Національний університет «Полтавська політехніка імені Юрія Кондратюка»

(повне найменування вищого навчального закладу)

Навчально-науковий інститут інформаційних технологій та робототехніки

(повна назва факультету)

Кафедра комп’ютерних та інформаційних технологій і систем

(повна назва кафедри)

**Пояснювальна записка**

**до дипломного проєкту (роботи)**

магістра

(рівень вищої освіти)

на тему

Розробка інформаційної інтелектуальної системи управління діяльності підприємства

Виконала: студентка 6 курсу, групи 601-ТН

спеціальності

\_\_\_\_\_\_\_\_\_\_\_\_122 Комп’ютерні науки\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва спеціальності)

Колесник Д.Д.

(прізвище та ініціали)

Керівник \_\_\_\_\_\_\_\_\_Фесенко Т.Г.

(прізвище та ініціали)

Рецензент \_\_\_\_\_\_\_\_

(прізвище та ініціали)

Полтава – 2022 року

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»**

**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ**

**КАФЕДРА КОМП’ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ**

**КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА**

**спеціальність 122 «Комп’ютерні науки»**

**на тему**

**«Розробка інформаційної інтелектуальної системи управління діяльності підприємства»**

**Студентки групи 601-ТН Колесник Дар’ї Дмитрівни**

Керівник роботи

доктор технічних наук,

професор Фесенко Т.Г.

Консультант

доктор педагогічних наук,

доцент Капітон А.М.

Завідувач кафедри

Полтава – 2022

**РЕФЕРАТ**

Кваліфікаційна робота магістра: 149 с., 32 малюнки, 5 таблиць, 1 додаток, 30 джерел.

**Об'єкт дослідження:** інтелектуальна системи управління діяльності підприємства.

**Предмет дослідження:** методи та засоби інтелектуального управління діяльністю підприємства.

**Мета дослідження:** розробка програмного забезпечення управління діяльністю підприємства, що дасть змогу впровадити у практику підприємством концепції «Інтелектуалізація управління», що в свою чергу є симбіозом двох понять – інновацій та інтелекту, що є запорукою успішного розвитку промислових підприємств в умовах ринкової економіки.

**Ключові слова:** підприємство, інтелектуальна система, автоматизація, бізнес процес, дочірні підприємства, управління.

**ABSTRACT**

Master`s qualification work: 149 p.,32 drawings, 5 tables, 1 appendix, 30 sources.

**Object of research:** intelligent management system of enterprise activity.

**The subject of research:** methods and means of intellectual management of enterprise activity.

**The purpose of the research:** the development of software for managing the company's activities, which will make it possible to put into practice the concept of "Intellectualization of management" by the company, which in turn is a symbiosis of two concepts - innovation and intelligence, which is the key to the successful development of industrial enterprises in the conditions of a market economy.

**Keywords:** enterprise, intelligent system, automation, business process, subsidiaries, management.

**ЗМІСТ**

[ВСТУП](#_Toc119745845) 8

РОЗДІЛ [1 ІНТЕЛЕКТУАЛІЗАЦІЯ УПРАВЛІННЯ ЯК НОВА ПАРАДИГМА УПРАВЛІННЯ ПІДПРИЄМСТВАМИ 11](#_Toc119745846)

[1.1 «Інтелектуалізація управління» як сформована дефініція 11](#_Toc119745847)

[1.2 Інтелектуалізація управління як невід'ємний елемент стратегічного розвитку підприємства та його успішного функціонування 17](#_Toc119745848)

[1.3 Інтелектуальний капітал підприємства – головна складова його системи знань 23](#_Toc119745849)

[1.4 Умови, необхідні упровадження концепції «Інтелектуалізація управління» у систему управління підприємствами 26](#_Toc119745850)

[1.5 Архітектура інтелектуальних систем 30](#_Toc119745851)

[1.5.1 Властивості знань 30](#_Toc119745852)

[1.5.2 Класифікація знань 32](#_Toc119745853)

[1.5.3 Бази знань 35](#_Toc119745854)

[1.5.4 Архітектура інтелектуальних систем 36](#_Toc119745855)

[1.6 Постановка задачі 37](#_Toc119745856)

[1.7 Висновок 38](#_Toc119745857)

РОЗДІЛ [2 ПРОЕКТУВАННЯ, РОЗРОБКА, ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ 41](#_Toc119745858)

[2.1 Аналіз вимог. Use-case діаграми. Основні прецеденти 41](#_Toc119745859)

[2.2 Архітектура проекту 52](#_Toc119745860)

[2.3 Особливості розробки бази даних. ERD з описанням сутностей 53](#_Toc119745861)

[2.4 Особливості розробки рівня BLL 55](#_Toc119745862)

[2.5 Особливість реалізації бізнес логіки – діаграма домена 57](#_Toc119745863)

[2.6 Особливості розробки рівня UI 58](#_Toc119745864)

[2.7 Особливості розробки DAL 61](#_Toc119745865)

[2.8 Висновок 63](#_Toc119745866)

РОЗДІЛ [3 ОСОБЛИВОСТІ ВИКОРИСТАННЯ ПРОГРАМНОГО ДОДАТКУ 65](#_Toc119745867)

[3.1 Вибір технологій 65](#_Toc119745868)

[3.2 Результати функціонального тестування розробленого додатку 68](#_Toc119745869)

[3.3 Інструкція користувачеві програми 71](#_Toc119745870)

[3.4 Висновок 81](#_Toc119745871)

[ВИСНОВКИ 84](#_Toc119745872)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 86](#_Toc119745873)

[ДОДАТОК А. Лістинги програм 89](#_Toc119745874)

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ**

**БЗ** – база знань.

**СУБЗ** – система управління базою знань.

**БД** – база даних.

**СУБД** – система управління базою даних.

**ПЗ** – програмне забезпечення.

**ERD** – діаграма сутність-зв’язок.

**UC** – діаграма прецедентів.

**REQ** – вимога.

**BLL** – рівень бізнес-логіки.

**DFD** – діаграма потоків даних.

# **ВСТУП**

Перехід багатьох країн на постіндустріальну та інформаційну щаблі розвитку призвів до зміни характеру вартості, що виробляється, і до підвищення значення в економіці інтелектуального потенціалу. У постіндустріальному суспільстві велику значущість набувають нематеріальні (організаційні, інтелектуальні) ресурси; Основним нематеріальним ресурсом соціально-історичного процесу завжди були і залишаються люди, і, отже, ключ до соціально-економічного прогресу в новому суспільстві лежить в активізації інтелектуального потенціалу на всіх рівнях соціального життя. На сьогоднішній день ефективна активізація є головним завданням управління.

Зі збільшенням обсягів використання інтелектуального капіталу підприємством зростає, у геометричній прогресії, позитивний ефект від його використання. Він є джерелом підвищення ефективності діяльності суб'єкта господарювання.

Отже, інтелектуальний капітал є одним із найбільш важливих ресурсів розвитку організацій незалежно від типу розв'язуваних завдань, місця у суспільному поділі праці, форми власності та рівня розвитку. Він є запорукою створення довготривалих конкурентних переваг, так званих відмінних компетенцій, набуття високих адаптивних якостей, що дозволяють пристосовуватися до надзвичайно мінливої кон'юнктури ринку.

У сучасних умовах перманентних змін у соціально-економічній та політичній сферах людської діяльності для того, щоб утримати позиції лідера, недостатньо використати стандартний менеджерський набір інструментів, необхідно взяти на озброєння так званий інноваційний підхід, суттю якого є використання інтелектуальних інструментів управління підприємством.

Одним із таких управлінських інноваційних інструментів може стати концепція «Інтелектуалізація управління», центральною ланкою якої є інтелектуальний капітал підприємства.

Аналіз сучасного стану інтелектуалізації управління свідчить, що практика застосування інтелектуальних технологій управління у різних країнах має різноманіття підходів і традицій, а також суперечливість кінцевих результатів. При цьому необхідно відзначити той факт, що в реальній діяльності підприємств подібний підхід практично не застосовується, тому доцільно провести аналіз світових тенденцій у цій галузі, виявити та систематизувати найкращі досягнення, максимально адаптувати їх до відповідних умов.

Нині «Інтелектуалізація управління» робить лише перші кроки. Перспективи її розвитку пов'язані з інтеграцією трьох найважливіших концепцій. Це концепції інтелектуального капіталу, менеджменту знань та організації, що навчається, які не просто перетинаються, а й взаємно доповнюють одна одну.

**Актуальність теми** полягає в тому, що в нинішніх умовах початку ринкових відносин потрібен пошук кращих управлінських рішень та вдосконалення системи управління діяльності підприємства.

**Мета роботи**  – розробка програмного забезпечення управління діяльністю підприємства, що дасть змогу впровадити у практику підприємством концепції «Інтелектуалізація управління», що в свою чергу є симбіозом двох понять– інновацій та інтелекту.

Для досягнення зазначеної мети було поставлено такі завдання:

* провести дослідження предметної області;
* на основі отриманих результатів аналізу даних зробити постановку завдання до майбутньої системи;
* здійснити розробку програмного забезпечення;
* провести тестування розробленої системи.

**Об'єктом дослідження** є інтелектуальна системи управління діяльності підприємства.

**Предметом дослідження** є методи та засоби інтелектуального управління діяльністю підприємства.

**Методи**:

* аналіз методів інтелектуального управління;
* дослідження методів стратегічного розвитку підприємства та його успішного функціонування;
* дослідити архітектуру інтелектуальних систем;
* проаналізувати проблеми та вимоги, які будуть стояти перед розробкою програмного забезпечення.

**Наукова новизна.** На основі дослідженого матеріалу було розроблено продукт для інтелектуальної системи управління діяльності підприємства. Головною ідеєю розробки даного продукту було безкоштовність та масовість у використанні, так як даний продукт можна легко переносити на будь-які операційні системи сімейства Windows.

# **РОЗДІЛ 1**

# **ІНТЕЛЕКТУАЛІЗАЦІЯ УПРАВЛІННЯ ЯК НОВА ПАРАДИГМА УПРАВЛІННЯ ПІДПРИЄМСТВАМИ**

## **«Інтелектуалізація управління» як сформована дефініція**

«Інтелектуалізація управління» – у чому суть цього терміну? Звернемося насамперед до самого поняття «управління» та його ролі в економічному розвитку як окремих підприємств, так і держави загалом.

Управління – це мистецтво виконання завдань з допомогою окремих чи формально організованих груп. Це мистецтво створення такого середовища, в якому люди можуть діяти як особи і, крім того, кооперуватися задля досягнення цілей групи. Це шлях оптимізації ефективності праці та досягнення цілей [1].

Управління є найважливішим ресурсом нації. На думку патріарха американського менеджменту Пітера Друкера, історичні успіхи нації на 80% визначаються не природними ресурсами та технологіями, а ефективністю управління. Тому одні країни прогресують, інші ні. Однак, як свідчить світовий досвід, така роль управління може бути реалізована лише за сприятливих внутрішніх та зовнішніх умов. Це насамперед опора на об'єктивні економічні закони, випробувані засади ринкового господарювання, світова практика успішного ведення бізнесу, здоровий глузд.

Як основний критерій ринкової ефективності сьогодні приймається не прибутковість, а ступінь задоволення інтересів конкретних споживачів. Управлінські кадри в організаціях розглядаються як головний ресурс знань та вмінь, як своєрідні центри певних високопрофесійних здібностей. При цьому вже не підприємство диктує свої правила гри, а працівники-професіонали задають тон їхнім взаєминам.

Сьогодні людина розглядається як головний об'єкт інтересів менеджменту. Завдання керівництва при цьому полягає у проведенні такої кадрової політики, коли метою всієї діяльності з управління персоналом стає найкраще задоволення зростаючих запитів та потреб кожного члена організації. Досі при всіх міркуваннях про важливість кадрів і створення найбільш сприятливих умов розкриття потенціалу кожного співробітника він розглядався як об'єкт, який підлягає управлінню з боку фірми. Новий підхід передбачає, що людина є змінною, але некерованою змінною підприємства. Завдання менеджменту в цьому випадку в тому, щоб пізнати закономірності розвитку особистості та розробити відповідні програми, що забезпечують краще, ніж конкуренти, задоволення потреб кожного працівника. Результатом цієї діяльності має стати збільшення прибутку, як наслідок зростання продуктивність праці.

Таким чином, змінюється система поглядів на людину в організації: не її змушують бігати, щоб забезпечити продуктивну роботу, а фірма крутиться навколо співробітника, намагаючись якнайкраще задовольнити його запити, що зростають [2]. Можливо, спочатку це здається дивним (так само, як при переході від адміністративної системи до ринкової спочатку здавалося дивним, що не фірма диктує покупцеві свою волю, а він їй). Хто не зможе цього вчасно зрозуміти, може виявитися неконкурентоспроможним на ринку, особливо зважаючи на складну демографічну обстановку в країні в цілому та на ринку праці зокрема.

Стрижневим елементом системи управління економікою є люди. Країні потрібна когорта професійних управлінців, здатних вивести її із кризи. Вони повинні мати принципово нові якості. В першу чергу це вміння інтегрувати інтелект людей, пробуджувати в них ініціативу, творчість, новаторство, прагнення успіху, оригінальним, самостійним діям і рішенням, а також здатність залучати в організації професіоналів і утримувати їх, формувати атмосферу справжньої зацікавленості всього персоналу в успіху працювати лише на рівні високих стандартів.

Головними вимогами до менеджерів-керівників стають високий професіоналізм, знання механізмів функціонування ринку разом із володінням сучасними інформаційними технологіями, зокрема використання з цією метою мережі Internet. Розв'язання цього завдання пов'язане не лише з поглибленням підготовки в школах та вузах у галузі економіки, бізнесу та менеджменту, а й зі створення суспільної структури (асоціації) зарубіжних країн щодо формування в країні інституту (категорії) професійних менеджерів.

Критеріями оцінки менеджерів мають бути творчий склад розуму, стратегічний тип мислення, схильність до інновацій, вміння акумулювати енергію та здібності до спільної діяльності. Інтелектуальна культура спеціаліста управління включає здатність до методичного мислення, як фактора високої результативності цієї діяльності. Це означає, що менеджер впевнено володіє сучасними методами програмно-цільового, інформаційно-аналітичного, прогнозного та проектного управління.

Ефективний менеджмент базується на знаннях, вміннях та інформації. Тому роль підготовки фахівців у цій галузі постійно має зростати, наповнюватися сучасними ідеями та підходами.

Нові тенденції у науковому менеджменті вимагатимуть нетрадиційних підходів у справі підвищення економічної та управлінської культури менеджерів.

У сучасному світі швидкими темпами може розвиватися тільки та країна, яка створить сприятливі можливості для максимального зростання інтелекту фахівців у галузі менеджменту та загалом інтелектуального потенціалу управління.

Стає зрозумілою актуальність та важливість використання інтелектуального підходу до управління. Інтелектуалізація управління це, безумовно, принципово нові методи управління, що мають яскраво виражений інноваційний характер, де ключовим поняттям є людський потенціал та ефективність його використання. Саме людський чинник є основою розвитку підприємства у довгостроковому періоді. Стратегія полягає в індивіді, який, своєю чергою, визначає пріоритетні напрями розвитку, необхідний інструментарій і, власне, втілює цю стратегію у життя.

Інтелектуалізація управління є необхідною умовою не тільки «виживання» господарюючого суб'єкта в умовах ринкової економіки та найжорстокішої конкуренції, але й насамперед невід'ємним фактором, що визначає його ефективне функціонування та розвиток на завтра.

Таким чином, підсумовуючи все сказане вище, можна сформулювати наступне: впровадження у практику менеджменту концепції «Інтелектуалізація управління» потребує використання спеціального інструментарію, пріоритетним призначенням якого є всебічний розвиток інтелектуального потенціалу підприємства. Детально мають бути вивчені такі інструменти інтелектуального розвитку. Насамперед, це система інтелектуального капіталу компанії, що включає безпосередньо інтелектуальний капітал (всі його складові) і систему оцінки його вартості. В рамках вивчення системи інтелектуального капіталу необхідно приділити особливу увагу його складовій – людині, і методам її оцінки. Другою важливою ланкою концепції "Інтелектуалізація управління" є менеджмент знань, створення бази знань, її розширення та примноження.

Таким чином, «Інтелектуалізація управління» є симбіозом двох ключових понять економіки XXI століття – інновацій та інтелекту, синергетичний ефект яких є запорукою успішного розвитку промислових підприємств та вузів в умовах ринкової економіки.

Розглянемо складові концепції "Інтелектуалізація управління" (рис. 1.1).

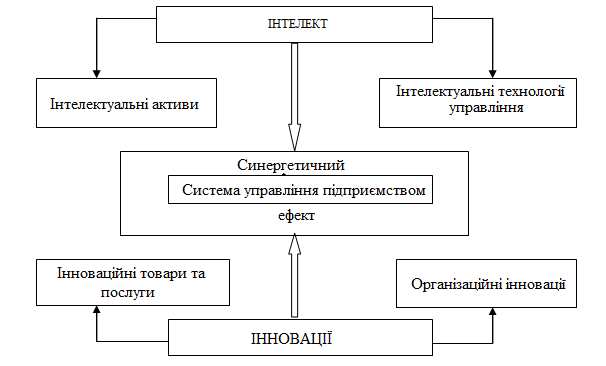


Рисунок 1.1 – Інтелектуалізація управління

Зупинимося докладніше на понятті синергетичного ефекту, який розкривається в роботі як складний системний синтез інновацій та інтелектуальних технологій управління. Синергетичний ефект – це ефект нелінійності в складних системах, що еволюціонують і саморегулюються, якими є економіко-управлінські комплекси, включаючи підприємства. Ці системи досліджуються як відкриті, тобто незамкнені, що активно взаємодіють з оточенням, шляхом обміну ресурсами, енергіями та інформацією. З переходом до інформаційного суспільства, загостренням конкуренції, посиленням процесів глобалізації увага до таких систем економіки різко зростає. Їх поведінка має принципово незворотний характер, отже отримання скільки-небудь істотного проміжного результату змінює ймовірності майбутніх станів, перетворюючи можливе раніше у неможливе чи створюючи нові можливості. Однак управління в таких системах з раціональної точки зору незручно тим, що зводить до мінімуму звичне в раціональних моделях центрування та адміністрування, виходить з уявлення про децентралізацію та розподіл владних повноважень, надаючи особливої цінності освіті, інтелекту, інноваційній політиці, різноманітності способів використання. Особливої ролі набуває інтуїція керівника будь-якого рангу, оскільки інтелектуальна модель управління допускає властиві синергетичним процесам елементи виникнення порядку з хаосу, отже, і ринкового регулювання, передбачає спочатку властиву відкритим системам здатність успішно функціонувати у вкрай нерівноважних станах.

