогнозовані цифри порівняно з найкращими економічними математичними моделями.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Архіпов В. В. Організація ресторанного господарства : навч. посібник / В. В. Архіпов. – К. : Центр учбової літератури, 2007. – Режим доступу: http://nkkep.com/wp-content/uploads/2015/10/ORG-Arhy-pov.pdf

2. Шок Патті Д., Боуен Джон. Маркетинг в ресторанному бізнесі. - М.: Ресторанні відомості, 2005. - 234 с.27.

3. П'ятницька Г. Т. Формування стратегії розвитку підприємств ресторанного господарства : автореф. дис. … д-ра екон. наук : 08.00.04 / Г. Т. П'ятницька; Київ. нац. торг.-екон. ун-т. – К., 2008. – 43 с.

4. UML (Unified Modeling Language) [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/

5. Методологія IDEF0 [Електронний ресурс] – режим доступу: https://stud.com.ua/87184/ekonomika/metodologiya_idef0

6. Гірняк Л.І., Глагола В.А. Сучасний стан, перспективи та тенденції розвитку ресторанного господарства в Україні. Економіка та управління підприємствами. Випуск 16. 2018.

7. Діденко Є.О., Дідук О. А. Види конкурентних переваг підприємств ресторанного господарства та особливості управління ними. Формування ринкових відносин в Україні №12 (175). 2015р.

8. Карпенко Н.В. Управління маркетингом на торговельному підприємстві / Н.В. Карпенко, Н.І. Яловега // Науковий вісник ПУЕТ. Серія: Економічні науки. - Полтава: ПУЕТ, 2018. - №1 (86). - С. 62-67. – Режим доступу: http://journal.puet.edu.ua/index.php/nven/article/view/1440/1252

9. Нейронні мережі : теорія та практика: навч. посіб. / С. О. Субботін. – Житомир : Вид. О. О. Євенок, 2020. – 184 с. ISBN 978-966-995-189-2

10. General Regression Neural Network (GRNN) [Електронний ресурс] – режим доступу: https://minds.wisconsin.edu/bitstream/handle/1793/7779/ch2.pdf?sequence=14

11. What are Radial Basis Functions Neural Networks? [Електронний ресурс] – режим доступу: https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-are-radial-basis-functions-neural-networks

12. Руденко О.Г. Штучні нейронні мережі: Навчальний посібник / О.Г.Руденко, Є.В.Бодянський. – Харків: Сміт, 2006. – 404 с.

13. Kaggle: Your Machine Learning and Data Science Community [Електронний ресурс] – режим доступу: https://www.kaggle.com/

14. Accord.NET [Електронний ресурс] – режим доступу: http://accord-framework.net/

15. Mastering.NET Machine Learning [Електронний ресурс] – режим доступу: https://www.packtpub.com/product/mastering-net-machine-learning/9781785888403

16. Krizhevsky A., Sutskever I., Hinton G. E. Imagenet classification with deep convolutional neural networks, 2012.

17. GRNN [Електронний ресурс] – режим доступу: https://www.mathworks.com/help/deeplearning/ug/generalized-regression-neural-networks.html

18. Kent D. Lee. Data Structures and Algorithms with Python / Kent D. Lee, Steve Hubbard., – Springer, 2015. – 363 с.

19. Sun Microsystems [Електронний ресурс] – Режим доступу: https://www.britannica.com/topic/Sun-Microsystems-Inc

20. Java [Електронний ресурс] – Режим доступу: https://dou.ua/lenta/articles/how-to-learn-java/

21. C# [Електронний ресурс] – Режим доступу до ресурсу: https://beetroot.academy/blog/courses/what-is-C

22. MVC (Model-View-Controller) [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Glossary/MVC

23. SOA (Service Oriented Architecture)[Електронний ресурс] – Режим доступу: https://aws.amazon.com/ru/what-is/service-oriented-architecture/

24. Perkins B. What is ERP? Key features of top enterprise resource planning systems / Bart Perkins. – 2020. [Електронний ресурс] – Режим доступу: https://www.cio.com/article/2439502/what-is-erp-key-features-of-top-enterprise-resourceplanning-systems.html

25. Engagement Rate (ER) [Електронний ресурс] – Режим доступу: https://popsters.ru/blog/post/55

26. MS Access [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/uk-ua/microsoft-365/access

27. Creating a Business Logic Layer [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/aspnet/web-forms/overview/data-access/introduction/creating-a-business-logic-layer-cs

28. Етапи розробки користувацького інтерфейсу. URL: https://studopedia.su/12_23303_etapi-rozrobki-koristuvatskogo-interfeysuIteratsiyna-priroda-rozrobki.html (Дата звернення: 25.05.2020).

29. Windows Forms [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/help/rider/Working_with_Windows_Forms.html#editing-windows-forms

30. Creating a Data Access Layer [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/aspnet/web-forms/overview/data-access/introduction/creating-a-data-access-layer-cs

ДОДАТОК А. Лістинги програм

Лістинг 1. Код класу «NeuronsBLL»

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SystemForBusinessService.Providers;
namespace SystemForBusinessService.BLL {
  class NeuronsBLL {
    NeuralNetworkLearningProvider _NeuralNetworkLearningProvider = new
NeuralNetworkLearningProvider();
    SalesSimulationsDataProvider _SalesSimulationsDataProvider = new
SalesSimulationsDataProvider();
    List<SalesSimulationsData> _SalesSimulationsData = new List<SalesSimulationsData>();
    public List<BildGraph> GetAllGraphicsData(int NeuralNetworkLearningId, DateTime
StartDT, DateTime EndDT) {
      int i = 0;
      List<BildGraph> allBildGraph = new List<BildGraph>();
      NeuralNetworkLearning selNeuralNetworkLearning = new NeuralNetworkLearning();
      selNeuralNetworkLearning =
_NeuralNetworkLearningProvider.SelectedNeuralNetworkLearningByNeuralNetworkLearningI
d(NeuralNetworkLearningId);
      _SalesSimulationsData =
_SalesSimulationsDataProvider.GetAllSalesSimulationsDataNumberOfSales(selNeuralNetwork
Learning.SalesSimulationsId);
      Neuron neurons = new Neuron();
      neurons.LastEror = Convert.ToDecimal(selNeuralNetworkLearning.LastErors);
      neurons.weigth = Convert.ToDecimal(selNeuralNetworkLearning.Weigths);

      DateTime oneDateTime = new DateTime();
      oneDateTime = StartDT;



      while (oneDateTime < EndDT) {
        if ((i + 1) < _SalesSimulationsData.Count) {
          i++;
        } else {
          i = 0;
        }

        BildGraph oneBildGraph = new BildGraph();
        //decimal dataFromTime = Convert.ToDecimal((oneDateTime.Ticks) /
1000000000000000);
        decimal dataFromTime = Convert.ToDecimal((oneDateTime.Month *
oneDateTime.Year));
        decimal dV = _SalesSimulationsData[i].NumberOfSales;
        oneBildGraph.PeriodDate = oneDateTime;
        oneBildGraph.Value = Convert.ToInt32(neurons.ProcessInputData(dV));
        allBildGraph.Add(oneBildGraph);
```

```
        oneDateTime = oneDateTime.AddMonths(1);

      }
      return allBildGraph;
    }

  }
}


public class BildGraph {
  private DateTime _PeriodDate;
  private int _Value;
  public BildGraph() {
    _PeriodDate = new DateTime();
    _Value = 0;
  }
  public DateTime PeriodDate {
    set { _PeriodDate = value; }
    get { return _PeriodDate; }
  }
  public int Value {
    set { _Value = value; }
    get { return _Value; }
  }
}

public class Neuron {
  public decimal weigth = 0.5m;
  public decimal LastEror { get; set; }
  public decimal Smoothing { get; set; } = 0.00001m;
  public decimal ProcessInputData(decimal input) {
    return input * weigth;
  }
  public decimal RestoreInputData(decimal output) {
    return output / weigth;
  }
  public void Train(decimal input, decimal expectedResult) {
    //actualResult  - актуальний результат
    var actualResult = input * weigth;
    //Обчислюємо помилку
    LastEror = expectedResult - actualResult;
    //корекція на помилку
    var correction = (LastEror / actualResult) * Smoothing;
    //корекцію додаємо до ваги нейрона
    weigth += correction;
  }
}


public class PrognosticationData {
  private string _MonthName; //місяця;
```

```
private int _MonthNames; //номер місяця
private int _Year; //рік
private string _ProductsName; // назва товару
private int _Count; //продажів за місяць
private double _PrognosticationLine; //прогнозне значення
private double _PrognosticationPoly; //прогнозне значення поліноміальної моделі
private double _SqErrorLine; //квадратична помилка
private double _SqErrorPol; //квадратична помилка поліноміальної моделі
public PrognosticationData() {
  _MonthName = String.Empty;
  _MonthNames = 0;
  _ProductsName = String.Empty;
  _Count = 0;
  _PrognosticationLine = 0.0;
  _PrognosticationPoly = 0.0;
  _SqErrorLine = 0.0;
  _SqErrorPol = 0.0;
}

public string MonthName {
  set { _MonthName = value; }
  get { return _MonthName; }
}
public int MonthNames {
  set { _MonthNames = value; }
  get { return _MonthNames; }
}
public int Year {
  set { _Year = value; }
  get { return _Year; }
}
public string ProductsName {
  set { _ProductsName = value; }
  get { return _ProductsName; }
}
public int Count {
  set { _Count = value; }
  get { return _Count; }
}
public double PrognosticationLine {
  set { _PrognosticationLine = value; }
  get { return _PrognosticationLine; }
}
public double PrognosticationPoly {
  set { _PrognosticationPoly = value; }
  get { return _PrognosticationPoly; }
}
public double SqErrorLine {
  set { _SqErrorLine = value; }
  get { return _SqErrorLine; }
}
public double SqErrorPol {
```

```
    set { _SqErrorPol = value; }
    get { return _SqErrorPol; }
  }

  public double LineIntervalPlus { get { return PrognosticationLine * (1 + SqErrorLine); } }
  public double LineIntervalMinus { get { return PrognosticationLine * (1 - SqErrorLine); } }
  public double PolIntervalPlus { get { return PrognosticationPoly * (1 + SqErrorPol); } }
  public double PolIntervalMinus { get { return PrognosticationPoly * (1 - SqErrorPol); } }

}

public class ErrorData {
  private int _Num; //номер періоду
  private double _Fact; //продаж за даний період
  private double _Model; //значення моделі за  період
  private double _Error; //помилка моделі(фактичне значення мінус значення моделі)
  private double _SqError; //середньоквадратична помилка
  public ErrorData() {
    _Num = 0;
    _Fact = 0.0;
    _Model = 0.0;
    _Error = 0.0;
    _SqError = 0.0;
  }

  public int Num {
    set { _Num = value; }
    get { return _Num; }
  }
  public double Fact {
    set { _Fact = value; }
    get { return _Fact; }
  }
  public double Model {
    set { _Model = value; }
    get { return _Model; }
  }
  public double Error {
    set { _Error = value; }
    get { return _Error; }
  }
  public double SqError {
    set { _SqError = value; }
    get { return _SqError; }
  }
  //(факт-модель)^2/модель^2
  private double GetSkoError() {
    return Math.Pow(Error, 2) / Math.Pow(Model, 2);
  }
}

public class SeasonsData {
```

```
  private int _Number;
  private string _MonthNames;
  private double _SOne;
  private double _STwo;
  private double _STree;
  private double _Averages;
  private double _Season;
  public SeasonsData() {
    _Number = 0;
    _MonthNames = String.Empty;
    _SOne = 0.0;
    _STwo = 0.0;
    _STree = 0.0;
    _Averages = 0.0;
    _Season = 0.0;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public string MonthNames {
    set { _MonthNames = value; }
    get { return _MonthNames; }
  }
  public double SOne {
    set { _SOne = value; }
    get { return _SOne; }
  }
  public double STwo {
    set { _STwo = value; }
    get { return _STwo; }
  }
  public double STree {
    set { _STree = value; }
    get { return _STree; }
  }
  public double Averages {
    set { _Averages = value; }
    get { return _Averages; }
  }
  public double Season {
    set { _Season = value; }
    get { return _Season; }
  }

}
```

Лістинг 2. Код класу «NeuralNetworkLearningForm»

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
```

```
using SystemForBusinessService.AppCode;
using SystemForBusinessService.Providers;

namespace SystemForBusinessService.Forms.Control {
  public partial class NeuralNetworkLearningForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private SalesSimulationsProvider _SalesSimulationsPrivider = new
SalesSimulationsProvider();
    private List<SalesSimulations> _SalesSimulationsList = new List<SalesSimulations>();
    private InstitutionsProvider _InstitutionsProvider = new InstitutionsProvider();
    private Institutions _SelectedInstitution = new Institutions();
    private ProductsProvider _ProductsProvider = new ProductsProvider();
    private Products _SelectedProduct= new Products();
    private bool _IsSalesSimulationsLoad = false;

    private SalesSimulationsDataProvider _SalesSimulationsDataDataProvider = new
SalesSimulationsDataProvider();
    private List<SalesSimulationsData> _SalesSimulationsDataList = new
List<SalesSimulationsData>();
    private NeuralNetworkLearningProvider _NeuralNetworkLearningProvider = new
NeuralNetworkLearningProvider();
    private List<NeuralNetworkLearning> _NeuralNetworkLearningList = new
List<NeuralNetworkLearning>();
    private Neuron _Neurons = new Neuron();
    private bool _IsLeatn = false;

    public NeuralNetworkLearningForm() {
      InitializeComponent();
      LoadAllDate();
      DataLoad();
    }

    private void LearnBtn_Click(object sender, EventArgs e) {
      _IsLeatn = true;
      int i = 0;
      int j = 0;
      long unixSeconds = DateTimeOffset.Now.ToUnixTimeSeconds();
      StatusTreinTBox.Text = String.Format("{0,10}|{1, 12}|\r\n", "К-сть епох", "Помилка");
      do {
        i++;
        if (j + 1 < _SalesSimulationsDataList.Count) {
          j++;
        } else {
          j = 0;
        }
        decimal dataFromTime =
Convert.ToDecimal((_SalesSimulationsDataList[0].DateOfSales.Month *
_SalesSimulationsDataList[0].DateOfSales.Year));
        //decimal oneValue =
(_SalesSimulationsDataList[0].DateOfSales.Ticks).ToString().Length;
```

```
        _Neurons.Train(dataFromTime, _SalesSimulationsDataList[j].NumberOfSales);
        if (i % 10000 == 0) {
          string line = String.Format("{0,10}|{1, 12}|\r\n", i, Math.Round((_Neurons.LastEror/500),
7));
          this.BeginInvoke(new LineReceivedEvent(LineReceived), line);
        }
      } while (i<200000);
      System.Threading.Thread.Sleep(1500);
      this.BeginInvoke(new LineReceivedEvent(LineReceived), "Навчання проведено
успішно!");
      _Neurons.LastEror = _Neurons.LastEror/500;

      //_IsLeatn = true;
      //int i = 0;
      //int j = 0;
      //long unixSeconds = DateTimeOffset.Now.ToUnixTimeSeconds();
      //StatusTreinTBox.Text = String.Format("{0,10}|{1, 12}|\r\n", "К-сть епох", "Помилка");
      //do {
      //  i++;
      //  if (j+1 < _SalesSimulationsDataList.Count) {
      //    j++;
      //  } else {
      //    j = 0;
      //  }
      //  decimal dataFromTime =
Convert.ToDecimal((_SalesSimulationsDataList[0].DateOfSales.Ticks)/ 1000000000000000);
      //  //decimal oneValue =
(_SalesSimulationsDataList[0].DateOfSales.Ticks).ToString().Length;

      //  _Neurons.Train(dataFromTime, _SalesSimulationsDataList[0].NumberOfSales);
      //  if (i % 10000 == 0) {
      //    string line = String.Format("{0,10}|{1, 12}|\r\n", i, Math.Round(_Neurons.LastEror, 5));
      //    this.BeginInvoke(new LineReceivedEvent(LineReceived), line);
      //  }
      //} while (_Neurons.LastEror > _Neurons.Smoothing || _Neurons.LastEror < -
_Neurons.Smoothing);
    }

    private void LineReceived(string line) {
      StatusTreinTBox.Text += line;
      StatusTreinTBox.Focus();
      StatusTreinTBox.SelectionStart = StatusTreinTBox.Text.Length;
      StatusTreinTBox.ScrollToCaret();
    }
    private delegate void LineReceivedEvent(string line);

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {

_NeuralNetworkLearningProvider.InsertNeuralNetworkLearning(NeuralNetworkLearningName
TBox.Text, StatusTreinTBox.Text,
        _SelectedInstitution.InstitutionsId, _SelectedProduct.ProductsId,
```

```
Convert.ToInt32(SalesSimulationsCBox.SelectedValue),
      _Neurons.weigth, _Neurons.LastEror);
       DataLoad();
     }
   }

   private void LoadAllDate() {
     _SalesSimulationsList = _SalesSimulationsPrivider.GetAllSalesSimulations();
     SalesSimulationsCBox.DataSource = _SalesSimulationsList;
     SalesSimulationsCBox.ValueMember = "SalesSimulationsId";
     SalesSimulationsCBox.DisplayMember = "SalesSimulationsName";
     _IsSalesSimulationsLoad = true;
     SalesSimulationsCBox_SelectedValueChanged(SalesSimulationsCBox, EventArgs.Empty);

   }

   private void DataLoad() {
     int firstRowIndex = 0;
     if (NeuralNetworkLearningGridView.FirstDisplayedScrollingRowIndex > 0) {
       firstRowIndex = NeuralNetworkLearningGridView.FirstDisplayedScrollingRowIndex;
     }
     try {
       _NeuralNetworkLearningList =
_NeuralNetworkLearningProvider.GetAllNeuralNetworkLearning();
       LoadDataInSalesSimulationsGridView(_NeuralNetworkLearningList);
       if (_selectedRowIndex == NeuralNetworkLearningGridView.Rows.Count) {
         _selectedRowIndex = NeuralNetworkLearningGridView.Rows.Count - 1;
       }
       if (_selectedRowIndex >= 0) {
         NeuralNetworkLearningGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
         NeuralNetworkLearningGridView.Rows[_selectedRowIndex].Selected = true;
       }
     } catch { }
   }

   private void LoadDataInSalesSimulationsGridView(List<NeuralNetworkLearning>
NeuralNetworkLearningList) {
     NeuralNetworkLearningGridView.DataSource = null;
     NeuralNetworkLearningGridView.Columns.Clear();
     NeuralNetworkLearningGridView.AutoGenerateColumns = false;
     NeuralNetworkLearningGridView.RowHeadersVisible = false;

     NeuralNetworkLearningGridView.DataSource = NeuralNetworkLearningList;

     if (NeuralNetworkLearningList.Count > 0) {
       if (NeuralNetworkLearningList[0].Message ==
NamesMy.NoDataNames.NoDataInNeuralNetworkLearning) {
         DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
         messageColumn.DataPropertyName = "Message";
         messageColumn.Width = NeuralNetworkLearningGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
         NeuralNetworkLearningGridView.Columns.Add(messageColumn);
```

```
        } else {
            DataGridViewColumn SalesSimulationsDataIdColumn = new
DataGridViewTextBoxColumn();
            SalesSimulationsDataIdColumn.DataPropertyName = "NeuralNetworkLearningId";
            NeuralNetworkLearningGridView.Columns.Add(SalesSimulationsDataIdColumn);
            NeuralNetworkLearningGridView.Columns[0].Visible = false;

            DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
            numberColumn.HeaderText = "№ ";
            numberColumn.DataPropertyName = "Number";
            numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
            numberColumn.Width = NamesMy.SizeOptins.NumberSize;
            NeuralNetworkLearningGridView.Columns.Add(numberColumn);

            DataGridViewColumn SalesSimulationsNameColumn = new
DataGridViewTextBoxColumn();
            SalesSimulationsNameColumn.HeaderText = "Навчання";
            SalesSimulationsNameColumn.DataPropertyName = "NeuralNetworkLearningName";
            SalesSimulationsNameColumn.Width = 200;
            NeuralNetworkLearningGridView.Columns.Add(SalesSimulationsNameColumn);

            DataGridViewColumn InstitutionsNameColumn = new DataGridViewTextBoxColumn();
            InstitutionsNameColumn.HeaderText = "Заклад";
            InstitutionsNameColumn.DataPropertyName = "InstitutionsName";
            InstitutionsNameColumn.Width = 150;
            NeuralNetworkLearningGridView.Columns.Add(InstitutionsNameColumn);

            DataGridViewColumn ProductsNameColumn = new DataGridViewTextBoxColumn();
            ProductsNameColumn.HeaderText = "Продукція";
            ProductsNameColumn.DataPropertyName = "ProductsName";
            ProductsNameColumn.Width = 200;
            NeuralNetworkLearningGridView.Columns.Add(ProductsNameColumn);

            DataGridViewColumn StartDateColumn = new DataGridViewTextBoxColumn();
            StartDateColumn.HeaderText = "Вага";
            StartDateColumn.DataPropertyName = "Weigths";
            StartDateColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
            StartDateColumn.Width = 100;
            NeuralNetworkLearningGridView.Columns.Add(StartDateColumn);

            DataGridViewColumn EndDateColumn = new DataGridViewTextBoxColumn();
            EndDateColumn.HeaderText = "Помилка";
            EndDateColumn.DataPropertyName = "LastErors";
            EndDateColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
            EndDateColumn.Width = 100;
            NeuralNetworkLearningGridView.Columns.Add(EndDateColumn);


            DataGridViewButtonColumn deleteBtn = new DataGridViewButtonColumn();
```

```
        deleteBtn.HeaderText = NamesMy.ProgramButtons.DeleteBtn;
        deleteBtn.Text = NamesMy.ProgramButtons.DeleteBtn;
        deleteBtn.UseColumnTextForButtonValue = true;
        deleteBtn.ToolTipText = NamesMy.ProgramButtons.DeleteBtn;
        deleteBtn.Width = NamesMy.SizeOptins.DeleteBtnSize;
        NeuralNetworkLearningGridView.Columns.Add(deleteBtn);
      }
      for (int i = 0; i < NeuralNetworkLearningGridView.Columns.Count; i++) {
        NeuralNetworkLearningGridView.Columns[i].HeaderCell.Style.BackColor =
Color.LightGray;
      }
    }
  }

  private void LoadSalesSimulationsData(List<SalesSimulationsData>
SalesSimulationsDataList) {
    SalesSimulationsDataGridView.DataSource = null;
    SalesSimulationsDataGridView.Columns.Clear();
    SalesSimulationsDataGridView.AutoGenerateColumns = false;
    SalesSimulationsDataGridView.RowHeadersVisible = false;

    SalesSimulationsDataGridView.DataSource = SalesSimulationsDataList;

    DataGridViewColumn NameColumn = new DataGridViewTextBoxColumn();
    NameColumn.HeaderText = "№";
    NameColumn.DataPropertyName = "Number";
    NameColumn.Width = 50;
    NameColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    SalesSimulationsDataGridView.Columns.Add(NameColumn);

    DataGridViewColumn DateOfSalesColumn = new DataGridViewTextBoxColumn();
    DateOfSalesColumn.HeaderText = "Дата";
    DateOfSalesColumn.DataPropertyName = "DateOfSales";
    DateOfSalesColumn.Width = 130;
    SalesSimulationsDataGridView.Columns.Add(DateOfSalesColumn);

    DataGridViewColumn NumberOfSalesColumn = new DataGridViewTextBoxColumn();
    NumberOfSalesColumn.HeaderText = "К-сть продукції";
    NumberOfSalesColumn.DataPropertyName = "NumberOfSales";
    NumberOfSalesColumn.Width = 120;
    NumberOfSalesColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
    SalesSimulationsDataGridView.Columns.Add(NumberOfSalesColumn);

    for (int i = 0; i < SalesSimulationsDataGridView.Columns.Count; i++) {
      SalesSimulationsDataGridView.Columns[i].HeaderCell.Style.BackColor =
Color.LightGray;
    }
  }

  private bool IsDataEnteringCorrect() {
    bool isCorrect = true;
```

```
      if (_validation.IsDataEntering(NeuralNetworkLearningNameTBox.Text)) {
        NeuralNetworkLearningNameValiadtionLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        NeuralNetworkLearningNameValiadtionLbl.Text =
NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (!_IsLeatn) {
        MessageBox.Show("Неможливо зберегти, навчання не було проведене!", "Увага!");
        isCorrect = false;
      }
      return isCorrect;
    }

    private void SalesSimulationsCBox_SelectedValueChanged(object sender, EventArgs e) {
      if (_IsSalesSimulationsLoad) {
        _SalesSimulationsDataList =
_SalesSimulationsDataDataProvider.GetAllSalesSimulationsDataNumberOfSales(Convert.ToInt
32(SalesSimulationsCBox.SelectedValue));
        LoadSalesSimulationsData(_SalesSimulationsDataList);
        SalesSimulations selSalesSimulations = new SalesSimulations();
        selSalesSimulations =
_SalesSimulationsPrivider.SelectedSalesSimulationsBySalesSimulationsId(Convert.ToInt32(Sal
esSimulationsCBox.SelectedValue));
        _SelectedInstitution =
_InstitutionsProvider.SelectedInstitutionsByInstitutionsId(selSalesSimulations.InstitutionsId);
        InstitutionLbl.Text = "Заклад: " + _SelectedInstitution.InstitutionsName;
        _SelectedProduct =
_ProductsProvider.SelectedProductsByProductsId(selSalesSimulations.ProductsId);
        ProductLbl.Text = "Продукт: " + _SelectedProduct.ProductsName;
        _IsLeatn = false;
      }
    }

    private void NeuralNetworkLearningGridView_CellClick(object sender,
DataGridViewCellEventArgs e) {
      if (e.ColumnIndex == 7) {
        if (MessageBox.Show("Ви дійсно бажаєте видалити це заняття?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
          int selectedNeuralNetworkLearningId =
Convert.ToInt32(NeuralNetworkLearningGridView[0, e.RowIndex].Value.ToString());

_NeuralNetworkLearningProvider.DeleteNeuralNetworkLearningByNeuralNetworkLearningId(
selectedNeuralNetworkLearningId);
          DataLoad();
        }
      }
    }
  }
}
```