Звідси і виникає гостра потреба в інтелектуалізації управління підприємством та опорі на постійні інновації. Певною мірою управління «розпливається» по всьому простору виробництва, перестає бути жорстко локалізованим, а тому передбачає істотну демократизацію менеджменту, високий рівень професіоналізму та інтелектуалізації на кожному рівні або підрівні виконання управлінських функцій[3].

## **1.2** **Інтелектуалізація управління як невід'ємний елемент стратегічного розвитку підприємства та його успішного функціонування**

Компанії третього тисячоліття мають істотні відмінності від своїх попередників, оскільки найбільшу цінність їм представляють працівники, а точніше, їх знання. «Скарбничка колективних знань та здібностей» компанії дає величезну перевагу в порівнянні з конкурентами у разі гідної підтримки та розвитку власних співробітників, розробки довгострокових програм у цьому напрямку та послідовної їх реалізації.

Оскільки центральною ланкою поняття «інтелектуалізація управління» насамперед є людські ресурси підприємства, проте інші чинники розвитку інтелектуального потенціалу похідні від нього, то проаналізуємо вплив управління людськими ресурсами на стратегічний розвиток підприємства[4].

На рис. 1.2 продемонстровано комплекс планів, що становлять основу стратегічного планування діяльності підприємства.

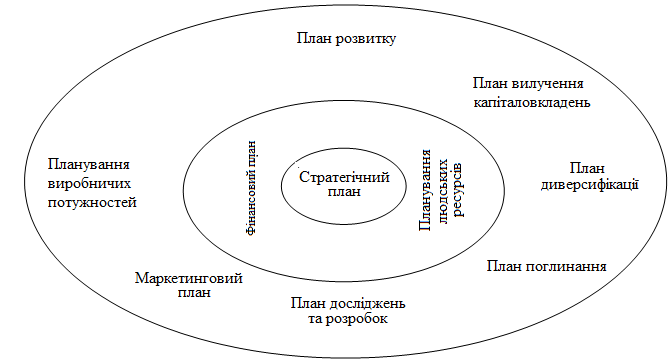


Рисунок 1.2 – Узагальнена схема стратегічного планування

Діаграма представлена у формі кола для того, щоб продемонструвати, що жоден з цих планів не існує сам по собі. Усі вони є взаємозалежними та взаємо-доповняльними. У центрі системи знаходиться стратегічний план, який є ключовою ланкою всього процесу, він визначає напрями розвитку, звідси виходять всі стратегічні директиви операційним підрозділам компанії. Примітно, що наступними за значущістю є фінансове планування та планування людських ресурсів. Причому останнє має яскраво виражений стратегічний характер, тоді як фінансове планування переважно є прерогативою операційного менеджменту. У цьому сумний той факт, що переважна більшість підприємств не приділяє практично жодної уваги людської складової своєї діяльності. Звідси й актуальність цієї проблеми, її нагальність.

Оскільки поки що не зміниться ставлення до кадрового менеджменту та всіх його складових, ні про який стратегічний розвиток та ефективне функціонування не може бути й мови. Досягнення будь-якої сукупності корпоративних цілей потребує використання двох основних ресурсів: людських та фінансових. Нестача будь-якого з цих ресурсів може перетворити оптимально продуману стратегію на нездійсненний повітряний замок.

На жаль, багато організацій вважають розробку стратегії, пов'язаної з людськими ресурсами, справою надмірно клопіткою, але насправді ця стратегія тісно пов'язана з бізнес-стратегією і є необхідною. Якщо обмежуючий вплив нестачі капіталу досить легко передбачити, то тиск, викликаний дефіцитом людських ресурсів, менш очевидний. Потреба у людських ресурсах має як кількісні, так і якісні характеристики.

Завдання стратегічного планування у сфері людських ресурсів складніше, ніж завдання задоволення потреб підприємства у капіталі.

Стратегічне планування людських ресурсів може бути результативним лише тому випадку, якщо його функція розглядається нарівні з іншими. Менеджери з персоналу мають бути повністю обізнаними у сфері бізнес-планів, оскільки вони можуть радикально змінити розуміння кадрових питань. Підрозділ управління персоналом, який розглядається виключно в ролі архіву і який не бере участі в процесі прийняття рішень, навряд чи буде здатний зробити результативний внесок у досягнення корпоративних цілей. Щоб відігравати важливу роль у майбутньому компанії, управління людськими ресурсами має займати повноправне місце за столом менеджменту та отримувати всеосяжну інформацію та право голосу.

Відповідальність за питання управління персоналом несуть усі менеджери, і ці питання необхідно розглядати у зв'язку з іншими релевантними факторами прийняття рішень. На додаток до цього на управлінні персоналом лежить особливий обов'язок відстежувати, щоб проблеми управління персоналом були вплетені в структуру кожного стратегічного рішення і кожен аспект політики управління персоналом працював на ефективність спільної стратегії.

Розвиток бізнесу згідно з його життєвим циклом вносить нові вимоги до завдань, роботи та типу людей, необхідних для цієї роботи. Менеджер, який є чудовим керуючим на стадії початку діяльності компанії, може виявитися нездатним керувати великою та розвиненою організацією. При керуванні портфелем бізнес-одиниць від різних команд менеджерів потрібні різні навички та вміння. Наприклад, потрібно мати різні знання, щоб керувати бізнесом, який є генератором прибутку, і новим ризиковим проектом, який у перспективі очікує бурхливе зростання. Вміння передбачати, які бізнес-одиниці та напрямки діяльності розроблятимуть компанія в майбутньому і працівники якої кваліфікації для цього будуть потрібні, – ключ до успішного розвитку компанії. У цьому важливий як рівень кваліфікації конкретного працівника, так і певний набір особистісних показників, необхідні виконання тієї чи іншої роботи.

Стратегія управління людськими ресурсами у вигляді представлена на рис. 1.3.

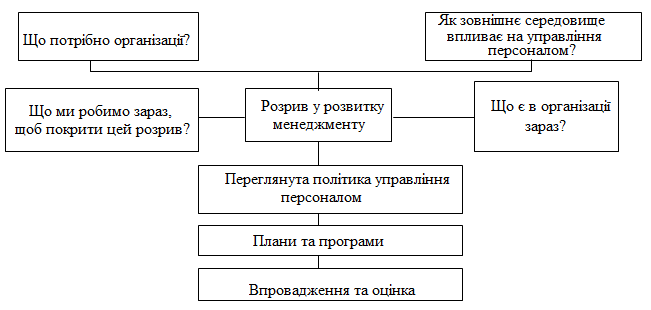


Рисунок 1.3 – Стратегія управління людськими ресурсами

Стратегія управління людськими ресурсами повинна брати до уваги стратегічні потреби організації та будь-які тенденції довкілля, які можуть вплинути на людські ресурси. Вона повинна ретельно досліджувати ті стратегії та політики, яким слідує організація, і оцінювати розрив між тим, що робиться, і тим, що потрібно робити.

Один із важливих моментів – розгляд впливу стратегії на людей. У рамках вирішення цього завдання беруться ключові моменти бачення, цінностей, цілей і стратегії та оцінюються деякі з компетенцій менеджменту, необхідні для успіху. Далі слід розглядати ті області, де може виникнути дефіцит[4].

Схема планування майбутніх потреб у трудових ресурсах представлена на рис. 1.4.

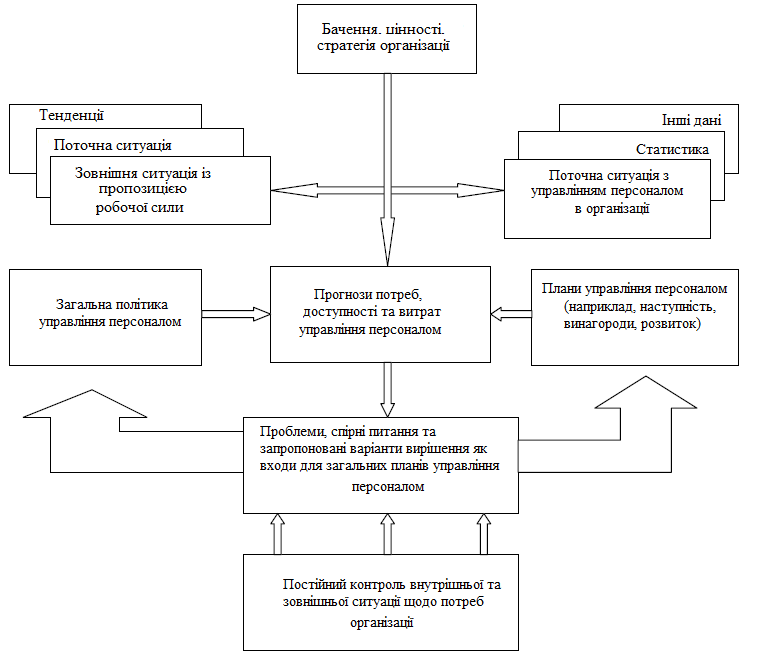


Рисунок 1.4 – Схема планування майбутніх потреб у людських ресурсах

Одним із способів вирішення проблеми планування потреб у трудових ресурсах є складання таблиці наступності менеджменту на кілька років уперед. Тут записуються дати можливих поступів кожного менеджера, називаються імена одного чи двох можливих наступників, а також показаний граничний рівень, якого може досягти кожна людина. Там, де співробітник не має подальшого потенціалу просування по службі, на підставі особистих якостей або віку, це має бути чітко зафіксовано. Здатність робити такі оцінки вимагає дієвих і чітких способів визначення персональних характеристик співробітника.

Планування людських ресурсів має надзвичайно важливе стратегічне значення, оскільки нездатність фірми забезпечити себе робочою силою, що відповідає її вимогам, може призвести до невдачі в досягненні корпоративних цілей.

Основні необхідні статистичні дані включають аналіз кількості працівників за окремими категоріями робіт та за місцем розташування. Система класифікації робіт може бути суто внутрішньою, а може бути заснована на опублікованих державних єдиних тарифно-кваліфікаційних довідниках для того, щоб була можливість порівняти її із національною статистикою. Кожну категорію необхідно розбивати за статтю та віком. Для кожної з цих підгруп необхідно проводити аналіз співробітників, які залишають фірму. Бажано проводити також їхню класифікацію залежно від причин догляду. З цих даних можна зробити важливі висновки. Організація може виявити, що її робоча сила старіє, і в цьому випадку можна передбачити потенційний дефіцит співробітників, пов'язаний з виходами на пенсію, що наближаються. Також можуть бути виявлені такі групи працівників, де відсоток звільнень буде більшим за нормальний рівень, таким чином, можна виявити помилки, що стосуються методів управління персоналом та умов найму.

Крім того, мають бути проведені заходи, що оцінюють середню заробітну плату та діапазон тарифних ставок, продуктивність праці та динаміку її зміни.

Знаючи ці дані, менеджер з персоналу повинен зіставити отримані результати з організаційною стратегією розвитку, щоб оцінити, наскільки може змінитися природа операцій протягом найближчих років. Нові починання, розширення бізнесу та будь-які сфери, де компанії доведеться згорнути свою діяльність, впливатимуть на майбутні потреби організації у персоналі, і цей вплив має бути виражений у кількості людей за кожною з категорій робіт.

Має бути спрогнозовано очікуване зростання продуктивності праці, частково – з тенденцій, частково – з навмисних дій, які є результатом стратегії. Крім того, необхідно брати до уваги будь-які передбачувані зміни у співвідношенні постійних та тимчасово зайнятих співробітників, а також має бути досліджено вплив будь-яких рішень щодо укладання суб-контрактів. Все це разом дозволяє розробити добре обґрунтований прогноз необхідної кількості людей.

Прогнози забезпечують організацію оцінкою того, скільки людей має бути доступно по кожному виду роботи в конкретний період. Щоб від цього була відчутна користь, необхідно робити подальші прогнози, беручи до уваги відходи на пенсію, смертність та показники звільнень, щоб компанія могла бачити, скільки людей буде в наявності по кожній категорії.

Наступний етап планування – оцінка того, як задовольнятимуться потреби у персоналі. Прогнози можуть виявити сферу, де зміни у методах навчання дозволять співробітникам перейти з однієї категорії до іншої. Менеджер з персоналу повинен оцінити, скільки вакансій буде заповнено внаслідок внутрішньо-фірмового просування працівників. У результаті порівняння зі статистикою всередині країни повинні бути виявлені ті сфери, де наймання нових співробітників досить легке, а де воно може бути пов'язане з певними труднощами (в даному випадку має бути здійснено низку превентивних дій, спрямованих на скорочення потреби в трудових ресурсах дефіцитного типу) .

Прогнозовані потреби у робочої сили можуть виявити зміни у типі найманих співробітників, у переліку вимог, що висуваються до них, у кваліфікаційно-особистісних характеристиках, необхідні певних посад.

При вирішенні своїх кадрових проблем багато організацій вдаються до спеціальних заходів впливу.

По-перше, це дії щодо створення позитивного іміджу підприємства в очах громадськості.

По-друге, це розробка гнучкіших систем оплати праці та преміювання, метою яких є відмінне позиціонування себе на ринку праці в порівнянні з конкурентами. На сьогоднішній день кваліфіковані трудові ресурси є найбільш обмеженим фактором виробництва, а, беручи до уваги демографічну ситуацію, що склалася в країні і в Свердловській області зокрема, можна припустити, що в найближчому майбутньому підприємства почнуть активну конкурентну боротьбу на ринку праці, в тому числі і за молодих спеціалістів.

Третім напрямком діяльності підприємств щодо вирішення кадрових проблем може стати налагодження довгострокового співробітництва підприємств та вищої школи у сфері підготовки та перепідготовки висококваліфікованих кадрів.

Тому за сучасних умов необхідно детально вивчити основні інструменти управління інтелектуальним потенціалом підприємства[5].

## **1.3** **Інтелектуальний капітал підприємства – головна складова його системи знань**

Поняття інтелектуального капіталу є значущим для нової економіки. У сучасному етапі технологічного розвитку значення поняття "інтелектуальний капітал" дозволяє говорити про корінну відмінність нової економіки від економіки промислової індустрії, що спирається на природно-сировинні ресурси та працю так званого промислово-виробничого персоналу.

Останнім часом все частіше на підприємствах постає питання про управління всім комплексом інтелектуальних ресурсів як єдиним цілим і, зокрема, підвищення ефективності праці творчих працівників. Дедалі гостріше відчувається необхідність збільшення продуктивності своєї діяльності. Без нових ідей бізнес не розвивається і швидко морально старіє.

Людський мозок вважається найпотужнішим «обладнанням» будь-якого підприємства, що дозволяє вирішувати будь-які проблеми. І саме тому сучасні менеджери намагаються сфокусувати свої зусилля на створенні умов розвитку творчого потенціалу персоналу своїх підприємств.

Якщо керівники підприємства прагнуть досягти переконливого й стійкого успіху в економічних умовах, вони неминуче повинні як пріоритет вибрати нарощування інтелектуальних переваг своєї організації. Розвиток інтелектуального чинника значно ефективніший, ніж, наприклад, вдосконалення організації використання різних застарілих технологій та використання капіталу компанії для цього. Це, зрештою, приносить значно більший прибуток і дозволяє в майбутньому досягти більшого успіху.

За даними експертів, більшість підприємств використовує лише 20% від наявного у них інтелектуального капіталу. Інші 80% являють собою втрати компанії, втрачену вигоду від недовикористання інтелектуального потенціалу.

Експерти та дослідники інтелектуального капіталу пропонують кілька способів його класифікації. Причому кожному конкретному випадку вибір, зазвичай, обумовлюється специфікою діяльності організації, яка робить спроби структурувати свій інтелектуальний капітал. Наприклад, одні дослідники поділяють інтелектуальний капітал активну і пасивну составляющие1. Активна раніше мала позначення – робоча сила, трудовий потенціал, кадри, персонал, інженерно-технічні працівники, службовці.

За кожним із названих визначень на підприємствах ця категорія капіталу структурувалася за спеціальностями, кваліфікацією та посадовим рівнем. Еквівалентом поняття «активна складова» сьогодні у країнах із розвиненою економікою є поняття «людський капітал», що з'явилося в США у 70-ті роки. Під ним американці пропонують розуміти освіту, кваліфікацію, набуті у процесі трудової діяльності, а також навички, досвід та знання, втілені у персоналі підприємства. До пасивної складової інтелектуального капіталу належать права на технології, винаходи, промислові зразки, товарні знаки, ноу-хау, тобто те, що називається інтелектуальною власністю.

Інші експерти, до яких належить провідна світова страхова та фінансова компанія Skandia – піонер у галузі вивчення інтелектуального капіталу, розглядають інтелектуальний капітал як [3]:

* «людський капітал» – сукупність знань, практичних навичок та творчих здібностей службовців компанії, прикладена до виконання поточних завдань. Іншими його складовими є моральні цінності підприємства, культура праці та загальний підхід до справи. Характерною особливістю і те, що людський капітал може бути власністю підприємства;
* «структурний капітал» у цю категорію входять технічне та програмне забезпечення, організаційна структура, патенти, торгові марки та все те, що дозволяє працівникам компанії реалізувати свій виробничий потенціал іншими словами, те, що залишається в офісі після догляду службовців додому. Структурний капітал також включає відносини, що склалися між компанією та її великими клієнтами. На відміну від людського капіталу, структурний може бути власністю компанії, а отже, і об'єктом купівлі-продажу.

Інтелектуальний капітал це – умови розвитку компанії, що приховані за видимим фасадом її будівель та товарного асортименту. У цьому полягає особливе значення інтелектуального капіталу. Інтелектуальний капітал – це знання, які мають потенційну цінність, тобто ідеї.

Під інтелектуальним капіталом розуміється сукупність наявних у підприємства законних прав на результати його творчої діяльності, його природних та набутих інтелектуальних здібностей та навичок, а також накопичених ним баз знань та корисних відносин з іншими суб'єктами. Цінність інтелектуального капіталу завжди має розглядатися у співвіднесенні з результатами його інтелектуальної діяльності і на цій підставі може оцінюватись відповідною вартістю. Інтелектуальний капітал є провідним капіталом та становить основу будь-якого підприємства. Головна функція інтелектуального капіталу – суттєво прискорити приріст маси прибутку за рахунок формування та реалізації необхідних підприємству систем знань, речей та відносин, які у свою чергу забезпечують його високоефективну господарську діяльність. Зокрема, інтелектуальний капітал підприємства визначає якість системи управління. За припущеннями експертів, найближчим десятиліттям основи менеджменту повністю базуватимуться на теорії управління інтелектуальним капіталом.

Інтелектуальні здібності, які дуже складно виразити цифрами, одиницями виміру, є динамічною складовою інтелектуального капіталу. Їх слід розглядати як реальні та потенційні можливості окремих осіб чи груп проводити ті чи інші складні роботи чи операції. Інтелектуальні здібності вимагають постійного розвитку, тренінгу, і тому менш стабільні і важче піддаються виміру. На Заході цю складову включають до людського капіталу.

Проаналізувавши різні джерела інформації, можна запропонувати наступну класифікацію складових інтелектуального капіталу (рис. 1.5).

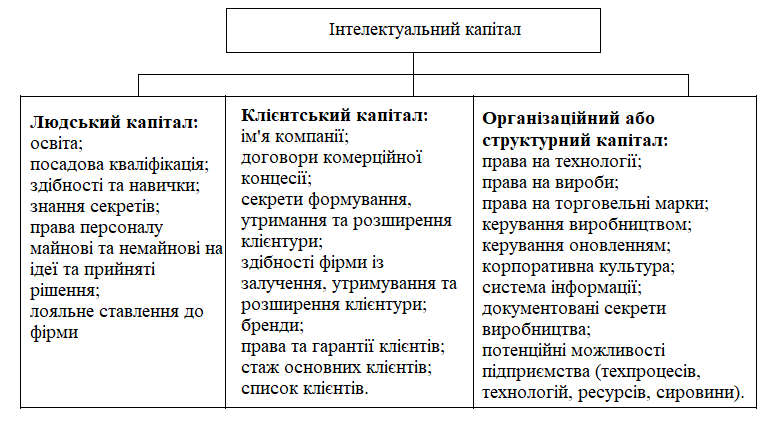


Рисунок 1.5 – Структура інтелектуального капіталу

Людський капітал є ключовою складовою інтелектуального капіталу, оскільки саме він є основою для розвитку як клієнтського, так і організаційного капіталу.

Клієнтський капітал - капітал відносин - це система капітальних, надійних, довгострокових, довірчих та взаємовигідних відносин компанії зі своїми клієнтами, споживачами її продукції, що склалася за роки її роботи на ринку. По суті це комплексний складний нематеріальний актив. Сюди відносяться бренди, клієнтура, фірмове найменування, відданість бренду, канали збуту, система ділових зв’язків, ліцензійні договори, опціонні угоди, договори комерційної концесії та інші.

Організаційний (структурний) капітал. Структурний капітал легше зрозуміти шляхом виключення. Все, що залишається на підприємстві, пов'язане з інтелектуальним потенціалом, після відходу персоналу з роботи додому, це і є організаційний (структурний) капітал. Сюди належить не лише інтелектуальна власність. До його складу, наприклад, входять філософія та процеси менеджменту, інформаційні системи, система фінансових взаємин, інструкції, положення, стандарти підприємства та інша організаційна документація, що дозволяє організувати виробництво та ефективно ним керувати[7].

## **1.4 Умови, необхідні упровадження концепції «Інтелектуалізація управління» у систему управління підприємствами**

Ключове питання в контексті впровадження концепції «Інтелектуалізація управління» в реальну практику – це те, якими властивостями повинні володіти організації (фірми, концерни, підприємства), щоб забезпечувати постійне пристосування до навколишнього середовища, що динамічно змінюється. У процесі перетворення навколишніх умов і об’єктивних вимог організація як адаптований механізм буде змінюватися.

Узагальнюючи процеси, що відбуваються, і тенденції, що намітилися, можна зробити висновок, що головними умовами, виконання яких необхідне для впровадження концепції «Інтелектуалізація управління» в систему стратегічного менеджменту, будуть наступні, представлені на рис. 1.6.



Рисунок 1.6 – Умови впровадження концепції «Інтелектуалізація управління» у систему управління підприємством

Детальніше всі ці умови розглянемо нижче:

1. Компанія повинна мати велику гнучкість. Організація має будуватися на засадах гнучкості. Її основними рисами стають динамізм та адаптація, як того вимагають зміни навколишнього середовища, що, безумовно, викликає суттєві зміни й у поведінці її клієнтів. В цьому перетворення мають торкнутися якісь другорядні елементи, а сутнісне «ядро» управління – структури, процеси, методи. Вони змінюються тим швидше, що вищі темпи зміни техніки, технології, умов розвитку ділової активності. Навіть процвітаючі компанії можуть постраждати від власного консерватизму. Відомі випадки, коли компаніям після скорочення десятків тисяч робочих місць доводилося заповнювати нові десятки тисяч робочих місць, які вимагають від працівників нових навичок та умінь. Загалом ці процеси спрямовані на прискорення та поглиблення змін у самій основі організації – у суті її діяльності.
2. Зміна ролі окремих працівників. Традиційний підхід до роботи індивіда у різних модифікаціях націлений на постійне зростання оплати праці, підвищення відповідальності та безпеки. Організація робить ставку на результат, а не на процес роботи, індивід же керується прагненням до цікавої роботи та професійного зростання незалежно від того, відбувається це в одній або в низці організацій. Організація майбутнього покликана подолати зазначену суперечність інтересів, оскільки мінливість соціального середовища створює умови, що дозволяють і організації, та працівникові добиватися єдності та високого рівня виконання своїх зобов’язань.
3. Переважне використання груп (команд). Серед навичок дедалі більшу вимогу організація пред’являє до вміння працювати у групах, які у перспективі почнуть відігравати у всіх видах діяльності. Самонацілені, самоврядні команди стають ключовими ланками, м’язами гнучкої організації. У міру зміни середовища ділової активності організація адаптується, а внутрішня організаційна структура є настільки гнучкою, що «вбирає» всі зміни.
4. Висока внутрішня конкурентоспроможність. Сила організації – у її глибинної конкурентоспроможності, у професійних знаннях працівників, дозволяють їм здійснювати діяльність якнайкраще. Компанія збирає воєдино весь свій потенціал, забезпечує раціональну його організацію та використання у повній відповідності до вимог клієнтів, ринку. Якщо керівництво компанії не орієнтує на це всі функції управління, останні не забезпечуватимуться ресурсами і фінансуватися. Зокрема, все частіше визнається, що однією з ключових конкурентоспроможних якостей є здатність адаптуватися до потреб клієнтів та надавати їм інноваційні послуги – організаційні, технологічні чи структурні.
5. Прагнення диверсифікації. Перспективна компанія має прагнути диверсифікації своєї робочої сили та своїх постачальників. Диверсифікація використовується для підвищення конкурентоспроможності та розширення спектру послуг, що надаються клієнтам. Всім працівникам мають бути забезпечені умови для того, щоб вони брали участь у намічених змінах та вносили в них свій внесок.
6. Компанія має бути досить зрілою і вже перетнути той рубіж, коли ефективність компанії обговорюється лише в термінах фінансів. У даному випадку йдеться про подолання початкової стадії, коли на чільне місце ставилося досягнення певних фінансових результатів. Концепція «Інтелектуалізація управління» передбачає інший погляд на стан речей і її впровадження необхідний принципово інший рівень розвитку організації.
7. Концепція «Інтелектуалізація управління» має бути глибоко впроваджена у бачення компанії. Напрям, ціль, бачення та місія компанії для концепції «Інтелектуалізація управління» завжди є відправною точкою, незалежно від галузевої спрямованості підприємства. Тому зусилля щодо впровадження інтелектуалізації управління мають становити саму основу ділового бачення, місії чи стратегії компанії.
8. Концепція «Інтелектуалізація управління» має бути впроваджена у мову компанії. Важливі поняття, які використовуються у розмовах і текстах, що стосуються місії та стратегії, мають бути ідентифіковані, а значення цих понять бути зрозумілими в компанії.
9. У компанії має бути чітко виражене прагнення рухатися вперед, яке, у свою чергу, має підтримуватися у явній формі її вищим керівництвом, тобто концепція «Інтелектуалізація управління» має бути глибоко впроваджена в систему стратегічного менеджменту. Більше того, щоб бути життєздатною, це поняття має бути узгоджене з існуючими управлінськими процесами[8].

Для застосування концепції «Інтелектуалізація управління» у систему управління підприємством велику роль відіграє організаційний контекст. Цей контекст включає:

* організаційну культуру; організаційну структуру;
* організаційну інфраструктуру та комунікації.

Створення організаційного контексту інтелектуальної організації вимагає, щоб організація навчалася не лише вести свій бізнес щодо бізнесу, а й здійснювати процес такого навчання. Вона має постійно прагнути покращувати свою культуру, структуру та інфраструктуру, якщо хоче покращити процеси індивідуального та організаційного навчання, ефективного використання свого інтелектуального потенціалу[9].

## **1.5 Архітектура інтелектуальних систем**

## **1.5.1 Властивості знань.** Дані – це інформація, отримана в результаті спостережень або вимірювань окремих властивостей (атрибутів), що характеризують об'єкти, процеси та явища предметної галузі[10].

Знання (з погляду подання знань в інтелектуальних системах) – це відносини та закономірності предметної галузі (принципи, моделі, закони), отримані в результаті практичної діяльності та професійного досвіду, що дозволяє фахівцям ставити та вирішувати завдання в даній галузі.

Знання від даних відрізняються низкою властивостей:

* внутрішня інтерпретація;
* структурованість;
* зв'язність;
* семантична метрика;
* активність.

Внутрішня інтерпретація. Дані, що зберігаються в пам'яті або зовнішніх носіях, позбавлені імен, таким чином, відсутня можливість їх однозначної ідентифікації системою. Дані може ідентифікувати лише програма, яка витягує їх за певним алгоритмом. При переході до знань на згадку вводиться додаткова інформація (атрибути: прізвище, рік народження, спеціальність, стаж). Атрибути можуть відігравати роль імен. За ними можна здійснювати пошук необхідної інформації[11].

Структурованість. Інформаційні одиниці повинні мати гнучку структуру. Інакше висловлюючись, має бути можливість довільного встановлення між окремими інформаційними одиницями відносин типу «частина – ціле», «рід -вид» або «елемент - клас».

Зв'язність. Між інформаційними одиницями має бути передбачена можливість встановлення зв'язків різного типу. Семантика відносин може мати декларативний чи процедурний характер. Наприклад, дві чи більше інформаційні одиниці може бути пов'язані ставленням «одночасно», дві інформаційні одиниці – ставленням «причина – слідство» чи «бути поруч».

Семантична метрика. На безлічі інформаційних одиниць у деяких випадках корисно ставити ставлення, що характеризує їхню ситуаційну близькість, тобто силу асоціативного зв'язку. Його можна назвати ставленням релевантності для інформаційних одиниць. Воно дає можливість виділяти в інформаційній базі деякі типові ситуації (наприклад, купівля, регулювання руху на перехресті). Ставлення релевантності під час роботи з інформаційними одиницями дозволяє знаходити знання, близькі вже знайденим[12].

Активність. Усі обчислювальні процеси ініціюються командами, а дані використовуються цими командами лише у разі потреби. Інакше висловлюючись, дані пасивні, а команди активні.

Знання дозволяють адаптуватися та діяти у реальній дійсності. Існує безліч різних знань, починаючи від рецепту приготування омлету до квантової фізики[13].

## **1.5.2 Класифікація знань.** Знання можна класифікувати за кількома критеріями (рис. 1.7).

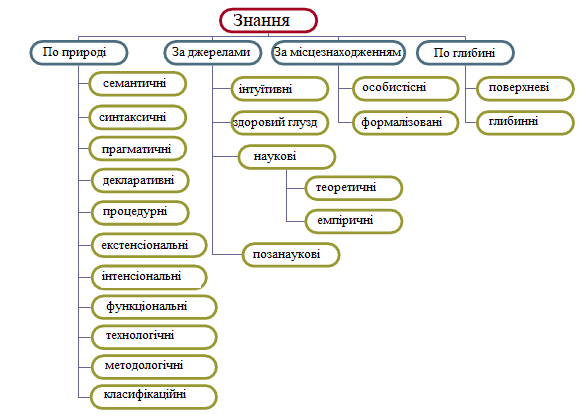


Рисунок 1.7 – Класифікація знань

Знання синтаксичного типу характеризують синтаксичну структуру потоку інформації, яка залежить від сенсу й змісту використовуваних у собі понять, тобто інтелектуальну систему не утворює[14].

Семантичне знання сприймається як структура, що утворює поточний контекст. Воно містить інформацію, безпосередньо пов'язану з поточними значеннями і змістом понять, що описуються, і визначає стан зв'язків даних в інформаційній базі.

Прагматичне знання визначає найбільш ймовірні зв'язки, що описують дані з точки зору розв'язуваної задачі (узагальнений або «об'єктивний» контекст), наприклад, з урахуванням специфічних критеріїв і угод, що діють у даному завданні.

Декларативні знання містять у собі уявлення про структуру понять. Ці знання наближені до даних, фактів. Наприклад, вищий навчальний заклад є сукупністю факультетів, а кожен факультет у свою чергу є сукупністю кафедр.

Процедурні знання мають активну природу. Вони визначають уявлення про засоби та шляхи отримання нових знань, перевірку знань. Це алгоритми різного роду. З розвитком інформатики дедалі більша частина знань зосереджувалася у структурах даних (таблиці, списки, абстрактні типи даних), тобто збільшувалася роль декларативних.

Істотними розуміння природи знань є способи визначення понять. Один із широко застосовуваних способів заснований на ідеї інтенсіоналу та екстенсіоналу.

Інтенсіональні поняття – це визначення через співвіднесення з поняттям вищого рівня абстракції із зазначенням специфічних властивостей.

Екстенсіонал поняття – визначення поняття через перерахування його конкретних прикладів, тобто понять нижчого рівня абстракції. Інтенсіонали формують знання про об'єкти, тоді як екстенсіонали поєднують дані.

Звідси інтенсіональні знання – це знання предметної області, які відбивають факти, закономірності, властивості і характеристики, справедливі будь-яких ситуацій, які можуть виникнути у цій предметної області[15].

Екстенсіональні знання – це знання предметної області, що відбивають факти, закономірності, властивості і характеристики, типові для конкретних ситуацій чи класів однотипних ситуацій, які можуть виникнути у цій галузі.

Функціональні знання - це знання про виконувані функції окремих предметів і про застосування їх у реальній дійсності.

Технологічні знання – спеціалізовані знання, які забезпечують підтримку технологічних параметрів виробництва; виробничий досвід та навички, що використовуються при вирішенні повсякденних виробничих питань. Це може бути знання послідовності операцій або знання технологічного ланцюжка, що дозволяють досягати поставленої мети відповідно до прийнятої технології.

Методологічні знання – знання методи перетворення дійсності, наукові знання про побудову ефективної діяльності. До методологічних знань відносять знання цілей, форм та напрямів розвитку теорії, методів та способів ефективного перетворення практики.

Класифікаційні знання застосовуються головним чином науці, є узагальненими, системними знаннями.

Інтуїція – це вид знання, специфіка якого зумовлена методом його придбання. Це знання, яке не потребує доказу і сприймається як достовірне. За способом отримання інтуїція – це пряме розсуд об'єктивного зв'язку речей, не спирається на підтвердження (інтуїція є розсуд внутрішнім зором; від латів. intueri – споглядати).

Під здоровим глуздом розуміють знання, що дозволяють приймати правильні рішення і робити правильні припущення, ґрунтуючись на логічному мисленні та накопиченому досвіді. У цьому значенні термін часто акцентує увагу на здатності людського розуму протистояти забобонам, помилкам, містифікаціям.

Наукові знання у будь-якому разі мають бути заснованими на емпіричній чи теоретичній доказовій основі.

Теоретичні знання – абстракції, аналогії, схеми, що відображають структуру та природу процесів, що протікають у предметній галузі. Ці знання пояснюють явища і можна використовувати для прогнозування поведінки об'єктів. Теоретичний рівень наукового знання передбачає встановлення законів, що дають можливість ідеалізованого сприйняття, опису та пояснення емпіричних ситуацій, тобто пізнання сутності явищ. Теоретичні закони мають суворіший, формальніший характер проти емпіричними. Терміни опису теоретичного знання відносяться до ідеалізованих, абстрактних об'єктів. Подібні об'єкти неможливо піддати безпосередньої експериментальної перевірки[16].

Емпіричні знання набувають у результаті застосування емпіричних методів пізнання: спостереження, виміру, експерименту. Це знання про видимі взаємозв'язки між окремими подіями та фактами у предметній галузі. Емпіричні знання, як правило, констатують якісні та кількісні характеристики об'єктів та явищ. Емпіричні закони часто мають імовірнісний характер і не є суворими.

Позанаукові знання можуть бути різними. Паранормальні знання – знання, несумісні з наявним гносеологічним стандартом. Широкий клас паранаукового (пара від грецьк. близько, при) знання включає вчення або роздуми про феномени, пояснення яких не є переконливим з точки зору критеріїв науковості.

Особистісні (неявні, приховані) знання – це знання людей, отримані з практики та досвіду.

Формалізовані (явні) знання – знання, які у документах, на компакт-дисках, в персональних комп'ютерах, в Інтернеті, в базах знань, в експертних системах. Формалізовані знання об'єктивуються знаковими засобами мови, охоплюють знання, про які ми знаємо, їх можна записати, повідомити іншим[17].

## **1.5.3 Бази знань.** База знань (БЗ) – основа будь-якої інтелектуальної системи, де знання описані деякою мовою уявлення знань, наближеною до природного. Сьогодні знання набули суто декларативної форми, тобто знаннями вважаються пропозиції, записані мовами подання знань, наближених до природної мови та зрозумілих нефахівцям[18].

Сукупність коштів, які забезпечують роботу із знаннями, утворює систему управління базою знань (СУБЗ). Нині немає баз знань, у яких повною мірою було б реалізовано внутрішню інтерпретованість, структуризацію, зв'язність, введено семантичну міру і забезпечено активність знань.

## **1.5.4 Архітектура інтелектуальних систем.** Архітектура інтелектуальних систем включає три комплекси обчислювальних засобів (рис. 1.8).