Лістинг 3. Код класу «PrognosticationsForm »

```
using LiveCharts;
using LiveCharts.Wpf;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using SystemForBusinessService.BLL;
using SystemForBusinessService.Providers;

namespace SystemForBusinessService.Forms.Control {
  public partial class PrognosticationsForm : Form {
    NeuronsBLL _NeuronsBLL = new NeuronsBLL();
    NeuralNetworkLearningProvider _NeuralNetworkLearningProvider = new
NeuralNetworkLearningProvider();
    NeuralNetworkLearning _SelectNeuralNetworkLearning = new NeuralNetworkLearning();
    ProductsProvider _ProductsProvider = new ProductsProvider();
    Products _SelectedProducts = new Products();
    InstitutionsProvider _InstitutionsProvider = new InstitutionsProvider();
    Institutions _SelectedInstitutions = new Institutions();
    List<NeuralNetworkLearning> _NeuralNetworkLearningList = new
List<NeuralNetworkLearning>();
    List<BildGraph> _BildGraphList = new List<BildGraph>();
    public PrognosticationsForm() {
      InitializeComponent();
      GetAllDate();
    }

    private void GetAllDate() {
      EndDTP.Value = EndDTP.Value.AddYears(1);
      _NeuralNetworkLearningList =
_NeuralNetworkLearningProvider.GetAllNeuralNetworkLearning();
      NeuralNetworkLearningCBox.DataSource = _NeuralNetworkLearningList;
      NeuralNetworkLearningCBox.ValueMember = "NeuralNetworkLearningId";
      NeuralNetworkLearningCBox.DisplayMember = "NeuralNetworkLearningName";

    }


    private void PrognosticationsBtn_Click(object sender, EventArgs e) {
      _SelectNeuralNetworkLearning =
_NeuralNetworkLearningProvider.SelectedNeuralNetworkLearningByNeuralNetworkLearningI
d(Convert.ToInt32(NeuralNetworkLearningCBox.SelectedValue));
      _SelectedProducts =
_ProductsProvider.SelectedProductsByProductsId(_SelectNeuralNetworkLearning.ProductsId);
      _SelectedInstitutions =
_InstitutionsProvider.SelectedInstitutionsByInstitutionsId(_SelectNeuralNetworkLearning.Instit
utionsId);
```

```
      BildGraphics();
      GetAllRaport();
    }

    private void BildGraphics() {
      GraphicsCC.AxisX.Clear();
      DateTime startDT = new DateTime(StartDTP.Value.Year, StartDTP.Value.Month,
StartDTP.Value.Day, 0, 0, 0);
      DateTime endDT = new DateTime(EndDTP.Value.Year, EndDTP.Value.Month,
EndDTP.Value.Day, 23, 59, 59);
      _BildGraphList =
_NeuronsBLL.GetAllGraphicsData(Convert.ToInt32(NeuralNetworkLearningCBox.SelectedVal
ue), startDT, endDT);

      SeriesCollection series = new SeriesCollection();
      ChartValues<double> sellingValue = new ChartValues<double>();
      List<string> dates = new List<string>();
      for (int i = 0; i < _BildGraphList.Count; i++) {
        sellingValue.Add(_BildGraphList[i].Value);
        dates.Add(_BildGraphList[i].PeriodDate.ToShortDateString());
      }


      GraphicsCC.AxisX.Add(new LiveCharts.Wpf.Axis() {
        Title = "Дати",
        Labels = dates
      });


      LineSeries sellingLine = new LineSeries();
      sellingLine.Title = "Продаж";
      sellingLine.Values = sellingValue;

      series.Add(sellingLine);
      GraphicsCC.Series = series;
    }

    private void GetAllRaport() {
      RaportTBox.Text = "Загальна інформація!\r\n ";
      RaportTBox.Text += "Період продажу :" + StartDTP.Value.ToShortDateString() + " по " +
EndDTP.Value.ToShortDateString() +  "\r\n ";
      RaportTBox.Text += "Продукція: " + _SelectedProducts.ProductsName  + "\r\n ";
      RaportTBox.Text += "К-сть проданої продукції: " + GetAllSumProd() + "\r\n ";
      RaportTBox.Text += "К-сть працівників: " + _SelectedInstitutions.NumberOfEmployees +
"\r\n ";
      RaportTBox.Text += "Загальна виручка: " +  GetAllProfit(GetAllSumProd(),
_SelectedProducts.SellingPrice) + "\r\n ";
      RaportTBox.Text += "Прибуток: " + GetProfit(GetAllSumProd(),
_SelectedProducts.SellingPrice, _SelectedProducts.ManufacturingPrice,
_SelectedProducts.DiscountOnProducts) +  "\r\n ";
      RaportTBox.Text += "Заклад: " + _SelectedInstitutions.InstitutionsName + "\r\n ";
    }
```

```
    public double GetProfit(int SumProd, double SellPrice, double ManufacturingPrice, double
DiscountOnProducts) {
      double profPrice = SellPrice - ManufacturingPrice - DiscountOnProducts;
      return SumProd * profPrice;
    }


    public double GetAllProfit(int SumProd, double Price) {
      return SumProd * Price;
    }

    private int GetAllSumProd() {
      int sum = 0;
      for (int i = 0; i < _BildGraphList.Count; i++) {
        sum += _BildGraphList[i].Value;
      }
      return sum;
    }
  }
}
```

Лістинг 4. Код класу « SalesSimulationsForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using SystemForBusinessService.AppCode;
using SystemForBusinessService.Providers;

namespace SystemForBusinessService.Forms.Control {
  public partial class SalesSimulationsForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private SalesSimulationsProvider _SalesSimulationsPrivider = new
SalesSimulationsProvider();
    private List<SalesSimulations> _SalesSimulationsList = new List<SalesSimulations>();
    private InstitutionsProvider _InstitutionsProvider = new InstitutionsProvider();
    private List<Institutions> _InstitutionsList = new List<Institutions>();
    private ProductsProvider _ProductsProvider = new ProductsProvider();
    private List<Products> _ProductsList = new List<Products>();
    private bool _IsInstitutionsLoad = false;
    private Institutions _SelectedInstitutions = new Institutions();

    private SalesSimulationsDataProvider _SalesSimulationsDataDataProvider = new
SalesSimulationsDataProvider();
    private List<SalesSimulationsData> _SalesSimulationsDataList = new
List<SalesSimulationsData>();
```

```
    public SalesSimulationsForm() {
      InitializeComponent();
      LoadAllDate();
      DataLoad();
    }

    private void SimulateBtn_Click(object sender, EventArgs e) {
      if (IsDataSimulationCorrect()) {
        Random rnd = new Random();
        int i = 0;
        DateTime oneDateTime = new DateTime();
        oneDateTime = StartDateDTP.Value;
        int maxValue = 500 * _SelectedInstitutions.NumberOfEmployees;
        _SalesSimulationsDataList.Clear();

        while (oneDateTime < EndDateDTP.Value) {
          i++;
          SalesSimulationsData oneSalesSimulationsData = new SalesSimulationsData();
          oneSalesSimulationsData.Number = i;
          oneSalesSimulationsData.DateOfSales = oneDateTime;
          oneSalesSimulationsData.NumberOfSales = rnd.Next(500, maxValue);
          _SalesSimulationsDataList.Add(oneSalesSimulationsData);
          oneDateTime = oneDateTime.AddMonths(1);
        }
        LoadSalesSimulationsData(_SalesSimulationsDataList);

      }
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _SalesSimulationsPrivider.InsertSalesSimulations(SalesSimulationsNameTBox.Text,
Convert.ToInt32(InstitutionsCBox.SelectedValue),
          Convert.ToInt32(ProductsCBox.SelectedValue), StartDateDTP.Value,
EndDateDTP.Value);
        int lastId = _SalesSimulationsPrivider.GetLastRecords();
        for (int i = 0; i < _SalesSimulationsDataList.Count; i++) {
          _SalesSimulationsDataList[i].SalesSimulationsId = lastId;
        }

_SalesSimulationsDataDataProvider.InsertBatchSalesSimulationsData(_SalesSimulationsDataLi
st);
        _SalesSimulationsDataList.Clear();
        LoadSalesSimulationsData(_SalesSimulationsDataList);
        DataLoad();
      }
    }

    private void LoadAllDate() {
      _InstitutionsList = _InstitutionsProvider.GetAllInstitutions();
      InstitutionsCBox.DataSource = _InstitutionsList;
```

```
    InstitutionsCBox.ValueMember = "InstitutionsId";
    InstitutionsCBox.DisplayMember = "InstitutionsName";
    _IsInstitutionsLoad = true;
    InstitutionsCBox_SelectedValueChanged(InstitutionsCBox, EventArgs.Empty);

    _ProductsList = _ProductsProvider.GetAllProducts();
    ProductsCBox.DataSource = _ProductsList;
    ProductsCBox.ValueMember = "ProductsId";
    ProductsCBox.DisplayMember = "ProductsName";
   }

   private void DataLoad() {
    int firstRowIndex = 0;
    if (SalesSimulationsGridView.FirstDisplayedScrollingRowIndex > 0) {
     firstRowIndex = SalesSimulationsGridView.FirstDisplayedScrollingRowIndex;
    }
    try {
     _SalesSimulationsList = _SalesSimulationsPrivider.GetAllSalesSimulations();
     LoadDataInSalesSimulationsGridView(_SalesSimulationsList);
     if (_selectedRowIndex == SalesSimulationsGridView.Rows.Count) {
      _selectedRowIndex = SalesSimulationsGridView.Rows.Count - 1;
     }
     if (_selectedRowIndex >= 0) {
      SalesSimulationsGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
      SalesSimulationsGridView.Rows[_selectedRowIndex].Selected = true;
     }
    } catch { }
   }

   private void LoadDataInSalesSimulationsGridView(List<SalesSimulations>
SalesSimulationsList) {
    SalesSimulationsGridView.DataSource = null;
    SalesSimulationsGridView.Columns.Clear();
    SalesSimulationsGridView.AutoGenerateColumns = false;
    SalesSimulationsGridView.RowHeadersVisible = false;