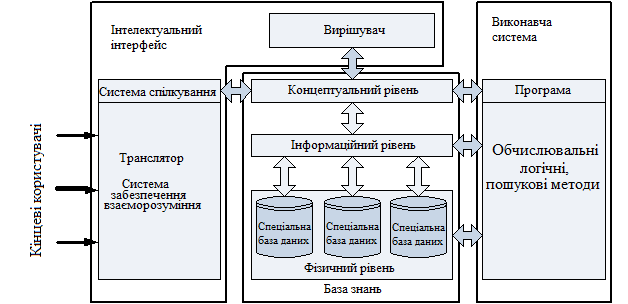


Рисунок 1.8 – Архітектура інтелектуальних систем

Перший комплекс є сукупність коштів, що виконують програми (виконавчу систему), спроектованих з позицій ефективного вирішення завдань, має у ряді випадків проблемну орієнтацію.

Другий комплекс є сукупністю засобів інтелектуального інтерфейсу, що мають гнучку структуру, яка забезпечує можливість адаптації в широкому спектрі інтересів кінцевих користувачів[19].

Третім комплексом засобів, за допомогою яких організується взаємодія перших двох, є база знань, що забезпечує використання обчислювальними засобами перших двох комплексів цілісної та незалежної від програм системи знань про проблемне середовище. Виконавча система поєднує всю сукупність коштів, які забезпечують виконання сформованої програми.

Інтелектуальний інтерфейс – система програмних і апаратних засобів, які забезпечують кінцевого користувача використання комп'ютера на вирішення завдань, які у середовищі його професійну діяльність або без посередників, або з незначною їх допомогою.

БЗ займає центральне положення стосовно інших компонентів обчислювальної системи. У цілому нині, через БЗ здійснюється інтеграція коштів обчислювальної системи, що у вирішенні завдань[20].

## **1.6 Постановка задачі**

Мета створення системи: підвищення ефективності процесу прогнозування для підприємства.

Призначення системи: моделювання попиту людей на продукцію, що виготовляє підприємство за вибраний період часу за допомогою нейронної мережі.

Отже, розробка програмного забезпечення припускає:

* розробку схеми БД;
* розробку та реалізацію додатка, що включає три основні модулі:

1. Модуль вводу та редагування інформації в БД.
2. Модуль авторизації користувачів системи.
3. Модуль прогнозування попиту на продукцію, що виготовляє підприємство.

Реалізація додатка виконується з використанням технологій .NET та мови програмування C#. Результат –Windows додаток.

Розроблена система повинна містити:

**Сутності:**

* користувачі системи (інформація про користувачів та їхні облікові записи): прізвище, ім’я, нікнейм облікового запису, пароль, загальна інформація, ідентифікатор ролі;
* дочірні відділи підприємства (інформація про дочірні відділи підприємства): назва відділу та опис;
* категорія продукції (інформація про категорію): назва та опис;
* співробітники (інформація про співробітників): ім’я, прізвище, номер телефону, адреса, електронна адреса та посада;
* продукція (інформація про продукцію): назва продукції, ідентифікатор категорії, одиниця виміру, ціна та опис;
* продаж продукції (інформація про продану продукцію): ідентифікатор продукції, дата продажу, кількість проданих одиниць, сума продажу;
* логи (інформація про активність користувачів системи) ідентифікатор користувача, подія в системі, дата/час події.

**Введення:**

* вводиться і редагується: користувачі системи, відділи, категорії продукції, продукція та співробітники;
* фіксується замовлення клієнтів на продукцію у підприємства;
* фіксується запис подій в системі.

**Пошук:** замовленої продукції у підприємствах.

**Прогнозування:** на основі інформації про продану продукцію здійснити прогнозування проданої продукції підприємства.

## **1.7 Висновок**

Створення інтелектуальної системи завжди включає стадії аналізу даних і бізнес-процесів замовника, проектування структур сховища з урахуванням його потреб і технологічних процесів.

Декілька десятків різних фірм випускають продукти, здатні вирішувати ті чи інші завдання, що виникають у процесі проектування та експлуатації інтелектуальних систем. Сюди входять СУБД, засоби вивантаження/ трансформації/ завантаження даних, інструменти для OLAP-аналізу та багато іншого. Самостійний аналіз ринку, вивчення хоча б кількох таких засобів – непросте та тривале завдання.

Враховуючи розмір фінансових та інших ресурсів, складність і багатоетапність проектів побудови інтелектуальної системи очевидна висока вартість помилок проектування. Помилки вибору програмного забезпечення можуть спричинити фінансові витрати, не кажучи вже про збільшення часу виконання проекту. Помилки проектування структури даних можуть вести як до неприйнятних бізнес рішень, так і коштуватиме часу витраченого на перезавантаження даних, який часом досягає кількох діб.

Тому глибоко розуміючи архітектуру сховищ даних, необхідно уникати будь-яких помилок, що спричиняє значне скорочення часу виконання проекту та можливість отримати максимальну віддачу від впровадження системи.

В даному розділі було визначено основні бізнес-процеси предметного середовища, що підлягають автоматизації. Визначено, що інтелектуалізація управління – це невід'ємний елемент стратегічного розвитку підприємства та його успішного функціонування. Поставлені мета та завдання, що мають бути вирішені для її досягнення. Визначено всю початкову інформацію, яка потрапляє до програми та всю звітну інформацію, яку програма видає в результаті роботи. На основі проведеного аналізу здійснено постановку задачі до створення інтелектуальної системи управління діяльності підприємства.

# **РОЗДІЛ 2**

# **ПРОЕКТУВАННЯ, РОЗРОБКА, ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ**

## **2.1 Аналіз вимог. Use-case діаграми. Основні прецеденти**

Перед розробкою інформаційної інтелектуальної системи управління діяльності підприємства необхідно визначити акторів системи та їхні ролі в ній.

Було прийняте рішення створити дві ролі: адміністратор системи та менеджер.

Для більш кращого розуміння функціональності створеної системи буде доцільним проектування діаграми прецедентів (use-case).

Діаграма прецедентів або діаграма варіантів використання в UML – діаграма, що відображає відносини між акторами та прецедентами і є складовою моделі прецедентів, що дозволяє описати систему на концептуальному рівні.

Прецедент – можливість моделювання системи завдяки якій користувач може отримати конкретний, вимірний і потрібний йому результат. Прецедент відповідає окремому сервісу системи, визначає один із варіантів її використання та визначає типовий спосіб взаємодії користувача з системою. Варіанти використання зазвичай застосовуються для специфікації зовнішніх вимог до системи

Таблиця 2.1 – Функціональні вимоги до додатку

|  |  |
| --- | --- |
| **Вимоги** | **Опис** |
| REQ-1 | Система повинна дозволяти здійснювати реєстрацію користувача в програмі |
| REQ-2 | Система повинна вести журнал подій, що відбулися в програмі |
| REQ-3 | Система повинна дозволяти користувачеві додати та редагувати інформацію про облікові записи користувачів |

Продовження табл. 2.1

|  |  |
| --- | --- |
| REQ-4 | Система повинна дозволяти користувачеві додати та редагувати інформацію про продукцію |
| REQ-5 | Система повинна дозволяти користувачеві додати та редагувати інформацію про співробітників |
| REQ-6 | Система повинна дозволяти користувачеві додати та редагувати інформацію про дочірні підприємства |
| REQ-7 | Система повинна дозволяти користувачеві здійснювати симуляцію продажу продукції |
| REQ-8 | Система повинна дозволяти користувачеві проводити навчання нейронної мережі на основі даних продажу продукції |
| REQ-9 | Система повинна дозволяти користувачеві здійснювати прогнозування продажу продукції |

Таблиця 2.2 – Нефункціональні вимоги до додатку

|  |  |
| --- | --- |
| **Вимоги** | **Опис** |
| REQ-10 | Додаток повинен мати простий дизайн та зручну навігації |
| REQ-11 | Поля повинні бути не порожніми, унікальними відносно вже існуючих записів |

Таблиця 2.3 – Актори та цілі додатку

|  |  |
| --- | --- |
| **Актори** | **Цілі** |
| Адміністратор | Мета адміністратора полягає у роботі із обліковими записами |
| Менеджер | Мета менеджера полягає у роботі із системою |
| База даних | Мета бази даних полягає у зберігання інформації |

Таблиця 2.4 – Опис варіантів використання додатку

|  |  |  |
| --- | --- | --- |
| **Варіант використання** | **Ім’я** | **Опис** |
| UC1 | Ідентифікація в системі | Дозволяє користувачу пройти ідентифікацію в системі |
| UC2 | Вивід каталогу користувачів | Дозволяє користувачеві з правами системного адміністратора вивести каталог всіх користувачів системи |
| UC3 | Додати користувача | Дозволяє користувачеві додати нового користувача системи |
| UC4 | Редагувати користувача | Дозволяє користувачеві редагувати інформацію вибраного із списку користувача |
| UC5 | Видалити користувача | Дозволяє користувачеві видалити вибраного із списку користувача |
| UC6 | Вивід каталогу продукції | Дозволяє користувачеві вивести каталог всієї продукції |
| UC7 | Додати продукцію | Дозволяє користувачеві додати нову продукцію |
| UC8 | Редагувати продукцію | Дозволяє користувачеві редагувати інформацію вибраної із списку продукції |
| UC9 | Видалити продукцію | Дозволяє користувачеві видалити вибрану із списку продукцію |
| UC10 | Вивід каталогу співробітників | Дозволяє користувачу вивести каталог всіх зареєстрованих у системі співробітників |

Продовження табл. 2.4

|  |  |  |
| --- | --- | --- |
| UC11 | Додати співробітника | Дозволяє користувачу додати інформацію про співробітника |
| UC12 | Редагувати співробітника | Дозволяє користувачеві редагувати інформацію вибраного із списку співробітника |
| UC13 | Видалити співробітника | Дозволяє користувачеві видалити вибраного співробітника |
| UC14 | Вивід каталогу дочірніх підприємств | Дозволяє користувачу вивести каталог всіх дочірніх підприємств |
| UC15 | Додати дочірнє підприємство | Дозволяє користувачу додати інформацію про дочірнє підприємство |
| UC16 | Редагувати дочірнє підприємство | Дозволяє користувачеві редагувати інформацію дочірнього підприємства |
| UC17 | Видалити дочірнє підприємство | Дозволяє користувачеві видалити дочірнє підприємство |
| UC18 | Симуляція продаж продукції | Дозволяє користувачеві здійснити симуляцію продажі продукції у вибраному дочірньому підприємстві |
| UC19 | Прогнозування попиту на продукцію | Дозволяє користувачеві здійснювати прогнозування попиту на продукцію |
| UC20 | Вивід системних подій | Дозволяє вивести всі події, які відбулися в системі |

З поставлених вимог тепер можна виставити повний опис вимог із сценаріями, що будуть вхідними даними для візуального моделювання мовою UML[21].

**UC1 Ідентифікація в системі**

Актор: адміністратор/менеджер.

Ціль актора: пройти ідентифікацію в системі.

Задіяний актор: база даних.

Передумова: користувач ввівши нікнейм на пароль натискає кнопку «Підтвердити».

Післяумова: система відображає головне вікно програми з меню.

**UC2 Вивід каталогу користувачів**

Актор: адміністратор.

Ціль актора: вивести інформацію про всіх користувачів системи.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Користувачі».

Післяумова: система відображає екран для виведення списку всіх зареєстрованих користувачів системи.

**UC3 Додати користувача**

Актор: адміністратор.

Ціль актора: додати нового користувача.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про користувача натискає на кнопку «Додати».

Післяумова: система додає нового користувача та відображає екран із списком всіх користувачів.

**UC4 Редагувати користувача**

Актор: адміністратор.

Ціль актора: редагувати інформацію про вибраного користувача.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку користувача.

Післяумова: система відображає екран для редагування інформації про вибраного користувача.

**UC5 Видалити користувача**

Актор: адміністратор.

Ціль актора: видалити користувача із бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку користувача.

Післяумова: система відображає екран з можливістю видалення даних про користувача.

**UC6 Вивід каталогу продукції**

Актор: адміністратор/менеджер.

Ціль актора: вивести інформацію про продукцію.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Продукція».

Післяумова: система відображає екран для виведення списку всієї продукції.

**UC7 Додати продукцію**

Актор: адміністратор/менеджер.

Ціль актора: додати нову продукцію.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про продукцію натискає на кнопку «Додати».

Післяумова: система додає нову продукцію і відображає екран із списком всієї продукції.

**UC8 Редагувати продукцію**

Актор: адміністратор/менеджер.

Ціль актора: редагувати вибрану із списку продукцію.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку продукцію.

Післяумова: система відображає екран для редагування інформації про вибрану із списку продукцію.

**UC9 Видалити продукцію**

Актор: адміністратор/менеджер.

Ціль актора: видалити вибрану продукцію із бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідну із списку продукцію.

Післяумова: система відображає екран з можливістю видалення даних про продукцію.

**UC10 Вивід каталогу співробітників**

Актор: адміністратор/менеджер.

Ціль актора: вивести інформацію про всіх співробітників.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Співробітники».

Післяумова: система відображає екран для виведення списку всіх співробітників.

**UC11 Додати співробітника**

Актор: адміністратор/менеджер.

Ціль актора: додати інформацію про нового співробітника.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про співробітника натискає на кнопку «Додати».

Післяумова: система додає нового співробітника і відображає екран із списком всіх співробітників.

**UC12 Редагувати співробітника**

Актор: адміністратор/менеджер.

Ціль актора: редагувати вибраного із списку співробітника.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку співробітника.

Післяумова: система відображає екран для редагування інформації про вибраного співробітника’.

**UC13 Видалити співробітника**

Актор: адміністратор/менеджер.

Ціль актора: видалити співробітника із бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідного із списку співробітника.

Післяумова: система відображає екран з можливістю видалення даних про співробітника.

**UC14 Вивід каталогу дочірніх підприємство**

Актор: адміністратор/менеджер.

Ціль актора: вивести інформацію про всі дочірні підприємства.

Задіяний актор: база даних.

Передумова: користувач натиснув на пункт меню «Дочірні підприємства».

Післяумова: система відображає екран для виведення списку всіх дочірніх підприємств.

**UC15 Додати дочірнє підприємство**

Актор: адміністратор/менеджер.

Ціль актора: додати інформацію про дочірнє підприємство.

Задіяний актор: база даних.

Передумова: користувач ввівши всю необхідну інформацію про дочірнє підприємство натискає на кнопку «Додати».

Післяумова: система додає нове дочірнє підприємство і відображає екран із списком всіх дочірніх підприємство.

**UC16 Редагувати дочірнє підприємство**

Актор: адміністратор/менеджер.

Ціль актора: редагувати вибране із списку дочірнє підприємство.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідне із списку дочірнє підприємство.

Післяумова: система відображає екран для редагування інформації про вибране дочірнє підприємство.

**UC17 Видалити дочірнє підприємство**

Актор: адміністратор/менеджер.

Ціль актора: видалити дочірнє підприємство із бази даних.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідне із списку дочірнє підприємство.

Післяумова: система відображає екран з можливістю видалення даних про дочірнє підприємство.

**UC18 Симуляція продаж продукції**

Актор: адміністратор/менеджер.

Ціль актора: дозволяє користувачеві провести симуляцію продажі продукції у дочірньому підприємстві за вибраний період часу.

Задіяний актор: база даних.

Передумова: користувач вибирає необхідне із списку дочірнє підприємство, продукт, вказує період часу та натискає на кнопку «Симулювати».

Післяумова: система здійснює симуляцію продаж у вибраному дочірньому підприємстві.

**UC19 Прогнозування попиту на продукцію**

Актор: адміністратор/менеджер.

Ціль актора: здійснити прогнозування попиту на продукцію у вибраному дочірньому підприємстві.

Задіяний актор: база даних.

Передумова: користувач користуючись меню «Прогнозування» має можливість здійснити прогнозування попиту на продукцію у вибраному дочірньому підприємстві.

Післяумова: система відображає екран з можливістю здійснення прогнозування попиту за вибраний користувачем проміжок часу.

**UC20 Вивід системних подій**

Актор: адміністратор.

Ціль актора: вивести список всіх подій в системі.

Задіяний актор: база даних.

Передумова: користувач вибирає пункт меню «Системний журнал» [22].

Післяумова: система відображає екран із списком всіх системних подій.



Рисунок 2.1 – Діаграма use-case для користувача системи із правами адміністратора

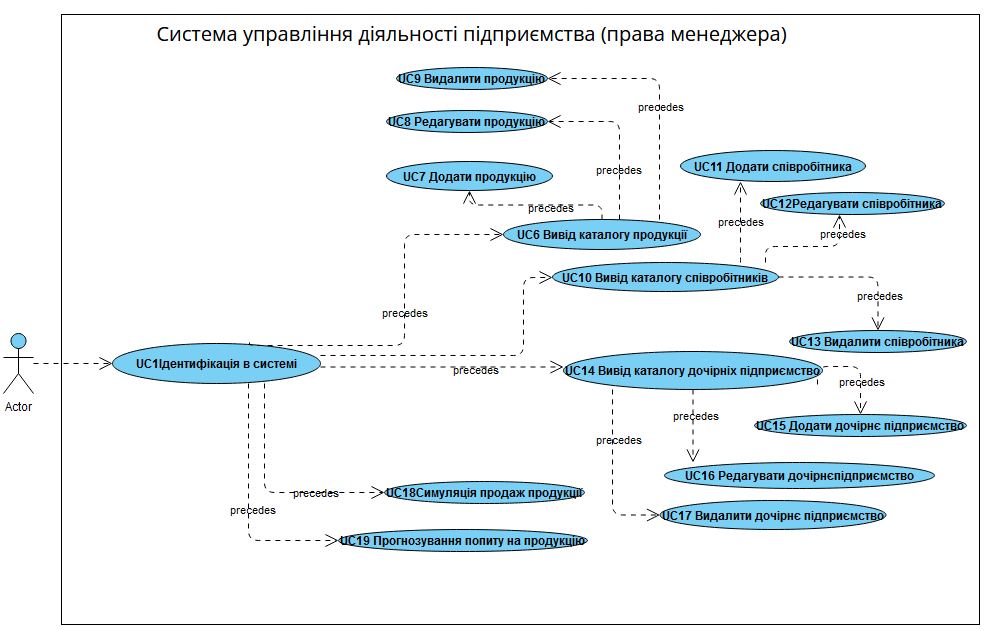


Рисунок 2.2 – Діаграма use-case для користувача системи із правами менеджера

Менеджер опрацьовує дані співробітників, продукції, дочірніх підприємств, проводить симуляцію продаж у дочірніх підприємствах, здійснює прогнозування попиту продаж у них.