    SalesSimulationsGridView.DataSource = SalesSimulationsList;

    if (SalesSimulationsList.Count > 0) {
     if (SalesSimulationsList[0].Message ==
NamesMy.NoDataNames.NoDataInSalesSimulations) {
      DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
      messageColumn.DataPropertyName = "Message";
      messageColumn.Width = SalesSimulationsGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
      SalesSimulationsGridView.Columns.Add(messageColumn);
     } else {
      DataGridViewColumn SalesSimulationsDataIdColumn = new
DataGridViewTextBoxColumn();
      SalesSimulationsDataIdColumn.DataPropertyName = "SalesSimulationsId";
      SalesSimulationsGridView.Columns.Add(SalesSimulationsDataIdColumn);
      SalesSimulationsGridView.Columns[0].Visible = false;
```

```
        DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
        numberColumn.HeaderText = "№ ";
        numberColumn.DataPropertyName = "Number";
        numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        numberColumn.Width = NamesMy.SizeOptins.NumberSize;
        SalesSimulationsGridView.Columns.Add(numberColumn);

        DataGridViewColumn SalesSimulationsNameColumn = new
DataGridViewTextBoxColumn();
        SalesSimulationsNameColumn.HeaderText = "Симуляції";
        SalesSimulationsNameColumn.DataPropertyName = "SalesSimulationsName";
        SalesSimulationsNameColumn.Width = 200;
        SalesSimulationsGridView.Columns.Add(SalesSimulationsNameColumn);

        DataGridViewColumn InstitutionsNameColumn = new DataGridViewTextBoxColumn();
        InstitutionsNameColumn.HeaderText = "Заклад";
        InstitutionsNameColumn.DataPropertyName = "InstitutionsName";
        InstitutionsNameColumn.Width = 150;
        SalesSimulationsGridView.Columns.Add(InstitutionsNameColumn);

        DataGridViewColumn ProductsNameColumn = new DataGridViewTextBoxColumn();
        ProductsNameColumn.HeaderText = "Продукція";
        ProductsNameColumn.DataPropertyName = "ProductsName";
        ProductsNameColumn.Width = 200;
        SalesSimulationsGridView.Columns.Add(ProductsNameColumn);

        DataGridViewColumn StartDateColumn = new DataGridViewTextBoxColumn();
        StartDateColumn.HeaderText = "Початок періоду";
        StartDateColumn.DataPropertyName = "StartDate";
        StartDateColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        StartDateColumn.Width = 130;
        SalesSimulationsGridView.Columns.Add(StartDateColumn);

        DataGridViewColumn EndDateColumn = new DataGridViewTextBoxColumn();
        EndDateColumn.HeaderText = "Кінець періоду";
        EndDateColumn.DataPropertyName = "EndDate";
        EndDateColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        EndDateColumn.Width = 130;
        SalesSimulationsGridView.Columns.Add(EndDateColumn);


        DataGridViewButtonColumn deleteBtn = new DataGridViewButtonColumn();
        deleteBtn.HeaderText = NamesMy.ProgramButtons.DeleteBtn;
        deleteBtn.Text = NamesMy.ProgramButtons.DeleteBtn;
        deleteBtn.UseColumnTextForButtonValue = true;
        deleteBtn.ToolTipText = NamesMy.ProgramButtons.DeleteBtn;
        deleteBtn.Width = NamesMy.SizeOptins.DeleteBtnSize;
        SalesSimulationsGridView.Columns.Add(deleteBtn);
```

```
        }
        for (int i = 0; i < SalesSimulationsGridView.Columns.Count; i++) {
          SalesSimulationsGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
        }
      }
    }

    private void LoadSalesSimulationsData(List<SalesSimulationsData>
SalesSimulationsDataList) {
      SalesSimulationsDataGridView.DataSource = null;
      SalesSimulationsDataGridView.Columns.Clear();
      SalesSimulationsDataGridView.AutoGenerateColumns = false;
      SalesSimulationsDataGridView.RowHeadersVisible = false;

      SalesSimulationsDataGridView.DataSource = SalesSimulationsDataList;

      DataGridViewColumn NameColumn = new DataGridViewTextBoxColumn();
      NameColumn.HeaderText = "№";
      NameColumn.DataPropertyName = "Number";
      NameColumn.Width = 50;
      NameColumn.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
      SalesSimulationsDataGridView.Columns.Add(NameColumn);

      DataGridViewColumn DateOfSalesColumn = new DataGridViewTextBoxColumn();
      DateOfSalesColumn.HeaderText = "Дата";
      DateOfSalesColumn.DataPropertyName = "DateOfSales";
      DateOfSalesColumn.Width = 130;
      SalesSimulationsDataGridView.Columns.Add(DateOfSalesColumn);

      DataGridViewColumn NumberOfSalesColumn = new DataGridViewTextBoxColumn();
      NumberOfSalesColumn.HeaderText = "К-сть продукції";
      NumberOfSalesColumn.DataPropertyName = "NumberOfSales";
      NumberOfSalesColumn.Width = 120;
      NumberOfSalesColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
      SalesSimulationsDataGridView.Columns.Add(NumberOfSalesColumn);

      for (int i = 0; i < SalesSimulationsDataGridView.Columns.Count; i++) {
        SalesSimulationsDataGridView.Columns[i].HeaderCell.Style.BackColor =
Color.LightGray;
      }
    }

    private bool IsDataSimulationCorrect() {
      bool isCorrect = true;
      if (Convert.ToInt32(InstitutionsCBox.SelectedValue) > 0) {
        InstitutionsValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        InstitutionsValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (Convert.ToInt32(ProductsCBox.SelectedValue) > 0) {
```

```
      ProductsValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
     } else {
      ProductsValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
     }
     if (StartDateDTP.Value.ToShortDateString() != EndDateDTP.Value.ToShortDateString() ) {
      StartDateValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      EndDateValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
     } else {
      StartDateValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      EndDateValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
     }


     return isCorrect;
    }

    private bool IsDataEnteringCorrect() {
     bool isCorrect = true;
     if (_validation.IsDataEntering(SalesSimulationsNameTBox.Text)) {
      SalesSimulationsNameValiadtionLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
     } else {
      SalesSimulationsNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
     }
     if (_SalesSimulationsDataList.Count == 0) {
      MessageBox.Show("Неможливо зберегти симуляцію, так як вона не була проведена!",
"Увага!");
      isCorrect = false;
     }
     if
(_SalesSimulationsPrivider.IsSelectedSalesSimulationsExist(Convert.ToInt32(InstitutionsCBox.
SelectedValue), Convert.ToInt32(ProductsCBox.SelectedValue))) {
      MessageBox.Show("На даний продукт із вибраного закладу вже проведена
симуляція!", "Увага!");
      isCorrect = false;
     }
     return isCorrect;
    }

    private void InstitutionsCBox_SelectedValueChanged(object sender, EventArgs e) {
     if (_IsInstitutionsLoad) {
      _SelectedInstitutions =
_InstitutionsProvider.SelectedInstitutionsByInstitutionsId(Convert.ToInt32(InstitutionsCBox.Sel
ectedValue));
     }
    }

    private void SalesSimulationsGridView_CellClick(object sender,
DataGridViewCellEventArgs e) {
```

```
      if (e.ColumnIndex == 7) {
        if (MessageBox.Show("Ви дійсно бажаєте видалити це заняття?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
          int selectedSalesSimulationsId = Convert.ToInt32(SalesSimulationsGridView[0,
e.RowIndex].Value.ToString());

_SalesSimulationsPrivider.DeleteSalesSimulationsBySalesSimulationsId(selectedSalesSimulatio
nsId);

_SalesSimulationsDataDataProvider.DeleteSalesSimulationsDataBySalesSimulationsId(selected
SalesSimulationsId);
          DataLoad();
        }
      }
    }

  }
}
```

Лістинг 5. Код класу «InstitutionsForm »

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using SystemForBusinessService.AppCode;
using SystemForBusinessService.Providers;

namespace SystemForBusinessService.Forms.Dictionary {
  public partial class InstitutionsForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private InstitutionsProvider _InstitutionsPrivider = new InstitutionsProvider();
    private List<Institutions> _InstitutionsList = new List<Institutions>();
    private InstitutionsTypesProvider _InstitutionsTypesProvider = new
InstitutionsTypesProvider();
    private List<InstitutionsTypes> _InstitutionsTypesList = new List<InstitutionsTypes>();


    public InstitutionsForm() {
      InitializeComponent();
      LoadAllDate();
      DataLoad();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _InstitutionsPrivider.InsertInstitutions(InstitutionsNameTBox.Text, DescriptionTBox.Text,
Convert.ToInt32(NumberOfEmployeesTBox.Text),
```

```
      Convert.ToInt32(InstitutionsTypesCBox.SelectedValue));
    DataLoad();
    ClearAllControls();
  }
}

private void ClearBtn_Click(object sender, EventArgs e) {
  ClearAllControls();
}

private void ExitBtn_Click(object sender, EventArgs e) {
  this.Close();
}

private void LoadAllDate() {
  _InstitutionsTypesList = _InstitutionsTypesProvider.GetAllInstitutionsTypes();
  InstitutionsTypesCBox.DataSource = _InstitutionsTypesList;
  InstitutionsTypesCBox.ValueMember = "InstitutionsTypesId";
  InstitutionsTypesCBox.DisplayMember = "InstitutionsTypesName";
}

private void DataLoad() {
  int firstRowIndex = 0;
  if (InstitutionsGridView.FirstDisplayedScrollingRowIndex > 0) {
    firstRowIndex = InstitutionsGridView.FirstDisplayedScrollingRowIndex;
  }
  try {
    _InstitutionsList = _InstitutionsPrivider.GetAllInstitutions();
    LoadDataInInstitutionsGridView(_InstitutionsList);
    if (_selectedRowIndex == InstitutionsGridView.Rows.Count) {
      _selectedRowIndex = InstitutionsGridView.Rows.Count - 1;
    }
    if (_selectedRowIndex >= 0) {
      InstitutionsGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
      InstitutionsGridView.Rows[_selectedRowIndex].Selected = true;
    }
  } catch { }
}

private void LoadDataInInstitutionsGridView(List<Institutions> InstitutionsList) {
  InstitutionsGridView.DataSource = null;
  InstitutionsGridView.Columns.Clear();
  InstitutionsGridView.AutoGenerateColumns = false;
  InstitutionsGridView.RowHeadersVisible = false;

  InstitutionsGridView.DataSource = InstitutionsList;

  if (InstitutionsList.Count > 0) {
    if (InstitutionsList[0].Message == NamesMy.NoDataNames.NoDataInInstitutions) {
      DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
      messageColumn.DataPropertyName = "Message";
      messageColumn.Width = InstitutionsGridView.Width -
```

```
NamesMy.SizeOptins.MinusSizePanel;
        InstitutionsGridView.Columns.Add(messageColumn);
       } else {
        DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
        DetailIdColumn.DataPropertyName = "InstitutionsId";
        InstitutionsGridView.Columns.Add(DetailIdColumn);
        InstitutionsGridView.Columns[0].Visible = false;

        DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
        numberColumn.HeaderText = "№ ";
        numberColumn.DataPropertyName = "Number";
        numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        numberColumn.Width = NamesMy.SizeOptins.NumberSize;
        InstitutionsGridView.Columns.Add(numberColumn);

        DataGridViewColumn InstitutionsNameColumn = new DataGridViewTextBoxColumn();
        InstitutionsNameColumn.HeaderText = "Назва закладу";
        InstitutionsNameColumn.DataPropertyName = "InstitutionsName";
        InstitutionsNameColumn.Width = NamesMy.SizeOptins.NameSize;
        InstitutionsGridView.Columns.Add(InstitutionsNameColumn);

        DataGridViewColumn AmountColumn = new DataGridViewTextBoxColumn();
        AmountColumn.HeaderText = "К-сть прац.";
        AmountColumn.DataPropertyName = "NumberOfEmployees";
        AmountColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        AmountColumn.Width = 100;
        InstitutionsGridView.Columns.Add(AmountColumn);
       }
       for (int i = 0; i < InstitutionsGridView.Columns.Count; i++) {
        InstitutionsGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
       }
      }
    }

    private void ClearAllControls() {
      InstitutionsNameTBox.Text = String.Empty;
      DescriptionTBox.Text = String.Empty;
      NumberOfEmployeesTBox.Text = "0";
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(InstitutionsNameTBox.Text)) {
        InstitutionsNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        InstitutionsNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataConvertToInt(NumberOfEmployeesTBox.Text)) {
        NumberOfEmployeesValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
```

```
      } else {
        NumberOfEmployeesValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void InstitutionsGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && InstitutionsGridView[0, e.RowIndex].Value.ToString() !=
_InstitutionsList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateInstitutionsForm updateInstitutionsForm = new
UpdateInstitutionsForm(Convert.ToInt32(InstitutionsGridView[0,
e.RowIndex].Value.ToString()));
        updateInstitutionsForm.ShowDialog();
        DataLoad();
      }
    }

  }
}
```