Адміністратор окрім функцій менеджера має можливість реєстрації користувачів у системі та надає їм права доступу згідно з роллю.

## **2.2 Архітектура проекту**

Розроблений продукт повинен відповідати характеристикам якості, таким як: стійкість, корисність, доступність, масштабованість, відкритість, гнучкість, можливість тестування. Це вимагає від процесу розробки додаткові обмеження/правила, а саме:

* дотримання шаблонів і стилів;
* документування розробки на різних рівнях;
* тестування компонентів, окремих модулів, підсистем;
* управління проектами, процесами.

З урахуванням вимог до забезпечення стійкості та гнучкості системи при її розробці було обрано шаблон Layers, який розбиває систему на дві частини: клієнт та сервер.

Проектування системи буде покладатись на предметну область та принципи SOLID.

При розробці покупця був обраний користувацький інтерфейс Windows Forms.

Сервер, в свою чергу, буде складатись з таких модулів:

1. BLL (англ. Business Logic Layer) – логіка та всі необхідні обчислення додатку на мові бізнесу;
2. DAL (англ. Data Acces Layer) – рівень доступу до даних.
3. DB (англ. Data Base) – база даних для зберігання даних.

З урахування всіх вище перерахованих шаблонів структура проекту буде виглядати наступним чином:

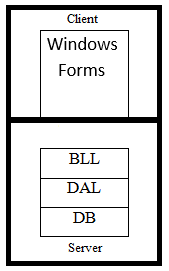


Рисунок 2.3 – Структура проекту

Оскільки бізнес постійно і відносно швидко змінюється, то кожен із компонентів повинен швидко адаптуватись під ці зміни. Для цього використаємо шаблон Dependency Injection (DI) [23].

## **2.3 Особливості розробки бази даних. ERD діаграма з описанням сутностей**

Схема «сутність-зв’язок» (також ERD або ER-діаграма) – це різновид блок-схеми, де показано, як різні «сутності» (люди, об’єкти, концепції і так далі) пов’язані між собою всередині системи. ER-діаграми найчастіше застосовуються для проектування та налагодження реляційних баз даних у сфері освіти, дослідження та розробки програмного забезпечення та інформаційних систем для бізнесу.

ER-діаграми (або ER-моделі) покладаються на стандартний набір символів, включаючи прямокутники, ромби, овали та сполучні лінії для відображення сутностей, їх атрибутів та зв’язків. Ці діаграми влаштовані за тим самим принципом, як і граматичні структури: сутності виконують роль іменників, а зв’язку — дієслів[24].

ER-діаграми – схожі зі схемами структури даних (DSD), де замість зв’язків між самими сутностями відображається відношення між елементами всередині них. ER-діаграми часто використовуються у поєднанні з діаграмами DFD, що схематично показують рух потоків інформації в рамках процесу або системи[25].

Згідно завдання було побуловано ERD діаграму.

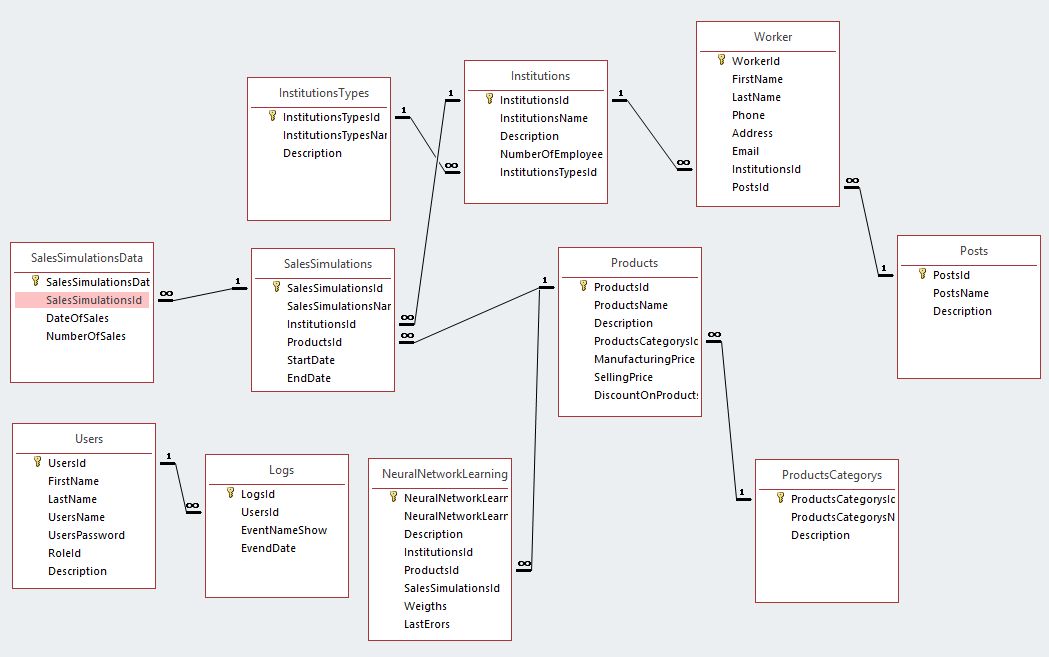


Рисунок 2.4 – ERD діаграма

Як можна побачити із рис. 2.4 база даних складається із 11 сутностей. Кожна сутність має свою таблицю, а саме:

1. Таблиця «Institutions» – зберігає інформацію про дочірні підприємства.
2. Таблиця «InstitutionsTypes» - зберігає інформацію про типи дочірніх підприємст.
3. Таблиця «NeuralNetworkLearning» - зберігає інформацію про навчання нейронної мережі.
4. Таблиця «Posts» - зберігає інформацію про посади працівників.
5. Таблиця «Products» - зберігає інформацію про продукцію.
6. Таблиця «ProductsCategorys» - зберігає інформацію про категорії продукції.
7. Таблиця «SalesSimulations» - зберігає інформацію про проведену симуляцію продажу продукції.
8. Таблиця «SalesSimulationsData» - зберігає інформацію про всі згенеровані дані продажу вибраного продукту та дочірнього підприємства.
9. Таблиця «Worker» - зберігає інформацію про працівників.
10. Таблиця «Logs» - зберігає інформацію про активність користувачів системи і їхні дії.
11. Таблиця «Users» - містить інформацію про всіх користувачів системи та їхні облікові дані.

## **2.4 Особливості розробки рівня BLL**

Даний шар містить клас «NeuronsBLL», в якому реалізована логіка формування звітів та пошуку. Діаграму даного класу показано на рис. 2.5.

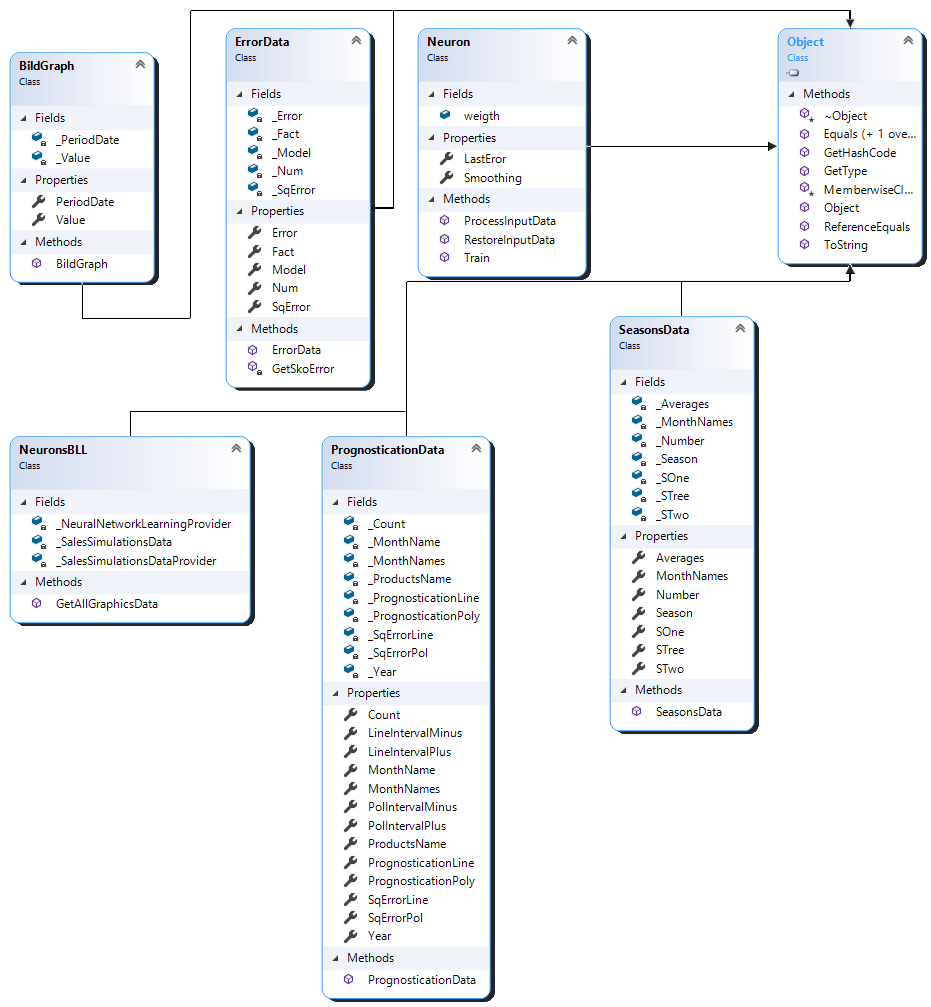


Рисунок 2.5 – Діаграма класу BLL «Система управління діяльності підприємства»

У класі «NeuronsBLL» реалізовано публічний метод «GetAllGraphicsData», що повердає дані для рисування графіку прогнозованих даних[26].

## **2.5 Особливість реалізації бізнес логіки – діаграма домена**

Будуємо діаграму класів домена. Кожен клас описує конкретну таблицю бази даних для зручного опрацювання даних (рис. 2.6).

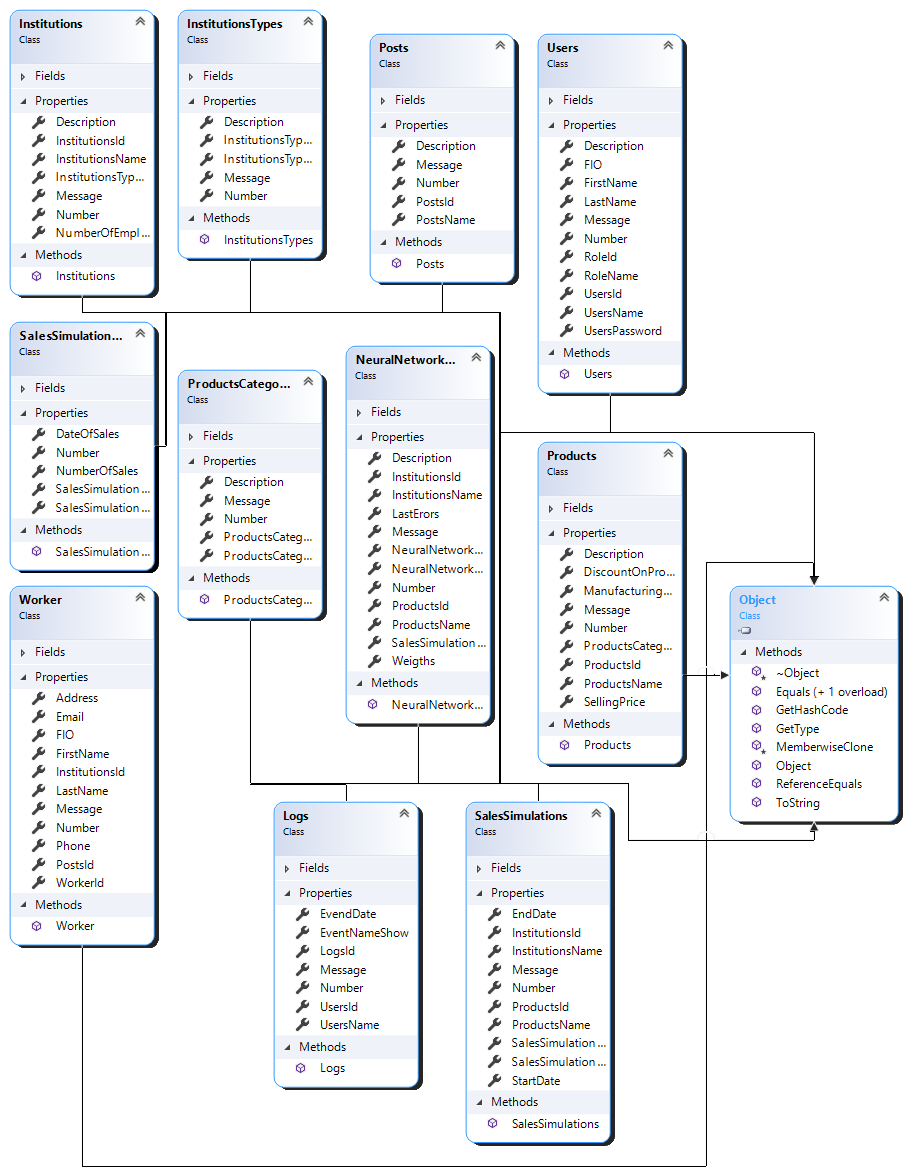
****

Рисунок 2.6 – Діаграма класів домена

Як можна побачити із рис. 2.6, у проекті реалізовано рівно стільки класів, скільки і сутностей у базі даних.

## **2.6 Особливості розробки рівня UI**

User interface (UI) елементи – це частини, які дизайнери використовують для створення програм або веб-сайтів. Вони додають інтерактивність в інтерфейс користувача, надаючи користувачеві точки зіткнення при навігації по них. Кнопки, смуги прокручування, пункти меню та чекбокси.

Інтерфейсу користувача (UI) використовує UI елементи для створення візуальної мови і забезпечення узгодженості продукту, що робить його зручним для користувача і простим у навігації без особливих зусиль з боку користувача.

UI елементи зазвичай поділяються на одну з наступних чотирьох груп:

* елементи керування введенням (Input Controls) – дозволяють користувачам вводити інформацію до системи;
* компоненти навігації (Navigation Components) – допомагають користувачам переміщатися продуктом або веб-сайтом. Загальні навігаційні компоненти включають панелі вкладок та головне меню програми;
* інформаційні компоненти (Informational Components) – діляться інформацією з користувачем;
* контейнери (Containers) – містять зв’язаний контент разом.

В даному проекті для розробки користувацького інтерфейсу було використано Windows Forms[26].

Windows Forms – це платформа користувача інтерфейсу для створення класичних додатків Windows. Вона забезпечує один з найефективніших способів створення класичних додатків за допомогою візуального конструктора в Visual Studio.

На рис. 3.6 показана діаграма програми «Система управління діяльності підприємства» з правами адміністратора системи.

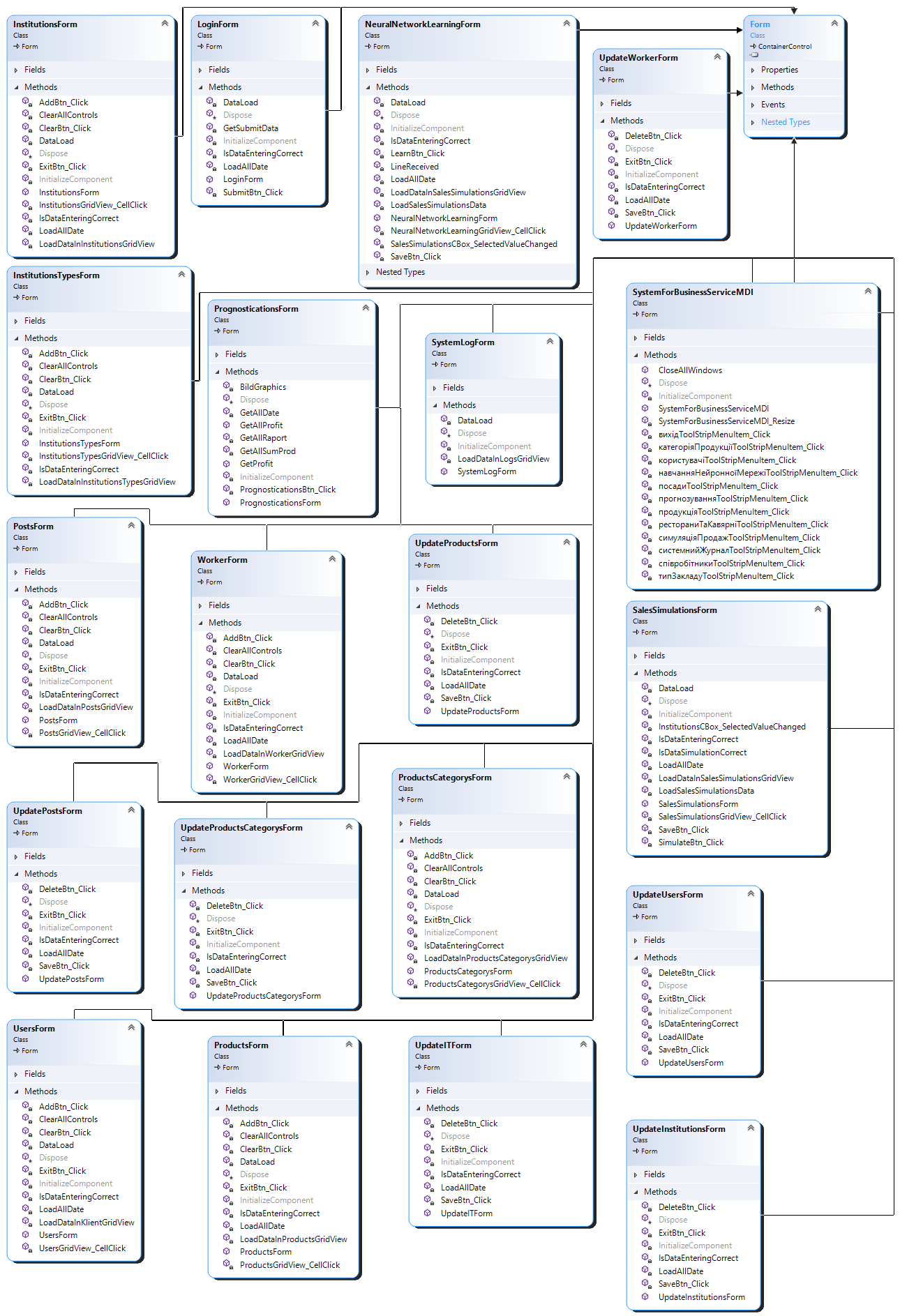


Рисунок 2.7 – Діаграма інтерфейсу програми «Система управління діяльності підприємства»

Програма складається з двадцяти класів рівня UI та є похідними від класу Form, тобто мають графічний інтерфейс[27].