Лістинг 6. Код класу «InstitutionsTypesForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using SystemForBusinessService.AppCode;
using SystemForBusinessService.Providers;

namespace SystemForBusinessService.Forms.Dictionary {
  public partial class InstitutionsTypesForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private InstitutionsTypesProvider _InstitutionsTypesProvider = new
InstitutionsTypesProvider();
    private List<InstitutionsTypes> _InstitutionsTypesList = new List<InstitutionsTypes>();

    public InstitutionsTypesForm() {
      InitializeComponent();
      DataLoad();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _InstitutionsTypesProvider.InsertInstitutionsTypes(InstitutionsTypesNameTBox.Text,
DescriptionTBox.Text);
        DataLoad();
```

```
      ClearAllControls();
    }
  }

  private void ClearBtn_Click(object sender, EventArgs e) {
    ClearAllControls();
  }

  private void ExitBtn_Click(object sender, EventArgs e) {
    this.Close();
  }

  private void DataLoad() {
    int firstRowIndex = 0;
    if (InstitutionsTypesGridView.FirstDisplayedScrollingRowIndex > 0) {
      firstRowIndex = InstitutionsTypesGridView.FirstDisplayedScrollingRowIndex;
    }
    try {
      _InstitutionsTypesList = _InstitutionsTypesProvider.GetAllInstitutionsTypes();
      LoadDataInInstitutionsTypesGridView(_InstitutionsTypesList);
      if (_selectedRowIndex == InstitutionsTypesGridView.Rows.Count) {
        _selectedRowIndex = InstitutionsTypesGridView.Rows.Count - 1;
      }
      if (_selectedRowIndex >= 0) {
        InstitutionsTypesGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
        InstitutionsTypesGridView.Rows[_selectedRowIndex].Selected = true;
      }
    } catch { }
  }

  private void LoadDataInInstitutionsTypesGridView(List<InstitutionsTypes>
InstitutionsTypesList) {
    InstitutionsTypesGridView.DataSource = null;
    InstitutionsTypesGridView.Columns.Clear();
    InstitutionsTypesGridView.AutoGenerateColumns = false;
    InstitutionsTypesGridView.RowHeadersVisible = false;

    InstitutionsTypesGridView.DataSource = InstitutionsTypesList;

    if (InstitutionsTypesList.Count > 0) {
      if (InstitutionsTypesList[0].Message ==
NamesMy.NoDataNames.NoDataInInstitutionsTypes) {
        DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
        messageColumn.DataPropertyName = "Message";
        messageColumn.Width = InstitutionsTypesGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
        InstitutionsTypesGridView.Columns.Add(messageColumn);
      } else {
        DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
        DetailIdColumn.DataPropertyName = "InstitutionsTypesId";
        InstitutionsTypesGridView.Columns.Add(DetailIdColumn);
        InstitutionsTypesGridView.Columns[0].Visible = false;
```

```
        DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
        numberColumn.HeaderText = "№ ";
        numberColumn.DataPropertyName = "Number";
        numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        numberColumn.Width = NamesMy.SizeOptins.NumberSize;
        InstitutionsTypesGridView.Columns.Add(numberColumn);

        DataGridViewColumn InstitutionsTypesNameColumn = new
DataGridViewTextBoxColumn();
        InstitutionsTypesNameColumn.HeaderText = "Категорії";
        InstitutionsTypesNameColumn.DataPropertyName = "InstitutionsTypesName";
        InstitutionsTypesNameColumn.Width = NamesMy.SizeOptins.NameSize;
        InstitutionsTypesGridView.Columns.Add(InstitutionsTypesNameColumn);


      }
      for (int i = 0; i < InstitutionsTypesGridView.Columns.Count; i++) {
        InstitutionsTypesGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
      }
    }
  }

  private void ClearAllControls() {
    InstitutionsTypesNameTBox.Text = String.Empty;
    DescriptionTBox.Text = String.Empty;
  }

  private bool IsDataEnteringCorrect() {
    bool isCorrect = true;
    if (_validation.IsDataEntering(InstitutionsTypesNameTBox.Text)) {
      InstitutionsTypesNameValiadtionLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
    } else {
      InstitutionsTypesNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
      isCorrect = false;
    }
    return isCorrect;
  }

  private void InstitutionsTypesGridView_CellClick(object sender,
DataGridViewCellEventArgs e) {
    if (e.RowIndex >= 0 && InstitutionsTypesGridView[0, e.RowIndex].Value.ToString() !=
_InstitutionsTypesList[0].Message) {
      _selectedRowIndex = e.RowIndex;
      UpdateITForm updateInstitutionsTypesForm = new
UpdateITForm(Convert.ToInt32(InstitutionsTypesGridView[0, e.RowIndex].Value.ToString()));
      updateInstitutionsTypesForm.ShowDialog();
      DataLoad();
    }
  }
```

```
  }
}
```
Лістинг 7. Код класу «PostsForm »
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using SystemForBusinessService.AppCode;
using SystemForBusinessService.Providers;

namespace SystemForBusinessService.Forms.Dictionary {
  public partial class PostsForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private PostsProvider _PostsProvider = new PostsProvider();
    private List<Posts> _PostsList = new List<Posts>();

    public PostsForm() {
      InitializeComponent();
      DataLoad();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _PostsProvider.InsertPosts(PostsNameTBox.Text, DescriptionTBox.Text);
        DataLoad();
        ClearAllControls();
      }
    }

    private void ClearBtn_Click(object sender, EventArgs e) {
      ClearAllControls();
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void DataLoad() {
      int firstRowIndex = 0;
      if (PostsGridView.FirstDisplayedScrollingRowIndex > 0) {
        firstRowIndex = PostsGridView.FirstDisplayedScrollingRowIndex;
      }
      try {
        _PostsList = _PostsProvider.GetAllPosts();
        LoadDataInPostsGridView(_PostsList);
```

```
        if (_selectedRowIndex == PostsGridView.Rows.Count) {
          _selectedRowIndex = PostsGridView.Rows.Count - 1;
        }
        if (_selectedRowIndex >= 0) {
          PostsGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
          PostsGridView.Rows[_selectedRowIndex].Selected = true;
        }
      } catch { }
    }

    private void LoadDataInPostsGridView(List<Posts> PostsList) {
      PostsGridView.DataSource = null;
      PostsGridView.Columns.Clear();
      PostsGridView.AutoGenerateColumns = false;
      PostsGridView.RowHeadersVisible = false;

      PostsGridView.DataSource = PostsList;

      if (PostsList.Count > 0) {
        if (PostsList[0].Message == NamesMy.NoDataNames.NoDataInPosts) {
          DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
          messageColumn.DataPropertyName = "Message";
          messageColumn.Width = PostsGridView.Width - NamesMy.SizeOptins.MinusSizePanel;
          PostsGridView.Columns.Add(messageColumn);
        } else {
          DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
          DetailIdColumn.DataPropertyName = "PostsId";
          PostsGridView.Columns.Add(DetailIdColumn);
          PostsGridView.Columns[0].Visible = false;

          DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
          numberColumn.HeaderText = "№ ";
          numberColumn.DataPropertyName = "Number";
          numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
          numberColumn.Width = NamesMy.SizeOptins.NumberSize;
          PostsGridView.Columns.Add(numberColumn);

          DataGridViewColumn PostsNameColumn = new DataGridViewTextBoxColumn();
          PostsNameColumn.HeaderText = "Посада";
          PostsNameColumn.DataPropertyName = "PostsName";
          PostsNameColumn.Width = NamesMy.SizeOptins.NameSize;
          PostsGridView.Columns.Add(PostsNameColumn);


        }
        for (int i = 0; i < PostsGridView.Columns.Count; i++) {
          PostsGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
        }
      }
    }
```

```
    private void ClearAllControls() {
      PostsNameTBox.Text = String.Empty;
      DescriptionTBox.Text = String.Empty;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(PostsNameTBox.Text)) {
        PostsNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PostsNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void PostsGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && PostsGridView[0, e.RowIndex].Value.ToString() !=
_PostsList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdatePostsForm updatePostsForm = new
UpdatePostsForm(Convert.ToInt32(PostsGridView[0, e.RowIndex].Value.ToString()));
        updatePostsForm.ShowDialog();
        DataLoad();
      }
    }

  }
}
```

Лістинг 8. Код класу « ProductsCategorysForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using SystemForBusinessService.AppCode;
using SystemForBusinessService.Providers;

namespace SystemForBusinessService.Forms.Dictionary {
  public partial class ProductsCategorysForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private ProductsCategorysProvider _ProductsCategorysProvider = new
ProductsCategorysProvider();
    private List<ProductsCategorys> _ProductsCategorysList = new List<ProductsCategorys>();

    public ProductsCategorysForm() {
      InitializeComponent();
```

```
    DataLoad();
  }

  private void AddBtn_Click(object sender, EventArgs e) {
    if (IsDataEnteringCorrect()) {
      _ProductsCategorysProvider.InsertProductsCategorys(ProductsCategorysNameTBox.Text,
DescriptionTBox.Text);
      DataLoad();
      ClearAllControls();
    }
  }

  private void ClearBtn_Click(object sender, EventArgs e) {
    ClearAllControls();
  }

  private void ExitBtn_Click(object sender, EventArgs e) {
    this.Close();
  }

  private void DataLoad() {
    int firstRowIndex = 0;
    if (ProductsCategorysGridView.FirstDisplayedScrollingRowIndex > 0) {
      firstRowIndex = ProductsCategorysGridView.FirstDisplayedScrollingRowIndex;
    }
    try {
      _ProductsCategorysList = _ProductsCategorysProvider.GetAllProductsCategorys();
      LoadDataInProductsCategorysGridView(_ProductsCategorysList);
      if (_selectedRowIndex == ProductsCategorysGridView.Rows.Count) {
        _selectedRowIndex = ProductsCategorysGridView.Rows.Count - 1;
      }
      if (_selectedRowIndex >= 0) {
        ProductsCategorysGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
        ProductsCategorysGridView.Rows[_selectedRowIndex].Selected = true;
      }
    } catch { }
  }

  private void LoadDataInProductsCategorysGridView(List<ProductsCategorys>
ProductsCategorysList) {
    ProductsCategorysGridView.DataSource = null;
    ProductsCategorysGridView.Columns.Clear();
    ProductsCategorysGridView.AutoGenerateColumns = false;
    ProductsCategorysGridView.RowHeadersVisible = false;

    ProductsCategorysGridView.DataSource = ProductsCategorysList;

    if (ProductsCategorysList.Count > 0) {
      if (ProductsCategorysList[0].Message ==
NamesMy.NoDataNames.NoDataInProductsCategorys) {
        DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
        messageColumn.DataPropertyName = "Message";
```

```
        messageColumn.Width = ProductsCategorysGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
        ProductsCategorysGridView.Columns.Add(messageColumn);
       } else {
        DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
        DetailIdColumn.DataPropertyName = "ProductsCategorysId";
        ProductsCategorysGridView.Columns.Add(DetailIdColumn);
        ProductsCategorysGridView.Columns[0].Visible = false;

        DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
        numberColumn.HeaderText = "№ ";
        numberColumn.DataPropertyName = "Number";
        numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        numberColumn.Width = NamesMy.SizeOptins.NumberSize;
        ProductsCategorysGridView.Columns.Add(numberColumn);