Клас «SystemForBusinessServiceMDI» становить головне вікно програми. Найважливішим елементом управління в головному вікні є розташування головного меню, за допомогою якого можна відкривати інші форми та вийти з програми.

Клас «InstitutionsTypesForm» призначений для ведення інформації про типи підприємств.

Клас «UpdateInstitutionsTypesForm» призначений для редагування інформації вибраного типу підприємства.

Клас «InstitutionsForm» призначений для ведення інформації про підприємство.

Клас «UpdateInstitutionsForm» призначений для редагування інформації вибраного підприємства.

Клас «PostsForm» призначений для ведення інформації про посади працівників.

Клас «UpdatePostsForm» призначений для редагування інформації про посади.

Клас «ProductsCategorysForm» призначений для ведення інформації про категорії продукції.

Клас «UpdateProductsCategorysForm» призначений для редагування інформації вибраної категорії продукції.

Клас «WorkerForm» призначений для ведення інформації про працівників.

Клас «UpdateWorkerForm» призначений для редагування інформації вибраного працівника.

Клас «ProductsForm» призначений для ведення інформації про продукцію.

Клас «UpdateProductsForm» призначений для редагування інформації вибраної продукції.

Клас «NeuralNetworkLearningForm» призначений для проведення навчання нейронної мережі.

Клас «PrognosticationsForm» призначений для проведення прогнозування попиту на продукцію.

Клас «SalesSimulationsForm» призначений для проведення симуляції продажу продукції.

Клас «UpdateGoodsForm» призначений для редагування інформації вибраного товару.

Клас «LoginForm» призначений для авторизації користувачів системи.

Клас «SystemLogForm» призначений для перегляду системних подій.

Клас «UsersForm» призначений для опрацювання даних про користувачів системи.

Клас «UpdateUsersForm» призначений для редагування інформації про користувачів системи.

Створення об’єктів інших команд та виклик їх методів відбувається в результаті взаємодії користувача з елементами графічного інтерфейсу програми[24].

## **2.7 Особливості розробки DAL**

Шар доступу до даних (Data Access Layer – DAL) у програмному забезпеченні – це шар комп'ютерної програми, який надає спрощений доступ до даних, що зберігаються у постійному сховищі якогось типу, такому як реляційна база даних. Цей акронім в основному використовується в оточенні Microsoft.NET.

DAL може повертати посилання на об'єкт (у термінах об'єктно-орієнтованого програмування) з його атрибутами замість рядків полів із таблиці бази даних. Це дозволяє створювати клієнтські модулі з більш високим рівнем абстракції. Така модель може бути реалізована шляхом створення класу з методами доступу до даних, які безпосередньо посилаються на відповідний набір процедур бази даних. Інша реалізація може потенційно отримувати або записувати записи або з файлової системи. DAL приховує складність сховища даних, що лежить в основі даних[29].

Замість використання таких команд як «створити», «видалити» або «оновити» в певній таблиці в базі, клас і кілька процедур, що зберігаються, можуть бути створені в базі. Ці процедури можуть викликатися методом усередині класу, який поверне об'єкт, що містить запитані значення. Або команди створення, видалення та оновлення можуть бути виконані всередині простих функцій, що зберігаються у шарі доступу до даних.

Також методи бізнес-логіки із програми можуть бути співвіднесені до шару доступу до даних.

Для роботи з базою даних було реалізовано 11 класів. Назва кожного класу починається із відповідній їй назві таблиці в базі даних та закінчується приставкою «Provider» [30].

На рис. 2.8 приведено діаграму класів з методами для роботи з базою даних.

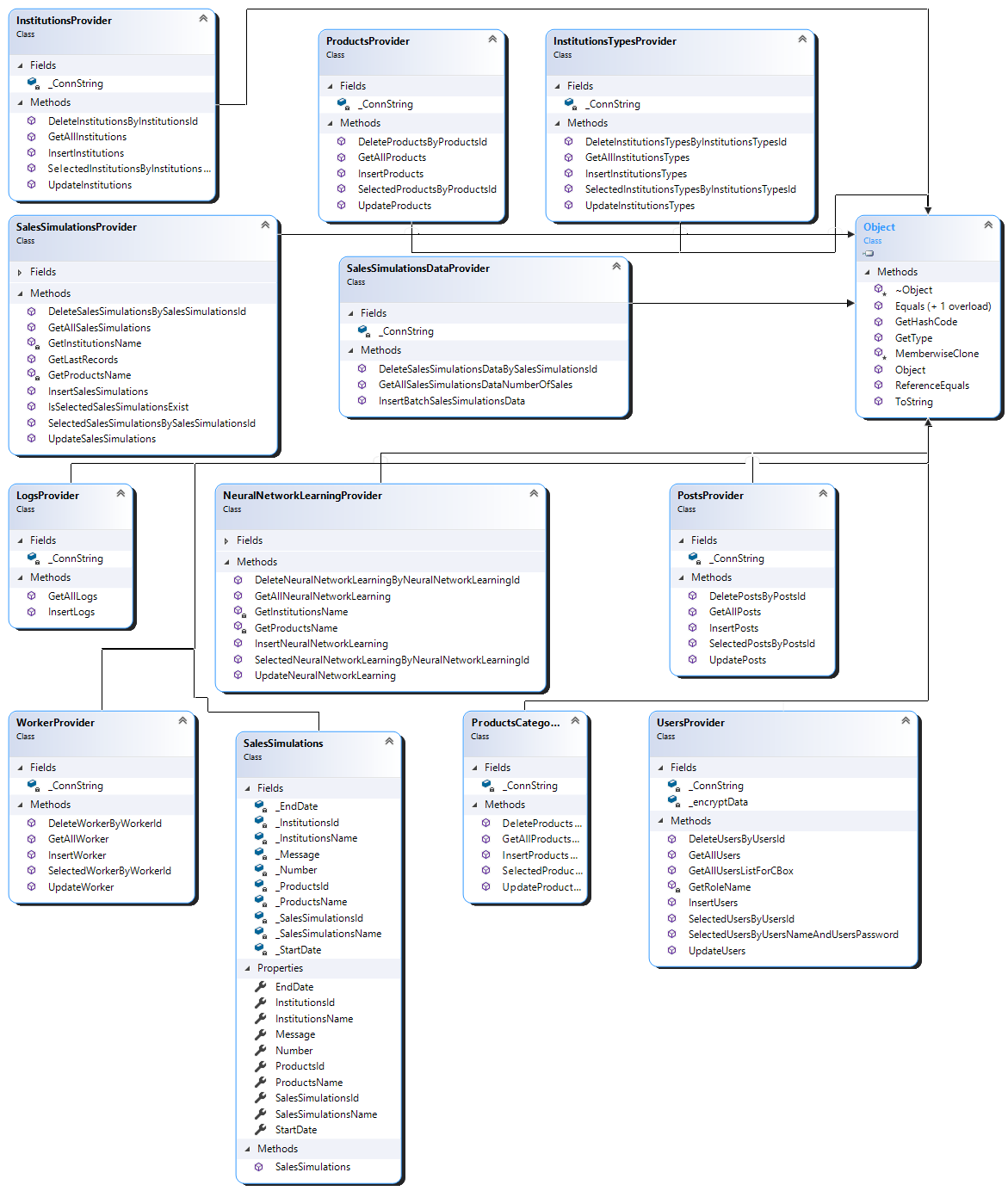


Рисунок 2.8 – Діаграма класів з методами для роботи з базою даних

## **2.8 Висновок**

Побудова сучасних інформаційних систем займає дуже багато часу. Вона починається з аналізу предметної області, детального планування системи, описання вимог, моделювання її поведінки за допомогою UML діаграм. Визначається шаблони, які будуть використовуватись при розробці.

Після планування починається стадія розробки. Розробка починається зі створення користувацького інтерфейсу і закінчується базою даних.

При розробці слід враховувати, що вимоги змінюються швидко і потрібно будувати систему так, щоб вона була гнучкою.

Розробка системи виконується по окремим компонентам. Кожний створений компонент потрібно детально тестувати, щоб мінімізувати помилки на наступних етапах розробки.

Якщо дотримуватися всіх вищеперерахованих вимог, то можна побудувати гнучку і стійку інформаційну систему.

# **РОДІЛ 3**

# **ОСОБЛИВОСТІ ВИКОРИСТАННЯ ПРОГРАМНОГО ДОДАТКУ**

## **3.1 Вибір технологій**

Для побудови користувацького інтерфейсу будемо використовувати середовище розробки Visual Studio 2019 та мову програмування C#.

С# широко використовується професіоналами для розробки великих програмних продуктів завдяки наступним аспектам мови:

1. Сучасна мова програмування загального призначення.
2. Підтримка об’єктно-орієнтованої парадигми.
3. Підтримка компонентно-орієнтованої парадигми.
4. Мова легка для вивчення.
5. Добре структурована.
6. Дозволяє розробляти ефективні програми.
7. Мова має підтримку різних комп’ютерних платформах.
8. Це частина .Net Framework.

C# спроектовано таким чином, що мова відповідає традиційним мовам високого рівня, C та C++ і є об’єктно-орієнтованою мовою програмування. Мова дуже схожа на Java, має численні сильні функції програмування, які роблять його привабливим для багатьох програмістів у всьому світі.

.NET Framework – це платформа для розробників із відкритим кодом, яку можна використовувати для створення широкого кола програм. Цей безкоштовний крос-платформний фреймворк підтримує декілька мов і має великі бібліотеки коду, які спрощують створення додатків для мобільних пристроїв, робочих столів та Інтернету.

Платформа .NET була розроблена для досягнення наступних цілей:

* сумісність;
* підтримка різних платформам;
* мовна незалежність;
* бібліотека базових класів;
* легка розробка;
* безпека.

Для розробки користувацького інтерфейсу платформа .Net має декілька технологій, одна з яких – WinForms. Не дивлячись на те, що ця технологія досить не нова, її важко назвати застарілою. Вона надає широкий спектр різних інструментів для побудови зручного та сучасного інтерфейсу. Крім того, IDE, які підтримують C# та .Net, надають зручний інтерфейс для графічної побудови користувацького застосунку, який розробляється.

Отже, C# та платформа .Net має низку характеристик, які задовольняють вимоги щодо розробки клієнтської частини системи складського обліку. Набір готових класів у стандартній бібліотеці, лаконічний зрозумілий синтаксис мови та зручний конструктор користувацького інтерфейсу зробить розробку зручною та достатньо швидкою. Об’єктно-орієнтована парадигма дозволить спроектувати систему таким чином, що розширення функціоналу буде без накладних розходів ресурсів розробки. Платформа .Net забезпечить безпеку, ефективність програмного забезпечення, а також підтримку декількох платформ.

Для розробки інформаційної бази використовувався Microsoft Access. MS Access – це СУБД, що входить до складу пакету офісних програм Microsoft Office. Дана система управління базами даних має широкий спектр функцій (зв’язні запити, сортування, зв’язки із зовнішніми таблицями та базами даних).

Переваги використання:

* простий інтерфейс користувача, що дозволяє розробляти додатки, використовуючи вбудовані бібліотеки;
* зберігає всі дані у одному файлі;
* пропонує велику кількість «Майстрів», які допомагають уникнути рутинних дій і полегшують роботу досвідченому в програмуванні користувачеві;
* поширеність, яка зумовлена тим, що Access є продуктом компанії Microsoft;
* постійно оновлюється виробником, підтримує безліч мов;
* орієнтованість на користувача з різною професійною підготовкою,;
* широкі можливості імпорту/експорту даних у різні формати;
* наявність розвинених інтегрованих засобів розробки додатків. Більшість програм, що розповсюджуються серед користувачів, містить той чи інший обсяг коду VBA (Visual Basic for Applications);
* наявність вбудованої мови макрокоманд.

Недоліки:

* обмежені можливості щодо забезпечення розрахованого на багато користувачів середовища;
* має нескладні способи захисту з використанням пароля БД (можливе застосування додаткових заходів щодо захисту від несанкціонованого доступу з використанням процедур VBA);
* в питаннях підтримки цілісності даних відповідає лише моделям БД невеликої та середньої складності;
* не розповсюджується безкоштовно.

Отже СУБД Access було обрано тому, що розроблена програма може легко переноситись на будь-які системи сімейства Windows і не потребує додаткового встановлення сервера бази даних. Реляційність бази даних дає можливість ефективно використовувати пам’ять та уникнути дублювання інформації. Завдяки відкритому коду не потрібно витрачати зайві ресурси на різні ліцензії, тобто розробка та використання системи на базі СУБД Access значно дешевша. Підтримка проекту відбувається вже багато років розробниками з усього світу, через що надійність та безпека продукту на високому рівні. Таким чином, СУБД Access повністю покриває вимоги щодо розробленого проекту.

Для роботи з Базою даних реалізовуємо класи (рис. 3.1).

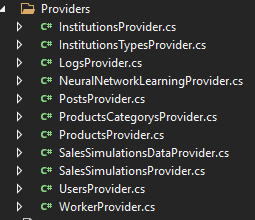


Рисунок 3.1 – Класи для роботи з БД

В кожному класі є методи, за допомогою яких можна: вибирати, вставляти, редагувати та видаляти дані. Також є методи, які працюють з вибіркою даних із декількох таблиць.

## **3.2 Результати функціонального тестування розробленого додатку**

1. Тестування – це процес аналізу та дослідження, який надає змогу виявити інформацію про якість продукту відносно умов, в яких він буде застосовуватись. Методика тестування також включає в себе процес пошуку дефектів, помилок, несправностей. Також це є випробуванням програмних складових з метою оцінити готовність програмного продукту до використання. Результат тестування оцінюється за наступними критеріями:

* відповідність вимогам, які надавалися розробниками та проектувальниками;
* відповідність вихідних даних;
* прийнятний час виконання функцій;
* практичність;
* відповідність вимогам замовника.

1. Кількість тестів навіть для простих програмних компонентів може бути ледь не нескінченним, тому тактика тестування має полягати в тому, що будуть проведені тільки необхідні тести з урахуванням доступного часу та ресурсів. Як результат, програмні засоби тестуються стандартним виконанням програми з метою виявлення багів, помилок або інших дефектів.
2. Існує багато видів тестування: одні зазвичай виконують самі розробники, а інші – спеціалізовані групи. У роботі буде використовуватись тестування системи.
3. Тестування системи – це виконання програмного забезпечення в його остаточної конфігурації, інтегрованого з іншими програмними та апаратними системами.
4. Одним із способів вивчення поставленого питання є дослідження методики «чорної скриньки», Основна роль тестування методів «чорної скриньки» – це інтерфейс програмного забезпечення. Ці тести демонструють:

* як будуть виконуватись функції програми;
* як будуть прийматися вихідні дані;
* як будуть формуватись результати;
* як буде збережена цілісність зовнішньої інформації.

1. Тестування програмних засобів буде доречно проводити використовуючи методику «чорної скриньки».
2. Вона базується на використанні шаблонів тестування або ж тест-кейсів. Це означає, що буде створено декілька ситуацій у яких перевіряється працездатність додатку, коректності основних функцій.
3. Таблиця 3.1 – Тест-кейси

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код тест-кейса | Опис тест-кейса | | | |
| Хід тестування | | Очікуваний результат | |
| Дата тестування | Результат | Примітка | |
| 001 | Перевірка реєстрації користувача програми | | | |
| 1. Запустити додаток; 2. Провести автентифікацію користувача із роллю «адміністратор». | | Вивід відповідного вікна із функціоналом для роботи користувача | |
| 19.11.2022 | Пройдено | – | |
| 002 | Перевірка опрацювання інформації про додавання нового дочірнього підприємства | | | |
| 1. Запустити додаток; 2. Провести автентифікацію; 3. Вибрати відповідний пункт меню; 4. Додати дочірнє підприємство; 5. Редагувати дочірнє підприємство; 6. Видалити дочірнє підприємство. | | Список дочірніх підприємств відображається відповідно до введених користувачем даних | |
| 19.11.2022 | Пройдено | – | |
| 003 | Проведення симуляції продажу продукції | | | | |
| 1. Запустити додаток; 2. Провести автентифікацію; 3. Вибрати відповідний пункт меню; 4. Провести симуляцію; 5. Зберегти дані. | | | Симуляцію проведена успішно, дані збережено у базі даних | |

Продовження табл. 3.1

|  |  |  |  |
| --- | --- | --- | --- |
|  | 19.11.2022 | Пройдено | – |
|  | Перевірка навчання нейронної мережі | | |
| 004 | 1. Запустити додаток; 2. Провести автентифікацію; 3. Вибрати відповідний пункт меню; 4. Провести навчання нейромережі. | | Успішне проведення навчання, дані зберігаються коректно |
|  | 19.11.2022 | Пройдено | – |
|  | Перевірка проведення прогнозування попиту на продукцію | | |
| 005 | 1. Запустити додаток; 2. Провести автентифікацію; 3. Вибрати відповідний пункт меню; 4. Провести прогнозування попиту. | | Успішне проведення прогнозування попиту на продукцію |
|  | 19.11.2022 | Пройдено | – |

## **3.3 Інструкція користувачеві програми**

На початку необхідно запустити додаток «Система управління діяльності підприємства». Після запуску програми буде виведено вікно, де користувачу буде запропоновано ввести ім’я та пароль (рис.3.2).

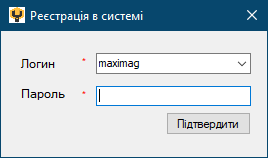


Рисунок 3.2 – Реєстрація користувача в системі

Якщо введена інформація буде не коректною, програма буде попереджувати про це користувача відповідним повідомленням (рис.3.3).

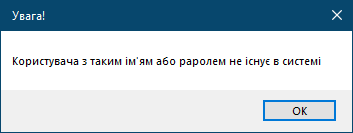


Рисунок 3.3 – Попередження про неправильне введення даних

Для швидшого пошуку можна вибрати ім’я із випадаючого списку або під час введення даних програма поставить у верх списку те ім’я, яке буде вводитись з клавіатури.

Після успішної ідентифікації користувача системи буде відкрите головне вікно програми з основним меню (рис.3.4).