        DataGridViewColumn ProductsCategorysNameColumn = new
DataGridViewTextBoxColumn();
        ProductsCategorysNameColumn.HeaderText = "Категорії";
        ProductsCategorysNameColumn.DataPropertyName = "ProductsCategorysName";
        ProductsCategorysNameColumn.Width = NamesMy.SizeOptins.NameSize;
        ProductsCategorysGridView.Columns.Add(ProductsCategorysNameColumn);


       }
       for (int i = 0; i < ProductsCategorysGridView.Columns.Count; i++) {
        ProductsCategorysGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
       }
     }
   }

   private void ClearAllControls() {
     ProductsCategorysNameTBox.Text = String.Empty;
     DescriptionTBox.Text = String.Empty;
   }

   private bool IsDataEnteringCorrect() {
     bool isCorrect = true;
     if (_validation.IsDataEntering(ProductsCategorysNameTBox.Text)) {
       ProductsCategorysNameValiadtionLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
     } else {
       ProductsCategorysNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
       isCorrect = false;
     }
     return isCorrect;
   }


   private void ProductsCategorysGridView_CellClick(object sender,
DataGridViewCellEventArgs e) {
```

```
      if (e.RowIndex >= 0 && ProductsCategorysGridView[0, e.RowIndex].Value.ToString() !=
_ProductsCategorysList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateProductsCategorysForm updateProductsCategorysForm = new
UpdateProductsCategorysForm(Convert.ToInt32(ProductsCategorysGridView[0,
e.RowIndex].Value.ToString()));
        updateProductsCategorysForm.ShowDialog();
        DataLoad();
      }
    }

  }
}
```

Лістинг 9. Код класу «ProductsForm »

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using SystemForBusinessService.AppCode;
using SystemForBusinessService.Providers;

namespace SystemForBusinessService.Forms.Dictionary {
  public partial class ProductsForm : Form {
    private int _selectedRowIndex = 0;
    private ValidationMy _validation = new ValidationMy();
    private ProductsProvider _ProductsPrivider = new ProductsProvider();
    private List<Products> _ProductsList = new List<Products>();
    private ProductsCategorysProvider _ProductsCategorysProvider = new
ProductsCategorysProvider();
    private List<ProductsCategorys> _ProductsCategorysList = new List<ProductsCategorys>();

    public ProductsForm() {
      InitializeComponent();
      LoadAllDate();
      DataLoad();
    }

    private void AddBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _ProductsPrivider.InsertProducts(ProductsNameTBox.Text, DescriptionTBox.Text,
Convert.ToInt32(ProductsCategorysCBox.SelectedValue),
          Convert.ToDouble(ManufacturingPriceTBox.Text),
Convert.ToDouble(SellingPriceTBox.Text),
Convert.ToDouble(DiscountOnProductsTBox.Text));
        DataLoad();
        ClearAllControls();
      }
```

```
  }

  private void ClearBtn_Click(object sender, EventArgs e) {
    ClearAllControls();
  }

  private void ExitBtn_Click(object sender, EventArgs e) {
    this.Close();
  }

  private void LoadAllDate() {
    _ProductsCategorysList = _ProductsCategorysProvider.GetAllProductsCategorys();
    ProductsCategorysCBox.DataSource = _ProductsCategorysList;
    ProductsCategorysCBox.ValueMember = "ProductsCategorysId";
    ProductsCategorysCBox.DisplayMember = "ProductsCategorysName";
  }

  private void DataLoad() {
    int firstRowIndex = 0;
    if (ProductsGridView.FirstDisplayedScrollingRowIndex > 0) {
      firstRowIndex = ProductsGridView.FirstDisplayedScrollingRowIndex;
    }
    try {
      _ProductsList = _ProductsPrivider.GetAllProducts();
      LoadDataInProductsGridView(_ProductsList);
      if (_selectedRowIndex == ProductsGridView.Rows.Count) {
        _selectedRowIndex = ProductsGridView.Rows.Count - 1;
      }
      if (_selectedRowIndex >= 0) {
        ProductsGridView.FirstDisplayedScrollingRowIndex = firstRowIndex;
        ProductsGridView.Rows[_selectedRowIndex].Selected = true;
      }
    } catch { }
  }

  private void LoadDataInProductsGridView(List<Products> ProductsList) {
    ProductsGridView.DataSource = null;
    ProductsGridView.Columns.Clear();
    ProductsGridView.AutoGenerateColumns = false;
    ProductsGridView.RowHeadersVisible = false;

    ProductsGridView.DataSource = ProductsList;

    if (ProductsList.Count > 0) {
      if (ProductsList[0].Message == NamesMy.NoDataNames.NoDataInProducts) {
        DataGridViewColumn messageColumn = new DataGridViewTextBoxColumn();
        messageColumn.DataPropertyName = "Message";
        messageColumn.Width = ProductsGridView.Width -
NamesMy.SizeOptins.MinusSizePanel;
        ProductsGridView.Columns.Add(messageColumn);
      } else {
        DataGridViewColumn DetailIdColumn = new DataGridViewTextBoxColumn();
```

```
        DetailIdColumn.DataPropertyName = "ProductsId";
        ProductsGridView.Columns.Add(DetailIdColumn);
        ProductsGridView.Columns[0].Visible = false;

        DataGridViewColumn numberColumn = new DataGridViewTextBoxColumn();
        numberColumn.HeaderText = "№ ";
        numberColumn.DataPropertyName = "Number";
        numberColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        numberColumn.Width = NamesMy.SizeOptins.NumberSize;
        ProductsGridView.Columns.Add(numberColumn);

        DataGridViewColumn ProductsNameColumn = new DataGridViewTextBoxColumn();
        ProductsNameColumn.HeaderText = "Назва продукції";
        ProductsNameColumn.DataPropertyName = "ProductsName";
        ProductsNameColumn.Width = NamesMy.SizeOptins.NameSize;
        ProductsGridView.Columns.Add(ProductsNameColumn);

        DataGridViewColumn ManufacturingPriceColumn = new
DataGridViewTextBoxColumn();
        ManufacturingPriceColumn.HeaderText = "Виготовлення";
        ManufacturingPriceColumn.DataPropertyName = "ManufacturingPrice";
        ManufacturingPriceColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        ManufacturingPriceColumn.Width = 100;
        ProductsGridView.Columns.Add(ManufacturingPriceColumn);

        DataGridViewColumn SellingPriceColumn = new DataGridViewTextBoxColumn();
        SellingPriceColumn.HeaderText = "Продаж";
        SellingPriceColumn.DataPropertyName = "SellingPrice";
        SellingPriceColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        SellingPriceColumn.Width = 100;
        ProductsGridView.Columns.Add(SellingPriceColumn);

        DataGridViewColumn DiscountOnProductsColumn = new
DataGridViewTextBoxColumn();
        DiscountOnProductsColumn.HeaderText = "Знижка";
        DiscountOnProductsColumn.DataPropertyName = "DiscountOnProducts";
        DiscountOnProductsColumn.DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
        DiscountOnProductsColumn.Width = 100;
        ProductsGridView.Columns.Add(DiscountOnProductsColumn);

      }
      for (int i = 0; i < ProductsGridView.Columns.Count; i++) {
        ProductsGridView.Columns[i].HeaderCell.Style.BackColor = Color.LightGray;
      }
    }
  }

  private void ClearAllControls() {
```

```
      ProductsNameTBox.Text = String.Empty;
      DescriptionTBox.Text = String.Empty;
      ManufacturingPriceTBox.Text = "0";
      SellingPriceTBox.Text = "0";
      DiscountOnProductsTBox.Text = "0";
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_validation.IsDataEntering(ProductsNameTBox.Text)) {
        ProductsNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ProductsNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataConvertToDouble(ManufacturingPriceTBox.Text)) {
        ManufacturingPriceValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        ManufacturingPriceValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataConvertToDouble(SellingPriceTBox.Text)) {
        SellingPriceValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        SellingPriceValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_validation.IsDataConvertToDouble(DiscountOnProductsTBox.Text)) {
        DiscountOnProductsValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        DiscountOnProductsValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

    private void ProductsGridView_CellClick(object sender, DataGridViewCellEventArgs e) {
      if (e.RowIndex >= 0 && ProductsGridView[0, e.RowIndex].Value.ToString() !=
_ProductsList[0].Message) {
        _selectedRowIndex = e.RowIndex;
        UpdateProductsForm updateProductsForm = new
UpdateProductsForm(Convert.ToInt32(ProductsGridView[0, e.RowIndex].Value.ToString()));
        updateProductsForm.ShowDialog();
        DataLoad();
      }
    }
  }
}
```

Лістинг 10. Код класу «UpdateInstitutionsForm »

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using SystemForBusinessService.AppCode;
using SystemForBusinessService.Providers;

namespace SystemForBusinessService.Forms.Dictionary {
  public partial class UpdateInstitutionsForm : Form {
    private int _InstitutionsId = 0;
    private Institutions _selectedInstitutions = new Institutions();
    private InstitutionsProvider _InstitutionsPrivider = new InstitutionsProvider();
    private ValidationMy _Validation = new ValidationMy();
    private InstitutionsTypesProvider _InstitutionsTypesProvider = new
InstitutionsTypesProvider();
    private List<InstitutionsTypes> _InstitutionsTypesList = new List<InstitutionsTypes>();

    public UpdateInstitutionsForm(int InstitutionsId) {
      InitializeComponent();
      _InstitutionsId = InstitutionsId;
      LoadAllDate();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _InstitutionsPrivider.UpdateInstitutions(InstitutionsNameTBox.Text,
DescriptionTBox.Text, Convert.ToInt32(NumberOfEmployeesTBox.Text),
          Convert.ToInt32(InstitutionsTypesCBox.SelectedValue), _InstitutionsId);
        this.Close();
      }
    }

    private void DeleteBtn_Click(object sender, EventArgs e) {
      if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
        _InstitutionsPrivider.DeleteInstitutionsByInstitutionsId(_InstitutionsId);
        this.Close();
      }
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void LoadAllDate() {
      _InstitutionsTypesList = _InstitutionsTypesProvider.GetAllInstitutionsTypes();
      InstitutionsTypesCBox.DataSource = _InstitutionsTypesList;
      InstitutionsTypesCBox.ValueMember = "InstitutionsTypesId";
      InstitutionsTypesCBox.DisplayMember = "InstitutionsTypesName";
```

```
      _selectedInstitutions =
_InstitutionsPrivider.SelectedInstitutionsByInstitutionsId(_InstitutionsId);
      InstitutionsNameTBox.Text = _selectedInstitutions.InstitutionsName;
      DescriptionTBox.Text = _selectedInstitutions.Description;
      NumberOfEmployeesTBox.Text = _selectedInstitutions.NumberOfEmployees.ToString();
      InstitutionsTypesCBox.SelectedValue = _selectedInstitutions.InstitutionsTypesId;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_Validation.IsDataEntering(InstitutionsNameTBox.Text)) {
        InstitutionsNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        InstitutionsNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      if (_Validation.IsDataConvertToInt(NumberOfEmployeesTBox.Text)) {
        NumberOfEmployeesValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        NumberOfEmployeesValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }
  }
}
```

Лістинг 11. Код класу « UpdateITForm»

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using SystemForBusinessService.AppCode;
using SystemForBusinessService.Providers;

namespace SystemForBusinessService.Forms.Dictionary {
  public partial class UpdateITForm : Form {
    private int _InstitutionsTypesId = 0;
    private InstitutionsTypes _selectedInstitutionsTypes = new InstitutionsTypes();
    private InstitutionsTypesProvider _InstitutionsTypesProvider = new
InstitutionsTypesProvider();
    private ValidationMy _Validation = new ValidationMy();

    public UpdateITForm(int InstitutionsTypesId) {
      InitializeComponent();
      _InstitutionsTypesId = InstitutionsTypesId;
      LoadAllDate();
```

```
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _InstitutionsTypesProvider.UpdateInstitutionsTypes(InstitutionsTypesNameTBox.Text,
DescriptionTBox.Text, _InstitutionsTypesId);
        this.Close();
      }
    }

    private void DeleteBtn_Click(object sender, EventArgs e) {
      if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {

_InstitutionsTypesProvider.DeleteInstitutionsTypesByInstitutionsTypesId(_InstitutionsTypesId);
        this.Close();
      }
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      this.Close();
    }

    private void LoadAllDate() {
      _selectedInstitutionsTypes =
_InstitutionsTypesProvider.SelectedInstitutionsTypesByInstitutionsTypesId(_InstitutionsTypesI
d);
      InstitutionsTypesNameTBox.Text = _selectedInstitutionsTypes.InstitutionsTypesName;
      DescriptionTBox.Text = _selectedInstitutionsTypes.Description;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_Validation.IsDataEntering(InstitutionsTypesNameTBox.Text)) {
        InstitutionsTypesNameValiadtionLbl.Text =
NamesMy.ProgramButtons.RequiredValidation;
      } else {
        InstitutionsTypesNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }


  }
}
```

Лістинг 12. Код класу «UpdatePostsForm »

```
using System;
using System.Windows.Forms;
using SystemForBusinessService.AppCode;
using SystemForBusinessService.Providers;
```

```
namespace SystemForBusinessService.Forms.Dictionary {
  public partial class UpdatePostsForm : Form {
    private int _PostsId = 0;
    private Posts _selectedPosts = new Posts();
    private PostsProvider _PostsProvider = new PostsProvider();
    private ValidationMy _Validation = new ValidationMy();

    public UpdatePostsForm(int PostsId) {
      InitializeComponent();
      _PostsId = PostsId;
      LoadAllDate();
    }

    private void SaveBtn_Click(object sender, EventArgs e) {
      if (IsDataEnteringCorrect()) {
        _PostsProvider.UpdatePosts(PostsNameTBox.Text, DescriptionTBox.Text, _PostsId);
        this.Close();
      }
    }

    private void DeleteBtn_Click(object sender, EventArgs e) {
      if (MessageBox.Show("Ви дійсно хочете видалити цей елемент?", "Видалити",
MessageBoxButtons.YesNo) == DialogResult.Yes) {
        _PostsProvider.DeletePostsByPostsId(_PostsId);
        this.Close();
      }
    }

    private void ExitBtn_Click(object sender, EventArgs e) {
      _selectedPosts = _PostsProvider.SelectedPostsByPostsId(_PostsId);
      PostsNameTBox.Text = _selectedPosts.PostsName;
      DescriptionTBox.Text = _selectedPosts.Description;
    }

    private void LoadAllDate() {
      _selectedPosts = _PostsProvider.SelectedPostsByPostsId(_PostsId);
      PostsNameTBox.Text = _selectedPosts.PostsName;
      DescriptionTBox.Text = _selectedPosts.Description;
    }

    private bool IsDataEnteringCorrect() {
      bool isCorrect = true;
      if (_Validation.IsDataEntering(PostsNameTBox.Text)) {
        PostsNameValiadtionLbl.Text = NamesMy.ProgramButtons.RequiredValidation;
      } else {
        PostsNameValiadtionLbl.Text = NamesMy.ProgramButtons.ErrorValidation;
        isCorrect = false;
      }
      return isCorrect;
    }

  }
```

```
}
```
Лістинг 13. Код класу « InstitutionsProvider»
```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SystemForBusinessService.AppCode;

namespace SystemForBusinessService.Providers {
  class InstitutionsProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    public void InsertInstitutions(string InstitutionsName, string Description, int
NumberOfEmployees, int InstitutionsTypesId) {
      string SqlString = "INSERT INTO Institutions (InstitutionsName, Description,
NumberOfEmployees, InstitutionsTypesId" +
        ") Values(?, ?, ?, ?)";

      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("InstitutionsName", InstitutionsName);
          cmd.Parameters.AddWithValue("Description", Description);
          cmd.Parameters.AddWithValue("NumberOfEmployees", NumberOfEmployees);
          cmd.Parameters.AddWithValue("InstitutionsTypesId", InstitutionsTypesId);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

    public List<Institutions> GetAllInstitutions() {
      int i = 0;
      string SqlString = "SELECT * FROM Institutions";

      List<Institutions> listInstitutions = new List<Institutions>();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              Institutions oneInstitutions = new Institutions();
              oneInstitutions.Number = ++i;
              oneInstitutions.InstitutionsId = Convert.ToInt32(reader["InstitutionsId"]);
              oneInstitutions.InstitutionsName = reader["InstitutionsName"].ToString();
              oneInstitutions.Description = reader["Description"].ToString();
              oneInstitutions.NumberOfEmployees =
```

```
Convert.ToInt32(reader["NumberOfEmployees"]);
            oneInstitutions.InstitutionsTypesId = Convert.ToInt32(reader["InstitutionsTypesId"]);
            listInstitutions.Add(oneInstitutions);
          }
        }
        conn.Close();
      }
    }

    if (listInstitutions.Count == 0) {
      Institutions noInstitutions = new Institutions();
      noInstitutions.InstitutionsId = 0;
      noInstitutions.Message = NamesMy.NoDataNames.NoDataInInstitutions;
      listInstitutions.Add(noInstitutions);
    }
    return listInstitutions;
  }

  public Institutions SelectedInstitutionsByInstitutionsId(int InstitutionsId) {
    string SqlString = "SELECT * FROM Institutions Where InstitutionsId=" +
InstitutionsId.ToString();

    Institutions oneInstitutions = new Institutions();
    using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
      using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
        conn.Open();
        using (OleDbDataReader reader = cmd.ExecuteReader()) {
          while (reader.Read()) {
            oneInstitutions.InstitutionsId = Convert.ToInt32(reader["InstitutionsId"]);
            oneInstitutions.InstitutionsName = reader["InstitutionsName"].ToString();
            oneInstitutions.Description = reader["Description"].ToString();
            oneInstitutions.NumberOfEmployees =
Convert.ToInt32(reader["NumberOfEmployees"]);
            oneInstitutions.InstitutionsTypesId = Convert.ToInt32(reader["InstitutionsTypesId"]);

          }
        }
      }
      conn.Close();
    }
    return oneInstitutions;
  }

  public void UpdateInstitutions(string InstitutionsName, string Description, int
NumberOfEmployees, int InstitutionsTypesId, int InstitutionsId) {
    string SqlString = "UPDATE Institutions SET InstitutionsName=?, Description=?,
NumberOfEmployees=?, InstitutionsTypesId=? " +
 "WHERE InstitutionsId=?";

    using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
      using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
        cmd.CommandType = CommandType.Text;
```

```
        cmd.Parameters.AddWithValue("InstitutionsName", InstitutionsName);
        cmd.Parameters.AddWithValue("Description", Description);
        cmd.Parameters.AddWithValue("NumberOfEmployees", NumberOfEmployees);
        cmd.Parameters.AddWithValue("InstitutionsTypesId", InstitutionsTypesId);
        cmd.Parameters.AddWithValue("InstitutionsId", InstitutionsId);
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
       }
      }
    }

    public void DeleteInstitutionsByInstitutionsId(int InstitutionsId) {
      string SqlString = "DELETE FROM Institutions WHERE InstitutionsId=" +
InstitutionsId.ToString();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
    }

  }
}

public class Institutions {
  private int _Number;
  private int _InstitutionsId;
  private string _InstitutionsName;
  private string _Description;
  private int _NumberOfEmployees;
  private int _InstitutionsTypesId;
  private string _Message;

  public Institutions() {
    _Number = 0;
    _InstitutionsId = 0;
    _InstitutionsName = String.Empty;
    _Description = String.Empty;
    _NumberOfEmployees = 0;
    _InstitutionsTypesId = 0;
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int InstitutionsId {
    set { _InstitutionsId = value; }
```

```
    get { return _InstitutionsId; }
   }
   public string InstitutionsName {
    set { _InstitutionsName = value; }
    get { return _InstitutionsName; }
   }
   public string Description {
    set { _Description = value; }
    get { return _Description; }
   }
   public int NumberOfEmployees {
    set { _NumberOfEmployees = value; }
    get { return _NumberOfEmployees; }
   }
   public int InstitutionsTypesId {
    set { _InstitutionsTypesId = value; }
    get { return _InstitutionsTypesId; }
   }
   public string Message {
    set { _Message = value; }
    get { return _Message; }
   }
}
```

Лістинг 14. Код класу « InstitutionsTypesProvider»

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using SystemForBusinessService.AppCode;

namespace SystemForBusinessService.Providers {
  class InstitutionsTypesProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    public void InsertInstitutionsTypes(string InstitutionsTypesName, string Description) {
      string SqlString = "INSERT INTO InstitutionsTypes (InstitutionsTypesName, Description" +
        ") Values(?, ?)";

      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("InstitutionsTypesName", InstitutionsTypesName);
          cmd.Parameters.AddWithValue("Description", Description);
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
        }
      }
```

```
    }

    public List<InstitutionsTypes> GetAllInstitutionsTypes() {
      int i = 0;
      string SqlString = "SELECT InstitutionsTypesId, InstitutionsTypesName, Description " +
        "FROM InstitutionsTypes";

      List<InstitutionsTypes> listInstitutionsTypes = new List<InstitutionsTypes>();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              InstitutionsTypes oneInstitutionsTypes = new InstitutionsTypes();
              oneInstitutionsTypes.Number = ++i;
              oneInstitutionsTypes.InstitutionsTypesId =
Convert.ToInt32(reader["InstitutionsTypesId"].ToString());
              oneInstitutionsTypes.InstitutionsTypesName =
reader["InstitutionsTypesName"].ToString();
              oneInstitutionsTypes.Description = reader["Description"].ToString();
              listInstitutionsTypes.Add(oneInstitutionsTypes);
            }
          }
          conn.Close();
        }
      }

      if (listInstitutionsTypes.Count == 0) {
        InstitutionsTypes noInstitutionsTypes = new InstitutionsTypes();
        noInstitutionsTypes.InstitutionsTypesId = 0;
        noInstitutionsTypes.Message = NamesMy.NoDataNames.NoDataInInstitutionsTypes;
        listInstitutionsTypes.Add(noInstitutionsTypes);
      }
      return listInstitutionsTypes;
    }

    public InstitutionsTypes SelectedInstitutionsTypesByInstitutionsTypesId(int
InstitutionsTypesId) {
      string SqlString = "SELECT InstitutionsTypesId, InstitutionsTypesName, Description " +
        "FROM InstitutionsTypes Where InstitutionsTypesId=" + InstitutionsTypesId.ToString();

      InstitutionsTypes oneInstitutionsTypes = new InstitutionsTypes();
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          conn.Open();
          using (OleDbDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              oneInstitutionsTypes.InstitutionsTypesId =
Convert.ToInt32(reader["InstitutionsTypesId"].ToString());
              oneInstitutionsTypes.InstitutionsTypesName =
reader["InstitutionsTypesName"].ToString();
              oneInstitutionsTypes.Description = reader["Description"].ToString();
```

```
          }
        }
      }
      conn.Close();
    }
    return oneInstitutionsTypes;
  }

  public void UpdateInstitutionsTypes(string InstitutionsTypesName, string Description, int
InstitutionsTypesId) {
    string SqlString = "UPDATE InstitutionsTypes SET InstitutionsTypesName=?,
Description=?  " +
 "WHERE InstitutionsTypesId=?";

    using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
      using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("InstitutionsTypesName", InstitutionsTypesName);
        cmd.Parameters.AddWithValue("Description", Description);
        cmd.Parameters.AddWithValue("InstitutionsTypesId", InstitutionsTypesId);
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
      }
    }
  }

  public void DeleteInstitutionsTypesByInstitutionsTypesId(int InstitutionsTypesId) {
    string SqlString = "DELETE FROM InstitutionsTypes WHERE InstitutionsTypesId=" +
InstitutionsTypesId.ToString();
    using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
      using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
      }
    }
  }