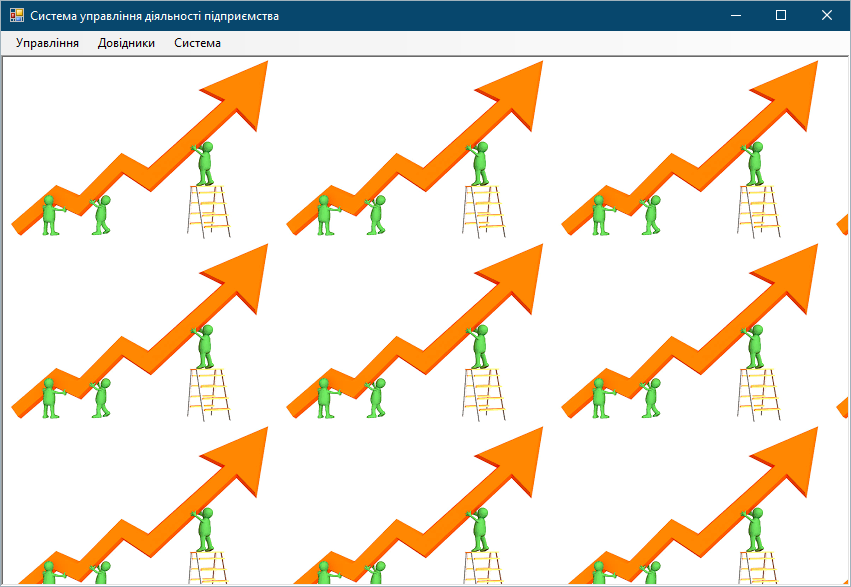


Рисунок 3.4 – Головне меню програми

Для проведення прогнозування попиту на продукцію підприємства необхідно наповнити інформаційну систему, а саме інформацією про: категорії продукції, продукцію, дочірні підприємства, типи підприємств, співробітників та їхні посади.

Програма дозволяє також редагувати та видаляти введені дані про категорії продукції, продукцію, дочірні підприємства, типи підприємств, співробітників та їхні посади.

Для початку потрібно додати інформацію про типи підприємств. Щоб це зробити, необхідно перейти по меню програми «Довідники» –> «Типи підприємств», після чого відкриється відповідне вікно, що показане на рис. 3.5.

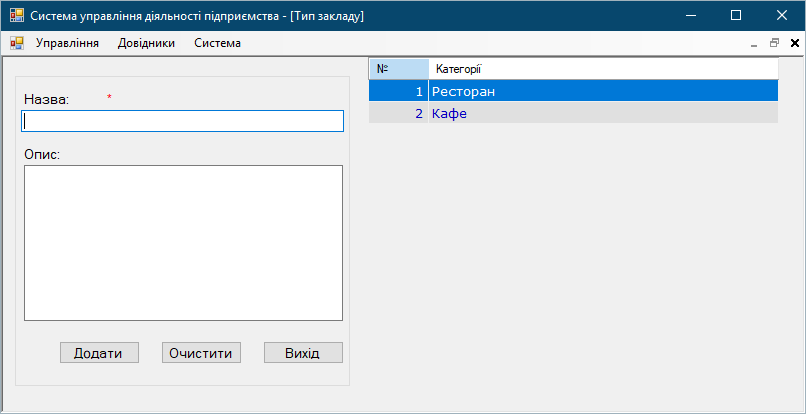


Рисунок 3.5 – Типи підприємств

Також реалізована можливість редагування та видалення даних у випадку якщо це є необхідним. На рис 3.6 зображено вікно для редагування даних про типи підприємств.

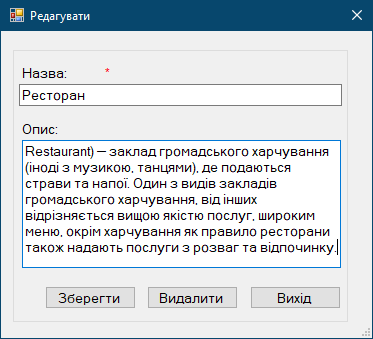


Рисунок 3.6 – Вікно для редагування типів підприємств

Додавання інормації про дочірні підприємства здійснюється при переході по меню програми «Довідники» –> «Дочірні підприємства», що показане на рис. 3.7.

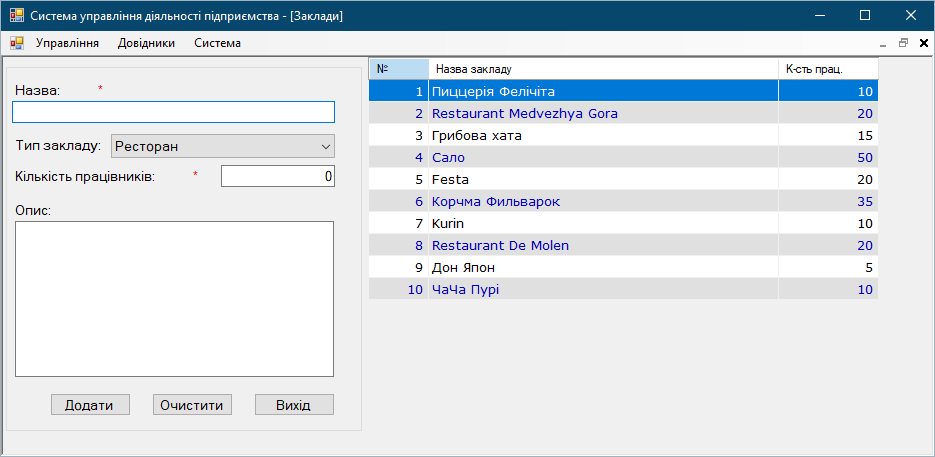


Рисунок 3.7 – Вікно для опрацювання даних дочірніх підприємств

Для того, щоб додати інформацію про співробітника необхідно перейти по меню програми «Довідники» –> «Співробітники», у відповідному вікні (рис. 3.8) ввести всі необхідні дані про співробітника та натиснути кнопку «Додати». Новий співробітник з’явиться у правій частині екрану у списку всіх співробітників.

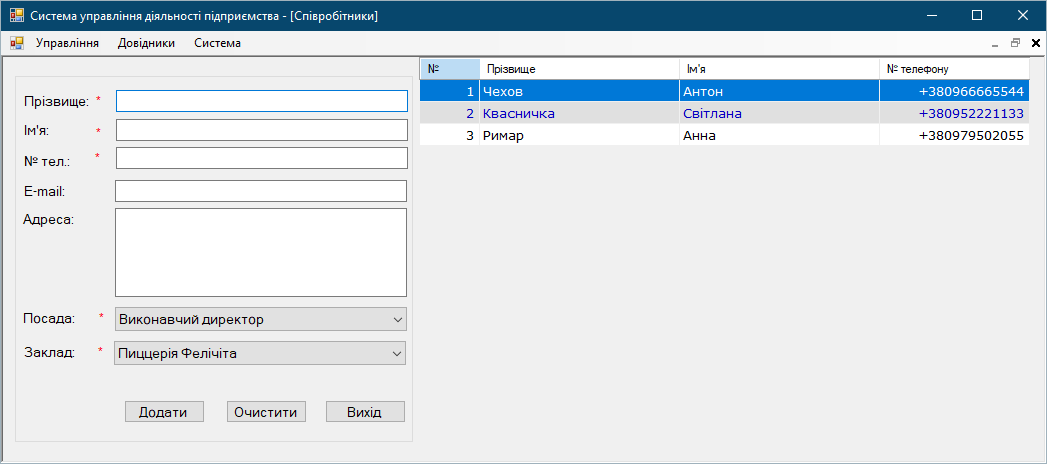


Рисунок 3.8 – Вікно для опрацювання даних про співробітників

Також реалізована можливість редагування та видалення даних у випадку якщо це є необхідним. На рис 3.9 зображено вікно для редагування даних про співробітників.

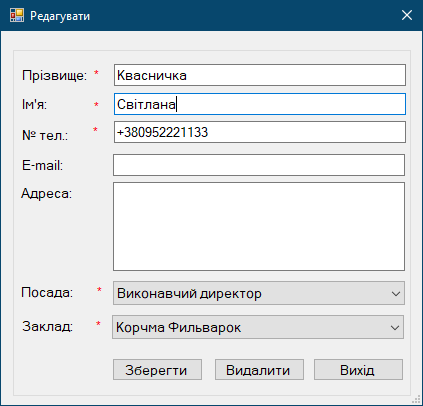


Рисунок 3.9 – Вікно для редагування даних співробітників

Методи опрацювання інформації про посади співробітників, категорії продукції та саму продукцію опрацьовуються аналогічним чином.

Отже, після заповнення довідників програми можна скористатись основними її функціональними можливостями, а саме: проведення симуляції продаж продукції, навчання нейронної мережі на основі згенерованих даних та здійснення прогнозування.

Для проведення симуляції продаж, необхідно перейти до меню програми «Управління» –> «Симуляція продаж», після чого відкриється вінкно, що представлено на рис. 3.10. Задавши параметри симуляції, такі як: дочірнє підприємство, продукція та період продажу, необхідно натиснути кнопку «Симулювати», після чого програма згенерує продаж продукції у вибраному закладі на вибрану продукцію.

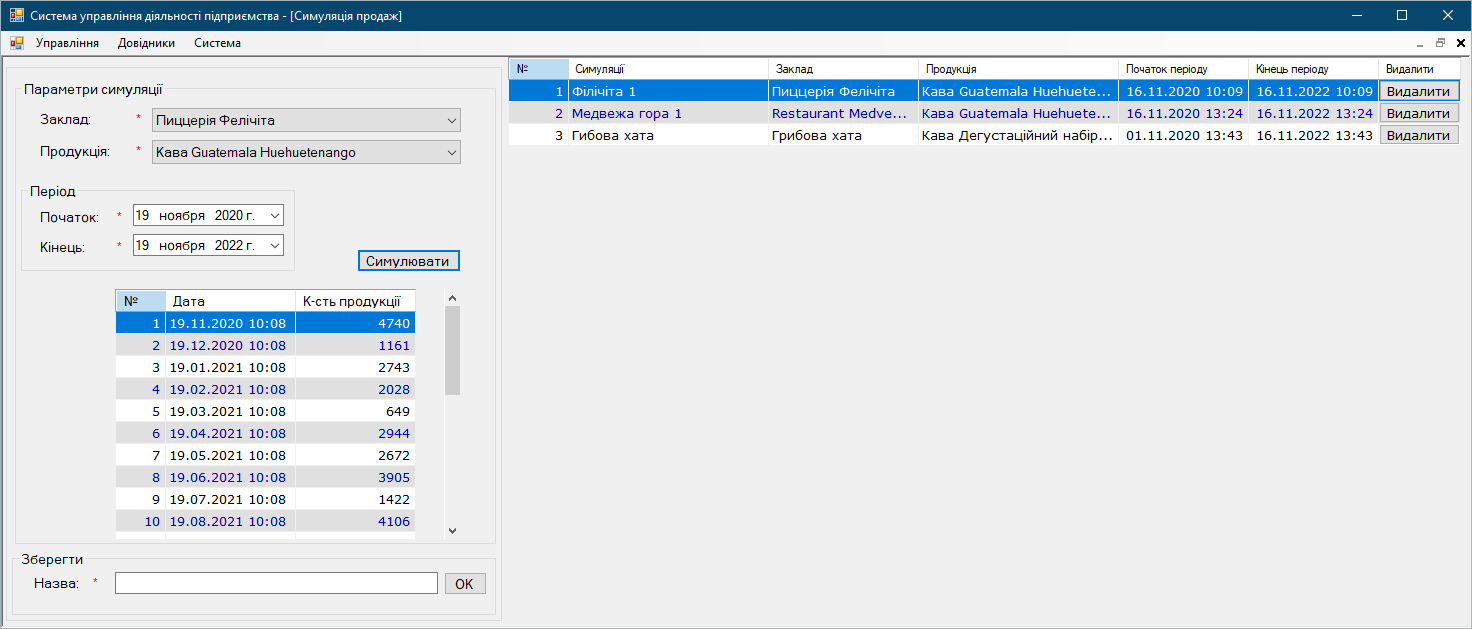


Рисунок 3.10 – Вікно для проведення симуляції продажу

Після цього дані прогнозування можна зберегти ввівши назву для збереження у відповідне поле.

Наступним кроком буде проведення навчання нейронної мережі для цього користувачу системи необхідно перейти до меню програми «Управління» –> «Навчання нейронної мережі», після чого відкриється вінкно, що представлено на рис. 3.11. Далі вибравши відповідну симуляцію та натиснувши кнопку «Навчати» буде проведено навчання нейронної мережі.

Про успіхи навчання нейронної мережі користувача буде повідомлено у відповідному вікні програми.

Дані навчання нейронної мережі можна зберегти у базі даних, для цього необхідно ввести назву проведеного навчання та натиснути кнопку «Ок», після чого програма збереже всі дані навчання.

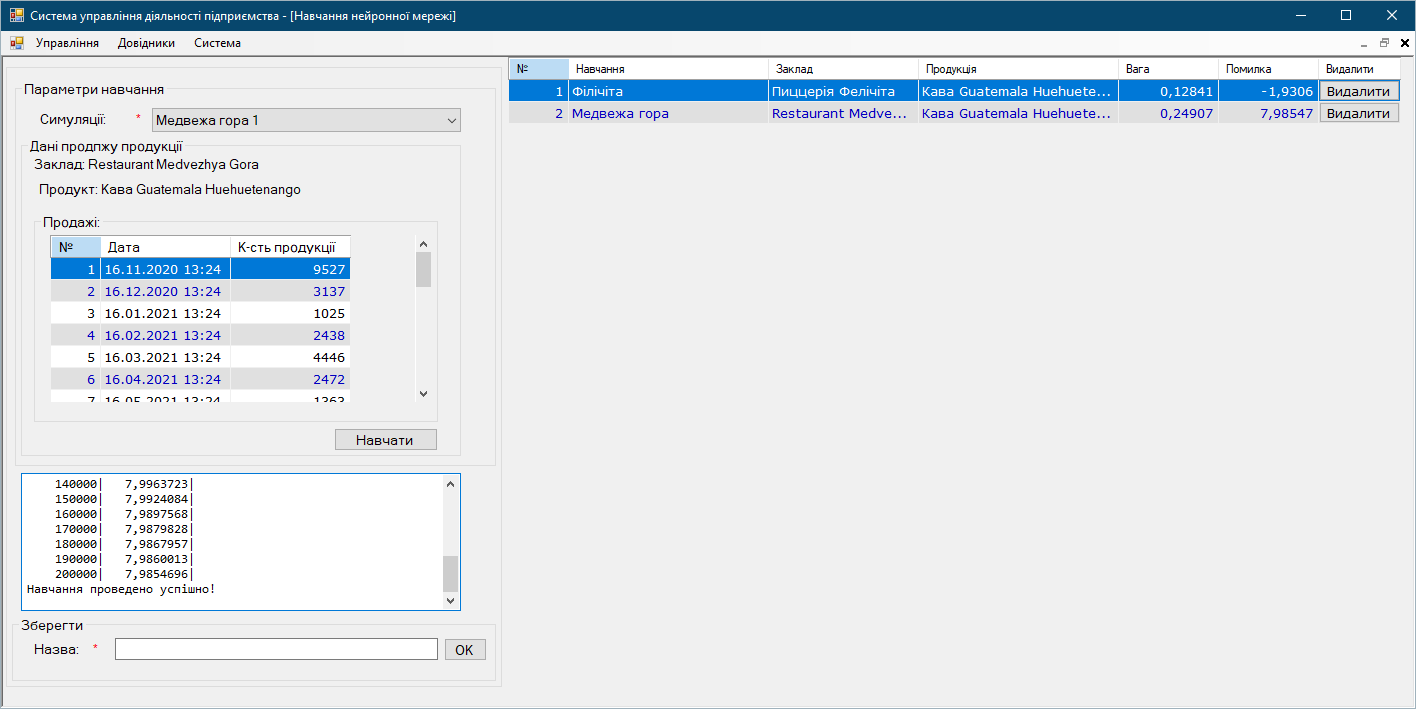


Рисунок 3.11 – Вікно для навчання нейронної мережі

Після того, як дані було збережено, користувач може провести прогнозування попиту на відповідну продукції. Для цьбого, йому необхідно перейти по меню програми «Управління» –> «Прогнозування», після чого відкриється вінкно, що представлено на рис. 3.12. Для проведення прогнозування необхідно вибрати необхідну нейронну мережу та період прогнозування.

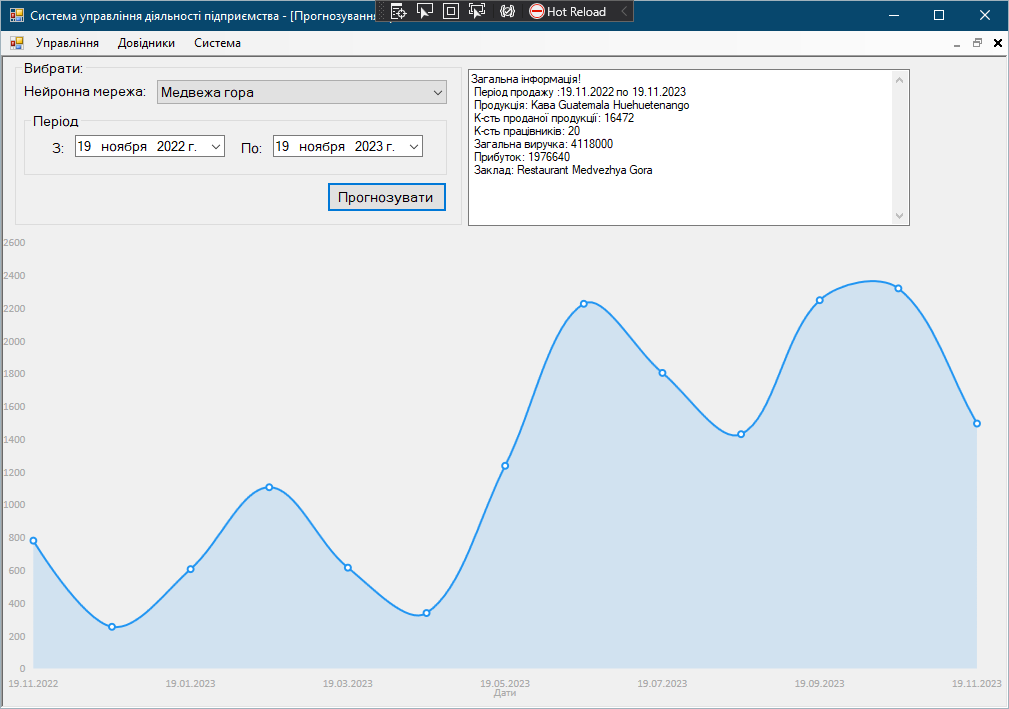


Рисунок 3.12 – Вікно для здійснення прогнозування

Окрім проведення прогнозування програма також виводить додаткову інформацію, а саме:

* період продажу;
* назва продукції;
* інформація про к-сть проданої продукції;
* інформація про к-сть працівників закладу;
* загальна виручка;
* прогнозований прибуток;
* назву закладу.