 }
}

public class InstitutionsTypes {
 private int _Number;
 private int _InstitutionsTypesId;
 private string _InstitutionsTypesName;
 private string _Description;
 private string _Message;

 public InstitutionsTypes() {
  _Number = 0;
  _InstitutionsTypesId = 0;
```

```
    _InstitutionsTypesName = String.Empty;
    _Description = String.Empty;
    _Message = String.Empty;
   }

   public int Number {
    set { _Number = value; }
    get { return _Number; }
   }
   public int InstitutionsTypesId {
    set { _InstitutionsTypesId = value; }
    get { return _InstitutionsTypesId; }
   }
   public string InstitutionsTypesName {
    set { _InstitutionsTypesName = value; }
    get { return _InstitutionsTypesName; }
   }
   public string Description {
    set { _Description = value; }
    get { return _Description; }
   }
   public string Message {
    set { _Message = value; }
    get { return _Message; }
   }
}
```

Лістинг 15. Код класу « LogsProvider»

```
using SystemForBusinessService.AppCode;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.OleDb;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SystemForBusinessService.Provider {
  class LogsProvider {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];
    public void InsertLogs(int UsersId, string EventNameShow, DateTime EvendDate) {
      string SqlString = "INSERT INTO Logs (UsersId, EventNameShow, EvendDate) Values(?,
?, ?)";
      using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
        using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
          cmd.CommandType = CommandType.Text;
          cmd.Parameters.AddWithValue("UsersId", UsersId);
          cmd.Parameters.AddWithValue("EventNameShow", EventNameShow);
          cmd.Parameters.AddWithValue("EvendDate", EvendDate.ToString());
          conn.Open();
          cmd.ExecuteNonQuery();
          conn.Close();
```

```
      }
     }
    }

    public List<Logs> GetAllLogs() {
     int i = 0;
     string SqlString = "SELECT Logs.LogsId, Logs.UsersId, Logs.EventNameShow,
Logs.EvendDate, Users.UsersName " +
     "FROM Logs INNER JOIN Users ON Users.UsersId = Logs.UsersId  ORDER BY
Logs.EvendDate DESC";
     List<Logs> listAllLogs = new List<Logs>();
     using (OleDbConnection conn = new OleDbConnection(_ConnString)) {
      using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {
       conn.Open();
       using (OleDbDataReader reader = cmd.ExecuteReader()) {
        while (reader.Read()) {
         Logs oneLogs = new Logs();
         oneLogs.Number = ++i;
         oneLogs.LogsId = Convert.ToInt32(reader["LogsId"]);
         oneLogs.UsersId = Convert.ToInt32(reader["UsersId"]);
         oneLogs.EventNameShow = reader["EventNameShow"].ToString();
         oneLogs.EvendDate = Convert.ToDateTime(reader["EvendDate"]);
         oneLogs.UsersName = reader["UsersName"].ToString();
         listAllLogs.Add(oneLogs);
        }
       }
       conn.Close();
      }
     }

     if (listAllLogs.Count == 0) {
      Logs noLogs = new Logs();
      noLogs.LogsId = 0;
      noLogs.Message = NamesMy.NoDataNames.NoDataInLogs;
      listAllLogs.Add(noLogs);
     }
     return listAllLogs;

    }
   }
}


public class Logs {
  private int _Number;
  private int _LogsId;
  private int _UsersId;
  private string _UsersName;
  private string _EventNameShow;
  private DateTime _EvendDate;
  private string _Message;
```

```
  public Logs() {
    _Number = 0;
    _LogsId = 0;
    _UsersId = 0;
    _UsersName = String.Empty;
    _EventNameShow = String.Empty;
    _EvendDate = new DateTime();
    _Message = String.Empty;
  }

  public int Number {
    set { _Number = value; }
    get { return _Number; }
  }
  public int LogsId {
    set { _LogsId = value; }
    get { return _LogsId; }
  }
  public int UsersId {
    set { _UsersId = value; }
    get { return _UsersId; }
  }
  public string UsersName {
    set { _UsersName = value; }
    get { return _UsersName; }
  }
  public string EventNameShow {
    set { _EventNameShow = value; }
    get { return _EventNameShow; }
  }
  public DateTime EvendDate {
    set { _EvendDate = value; }
    get { return _EvendDate; }
  }
  public string Message {
    set { _Message = value; }
    get { return _Message; }
  }
}
```

★★★★★

Комп'ютерні інформаційні системи підтримки прийняття рішень для ресторанного бізнесу

Презентація

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ

★★★★★

Метою даної роботи є аналіз бізнес-процесів, пов'язаних з бізнесом обслуговування, як в кав'ярного так і ресторанного секторів, а також розробка програмного забезпечення, спрямованого на підвищення ефективності та збільшення обсягів продажу.

Об'єктом дослідження є інформаційна система підтримки прийняття рішень для бізнесу обслуговування в контексті кавової та ресторанної галузей.

Предметом дослідження є методи та засоби управління бізнесом обслуговування, що включають стратегії, технології, процеси та інструменти, які сприяють ефективному управлінню та оптимізації бізнес-процесів в сфері обслуговування в кавових та ресторанних підприємствах.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

1. На основі обраного методу був розроблений алгоритм для побудови прогнозу та оцінки його помилки. Цей алгоритм був успішно реалізований та впроваджений в інформаційну систему бізнесу обслуговування. Програма показала задовільну точність прогнозування та повністю відповідає вимогам, поставленим перед нею.

2. Розроблена система відповідає вимогам розробки, має інтерактивні елементи управління та інтуїтивно зрозумілий інтерфейс. Графічна оболонка має просту навігаційну структуру, що полегшує доступ до необхідної інформації.

Ресторанний та кав'ярний бізнеси представляють собою важливі напрямки, що активно розвиваються в контексті суспільного прогресу. Промисловість громадського харчування тісно пов'язана з ритмом сучасного життя. Швидкість, з якою людина функціонує в сучасному світі, постійно прискорюється, що робить все важчим витрачати час на приготування їжі. У таких обставинах заклади громадського харчування, зокрема ресторани та кафе, можуть бути надзвичайно корисними.

На рисунку у формі діаграми станів представлені всі основні стани системи, які відображають різні аспекти функціонування бізнесу

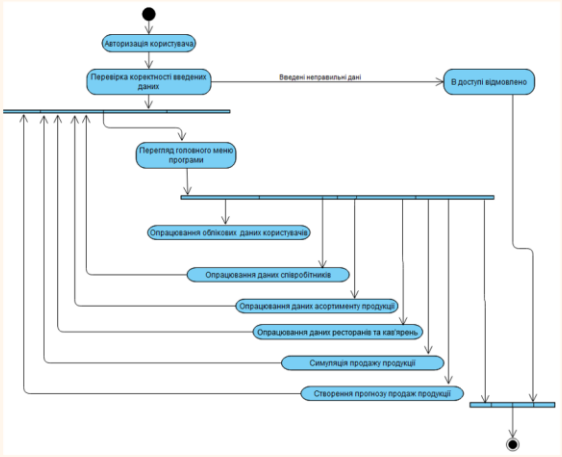
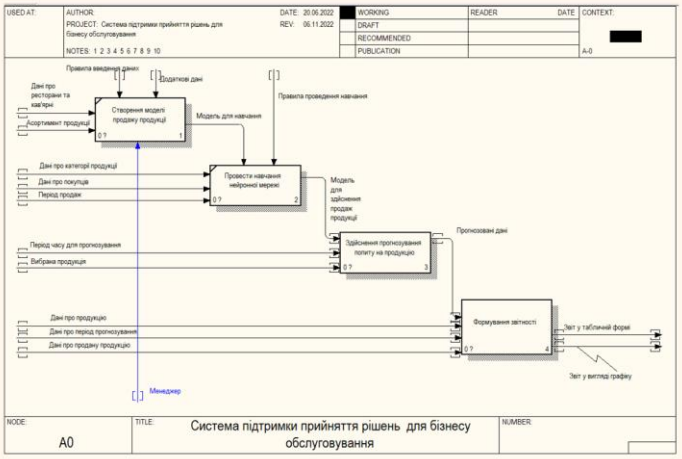
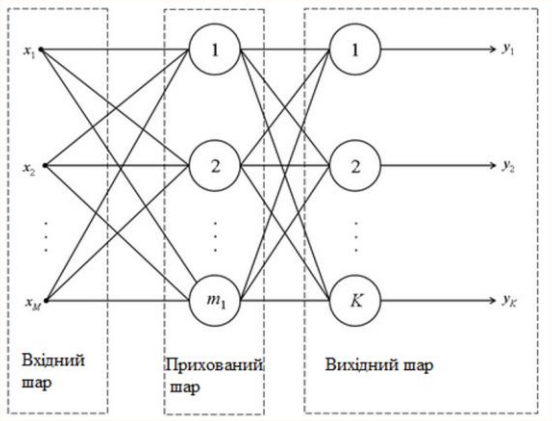


Рисунок демонструє діаграму декомпозиції моделі прогнозування попиту на продукцію, представлену в нотації *IDEFO*. Ця діаграма відображає структуру задач та взаємозв'язки між елементами системи та її зовнішнім середовищем



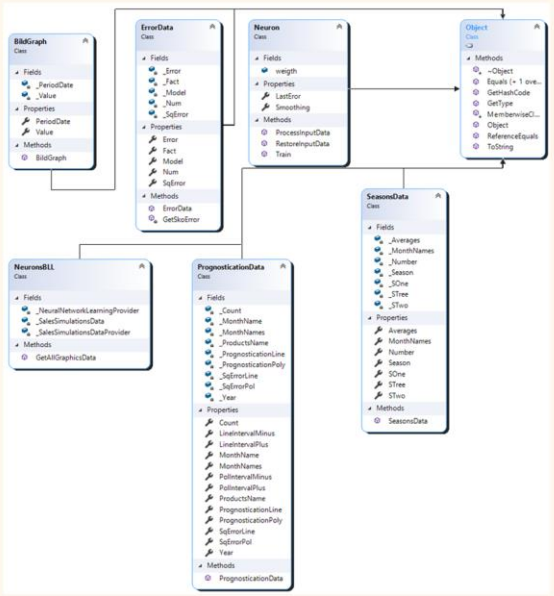
У разі необхідності застосування мережі складнішої структури до багатошарового перцептрона можна додати додаткові приховані шари або збільшити кількість нейронів у наявних прихованих шарах



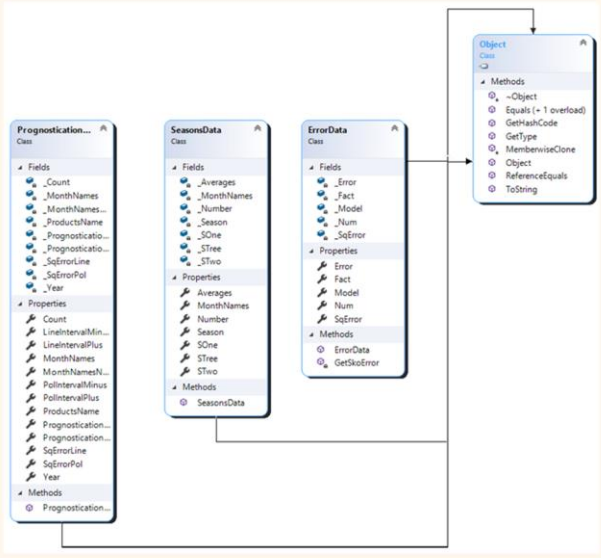
Параметри проведення навчання моделі прогнозування

Назва параметру	*Тип даних*	*Розмір*
Назва моделі	string	Рядок символів Unicode
К-сть епох навчання	int	32 біт
К-сть покупців	int	32 біт
Продукція	int	32 біт
Ціна	double	64 біт
Знижка	double	64 біт
Категорія	int	32 біт
Період продажу продукції	DateTime	Структура

Діаграма класів бізнес-логіки



Діаграма класів роботи алгоритму прогнозування



Дана робота була спрямована на розробку інформаційної системи для підтримки прийняття рішень в галузі бізнесу обслуговування. Розробка даної системи була виконана у середовищі Microsoft Visual Studio 2019 з використанням мови програмування C# та СУБД MS Access. Основною метою цієї системи було полегшити роботу менеджерів та аналітиків шляхом надання зручного перегляду даних, можливості додавання, видалення та редагування записів, а також здійснення пошуку та фільтрації за допомогою запитів.

У подальших дослідженнях можна розглянути врахування рідкісних подій, які час від часу відбуваються у світі, на рівні ймовірностей (наприклад, передсвяткові дні, введення карантину тощо). Це може значно змінити прогнозовані цифри порівняно з найкращими економічними математичними моделями.