Користувач із роллю «Системний адміністратор» має доступ до управління обліковими записами системи (рис. 3.13).

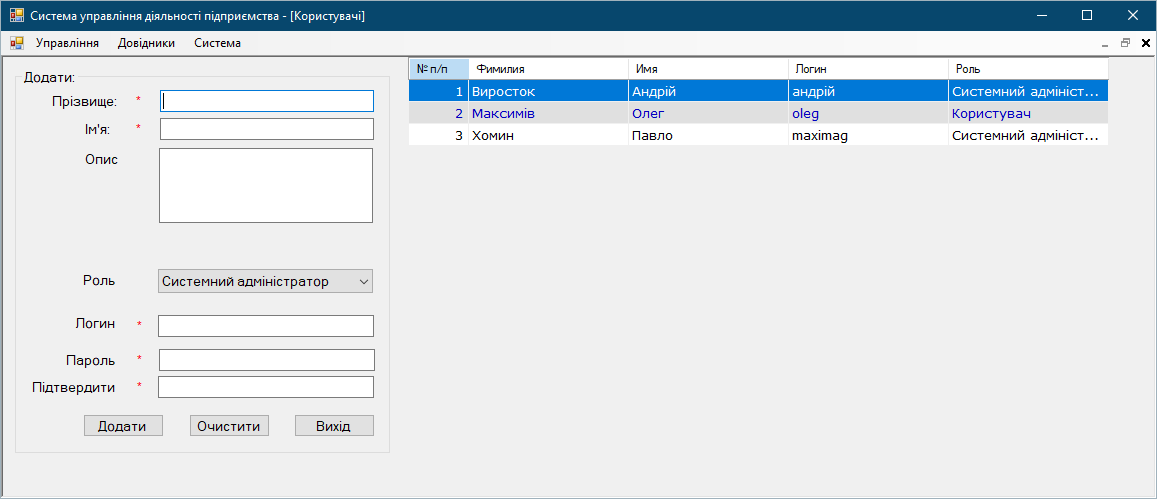


Рисунок 3.13 – Вікно для управління користувачами системи

При необхідності дані та пароль будь-якого користувача можна змінити (рис. 3.14).

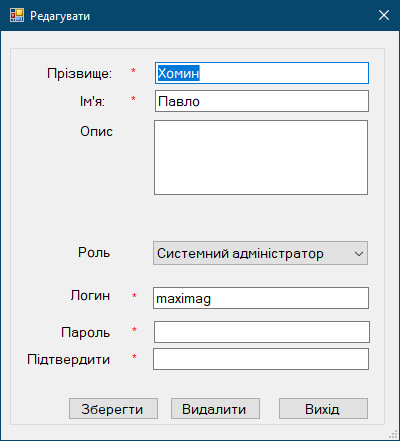


Рисунок 3.14 – Редагування даних користувача системи

Якщо ж поля «Пароль» та «Підтвердити» не співпадають, програма виведе повідомлення про це (рис. 3.15).

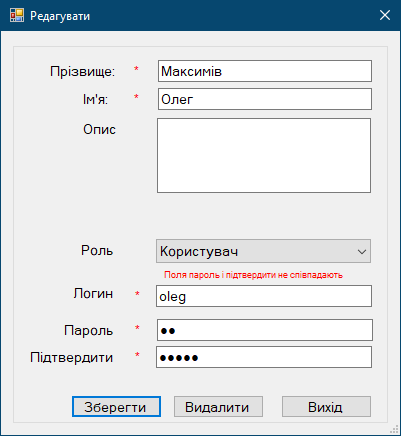


Рисунок 3.15 – Пароль» та «Підтвердити» не співпадають

Також в системі можна бачити активність користувачів, та те, що вони робили в системі. Це можливо зробити за допомогою облікового запису, який має права системного адміністратора. Для цього перейдемо по меню «Система» –> «Системний журнал». В цьому вікні виводяться всі події системи (рис. 3.16).

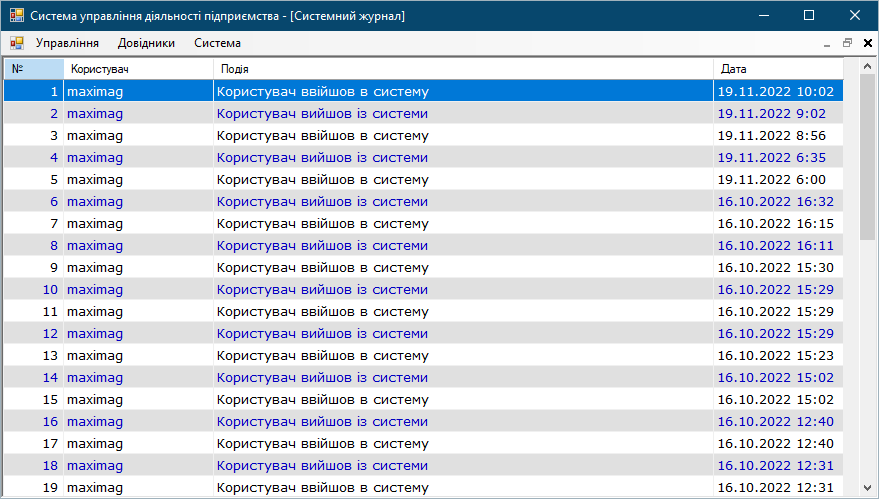


Рисунок 3.16 – Вікно «Системний журнал»

Треба сказати, що дана система є не сильно функціональною, але вона є досить простою в користуванні. ЇЇ інтерфейс є інтуїтивно зрозумілим для користувача.

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

## **3.4 Висновок**

В ході виконання роботи мовою С# в середовищі Visual Studio 2019 реалізовано додаток для підприємства роздрібної торгівлі.

Реалізовано такі функціональні вимоги:

* можливість додавати, редагувати та видаляти облікові записи користувачів;
* можливість додавати, редагувати та видаляти дані про співробітників;
* можливість додавати, редагувати та видаляти дані про посади співробітників;
* можливість додавати, редагувати та видаляти дані про категорії продукції;
* можливість додавати, редагувати та видаляти дані про продукцію;
* можливість додавати, редагувати та видаляти дані про типи підприємств;
* можливість додавати, редагувати та видаляти дані про дочірні підприємства;
* можливість проведення симуляції продаж для дочірніх підприємств;
* можливість проведення навчання нейронної мережі та зберігання даних проведеного навчання;
* можливість проведення прогнозування на попит продукції за вибраний період часу;
* фіксування активності користувачів системи. Документування подій системи у системний журнал.

**ВИСНОВКИ**

В результаті проведеної роботи було розроблено інформаційну інтелектуальну систему управління діяльності підприємства. Розробку інформаційної системи було виконано у середовищі Microsoft Visual Studio 2019 при використанні мови програмування C # та СУБД MS Access. Дана інформаційна система повинна значно полегшити роботу менеджерів та аналітикітим, що розроблене ПЗ має зручний перегляд даних, додавання та вилучення записів та виконання пошуку і фільтрації потрібних записів за допомогою запитів.

У розробленому продукту реалізовано такі функціональні можливості:

* можливість додавати, редагувати та видаляти облікові записи користувачів;
* можливість додавати, редагувати та видаляти дані про співробітників;
* можливість додавати, редагувати та видаляти дані про посади співробітників;
* можливість додавати, редагувати та видаляти дані про категорії продукції;
* можливість додавати, редагувати та видаляти дані про продукцію;
* можливість додавати, редагувати та видаляти дані про типи підприємств;
* можливість додавати, редагувати та видаляти дані про дочірні підприємства;
* можливість проведення симуляції продаж для дочірніх підприємств;
* можливість проведення навчання нейронної мережі та зберігання даних проведеного навчання;
* можливість проведення прогнозування на попит продукції за вибраний період часу;
* фіксування активності користувачів системи. Документування подій системи у системний журнал.

У першому розділі було досліджено основні аспекти інтелектуального управління для підприємства та стратегію розвитку підприємства, що необхідна для його успішного функціонування. Після чого, було здійснено постановку задачі до майбутньої системи.

У другій частині роботи розглянуто дослідження системи та описано її архітектуру. Зроблено аналіз вимог до програмного забезпечення та побудовано use-case діаграми основних прецедентів.

У третій частині було розглянуто засоби розробки ПЗ та виділено його основні переваги та недоліки. Після цього було здійснено тестування програмного забезпечення та розроблено інструкцію користувача програми. Результати тестування були успішними, помилок не було виявлено.

В результаті проведеної роботи вирішено актуальне технічне завдання для здійснення прогнозу продажу товарів для підприємств. У процесі вирішення завдання розроблено інженерну методику процедури проведення симуляції продаж продукції та здійснення прогнозування на майбутній попит, і таким чином поставленої мети досягнуто. У ході виконання роботи отримано такі основні наукові та практичні результати:

1. Розроблено гнучку систему, призначену для проведення прогнозування попиту на продукцію підприємства.
2. Розроблені методи в даній роботі можуть бути використані для широкого класу завдань, тому можливий подальший розвиток розробленого програмного забезпечення.

# **СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Хаврова К.С. Інтелектуалізація діяльності підприємства в контексті усталеного розвитку економіки. Механізми стабілізації й розвитку діяльності підприємств: монографія / під ред. О.О. Шубіна, Л.В. Фролової. Донецьк: Видво «Ноулідж», 2013. С. 254‒263. 4.
2. Хаврова К.С. Визначення основних принципів втручання інтелектуалізації в механізм інноваційного розвитку підприємств. Інноваційні механізми економічного управління підприємством: монографія / за ред. Л.В. Фролової. Кривий Ріг: ФОП Чернявський Д.О., 2016. С. 98‒104.
3. Хаврова К.С. Інтелектуалізація як найважливіша умова розвитку підприємства. Стратегія підприємства в контексті підвищення його конкурентоспроможності: Міжнар.наук.-практ. інтернет-конф., 12-15 лютого 2012 р., Донецьк, С. 164‒166.
4. Андрушків Б. М. Основи менеджменту / Андрушків Б. М., Кузьмін О. Є. – Львів: Світ. 1995. – 296 с. 18.
5. Гринчуцький В.І. Економіка підприємства: навч. посіб. / В.І. Гринчуцький, Е.Т. Карапетян, Б.В. Погріщук. – К.: Центр учбової літератури, 2010. – 304 с. 19.
6. Зьолко О. О. Сутність інтелектуального капіталу: підходи до визначення / О. О. Зьолко // Гуманітарний вісник Здіа. – 2013. – № 53. – С. 13- 20
7. Поліщук О. поняття і роль інтелектуального капіталу в розвитку сучасних підприємств. /О. Поліщук // Матеріали студентської конференції «Соціально-економічний розвиток України на початку ХХІ століття» 2012 р. [Електронний ресурс] – Режим доступу:

https://feu.kneu.edu.ua/ua/confere\_nce/%20conf\_social\_dev%20\_ukr\_12/section4/ polischuk/.

1. Пономаренко В. С. Стратегічне управління підприємством. -- Х.: Основа, 1999. -- 620 с. 24.
2. Редченко К. І. Стратегічний аналіз у бізнесі: Навч. посіб. - 2ге вид., доповн. - Л.: Новий світ -- 2000, 2003. - 272 с
3. Штучний інтелект в задачах кібербезпеки[Електронний ресурс]: – Режим доступу:

<https://learn.ztu.edu.ua/pluginfile.php/188961/mod_resource/content/1/%D0%A8%D0%86_%D0%9A%D0%91_%D0%9B-8_%D0%97%D0%BD%D0%B0%D0%BD%D0%BD%D1%8F.pdf>

1. Архітектура і основні складові [Електронний ресурс]: – Режим доступу: <http://um.co.ua/8/8-15/8-157985.html>
2. Програмне забезпечення інтелектуальних систем [Електронний ресурс]: – Режим доступу: <http://library.nuft.edu.ua/ebook/file/37.21.pdf>
3. Класифікація архітектури[Електронний ресурс]: – Режим доступу: <https://studfile.net/preview/5283206/page:5/>
4. Конспект лекцій [Електронні дані]: – Режим доступу: <https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/25833/1/%D0%9A%D0%BE%D0%BD%D1%81%D0%BF%D0%B5%D0%BA%D1%82%20%D0%BB%D0%B5%D0%BA%D1%86%D1%96%D0%B9.pdf>
5. Класифікації знань[Електронний ресурс]: – Режим доступу: <http://ekmair.ukma.edu.ua/bitstream/handle/123456789/8921/Meitus_Podannia_znan.pdf>
6. Guilfогd J. P. The structure of intellect / J. P. Guilfогd // Psychol. Bull. – 1956. – Vol. 5. – P. 267–293.
7. Бази знань інтелектуальних систем [Електронний ресурс]: – Режим доступу: <http://vlp.com.ua/node/6302>
8. Тарасов О.В. Проектування баз даних : навч. посіб. / О.В. Тарасов, В.В. Федько, М.Ю. Лосєв. – Х. : Вид. ХНЕУ, 2011. – 200 с.
9. Тарарака В.Д. Т19 Архітектура комп'ютерних систем: навчальний посібник. – Житомир : ЖДТУ, 2018. – 383 с.
10. Валецька Т.М. Комп'ютерні мережі. Апаратні засоби: навчальний посібник.-К.: Центр навчальної літератури, 2007. – 208 с.
11. Деякі аспекти застосування UML при розробленні складних програмних систем [Електронний ресурс]. – Акименко А.М. – Режим доступу: <http://vistnic.stu.cn.ua/index.pl?task=arcl&l=ru&j=8&id=21>
12. Методологiя IDEF0 [Електронний ресурс] – Режим доступу: <https://stud.com.ua/87184/ekonomika/metodologiya_idef0>
13. Best ERP Software 2020: Top Rated ERP Systems Comparison [Електронний ресурс]. – Режим доступу до ресурсу:

<https://www.softwaretestinghelp.com/besterp-software-systems/>

1. UML examples and algorithms [Електронний ресурс] – Режим доступу: <http://www.omg-portal.ru/articlematerial17>
2. Perkins B. What is ERP? Key features of top enterprise resource planning systems [Електронний ресурс]: – Режим доступу: <https://www.cio.com/article/2439502/what-is-erp-key-features-of-top-enterprise-resourceplanning-systems.html>
3. .MATT WATSON. What is C# used for? [Електронний ресурс]: – Режим доступу: <https://stackify.com/what-is-c-used-for/> .
4. Користувацький інтерфейс операційної системи Windows. [Електронний ресурс]: – Режим доступу:

https://pidruchniki.com/1402040448804/dokumentoznavstvo/koristuvatskiy\_in terfeys\_operatsiynoyi\_sistemi\_windows

1. Етапи розробки користувацького інтерфейсу. [Електронний ресурс]: – Режим доступу: <https://studopedia.su/12_23303_etapi-rozrobki-koristuvatskogo-interfeysuIteratsiyna-priroda-rozrobki.html>.
2. 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>
3. Data-Access Layer? [Електронний ресурс]: – Режим доступу: <https://www.geeksforgeeks.org/data-access-layer/>

# **ДОДАТОК А. Лістинги програм**

Лістинг 1. Код класу «CategorysProvider»

using RetailTradeEnterprise.AppCode;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.OleDb;

namespace RetailTradeEnterprise.DAO {

class CategorysProvider {

private string \_ConnString = System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

public void InsertCategorys(string CategorysName, string Description) {

string SqlString = "INSERT INTO Categorys (CategorysName, Description" +

") Values(?, ?)";

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("CategorysName", CategorysName);

cmd.Parameters.AddWithValue("Description", Description);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

public List<Categorys> GetAllCategorys() {

int i = 0;

string SqlString = "SELECT CategorysId, CategorysName, Description " +

"FROM Categorys";

List<Categorys> listCategorys = new List<Categorys>();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

using (OleDbDataReader reader = cmd.ExecuteReader()) {

while (reader.Read()) {

Categorys oneCategorys = new Categorys();

oneCategorys.Number = ++i;

oneCategorys.CategorysId = Convert.ToInt32(reader["CategorysId"].ToString());

oneCategorys.CategorysName = reader["CategorysName"].ToString();

oneCategorys.Description = reader["Description"].ToString();

listCategorys.Add(oneCategorys);

}

}

conn.Close();

}

}

if (listCategorys.Count == 0) {

Categorys noCategorys = new Categorys();

noCategorys.CategorysId = 0;

noCategorys.Message = NamesMy.NoDataNames.NoDataInCategorys;

listCategorys.Add(noCategorys);

}

return listCategorys;

}

public Categorys SelectedCategorysByCategorysId(int CategorysId) {

string SqlString = "SELECT CategorysId, CategorysName, Description " +

"FROM Categorys Where CategorysId=" + CategorysId.ToString();

Categorys oneCategorys = new Categorys();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

using (OleDbDataReader reader = cmd.ExecuteReader()) {

while (reader.Read()) {

oneCategorys.CategorysId = Convert.ToInt32(reader["CategorysId"].ToString());

oneCategorys.CategorysName = reader["CategorysName"].ToString();

oneCategorys.Description = reader["Description"].ToString();

}

}

}

conn.Close();

}

return oneCategorys;

}

public void UpdateCategorys(string CategorysName, string Description, int CategorysId) {

string SqlString = "UPDATE Categorys SET CategorysName=?, Description=? " +

"WHERE CategorysId=?";

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("CategorysName", CategorysName);

cmd.Parameters.AddWithValue("Description", Description);

cmd.Parameters.AddWithValue("CategorysId", CategorysId);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

public void DeleteCategorysByCategorysId(int CategorysId) {

string SqlString = "DELETE FROM Categorys WHERE CategorysId=" + CategorysId.ToString();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

}

}

public class Categorys {

private int \_Number;

private int \_CategorysId;

private string \_CategorysName;

private string \_Description;

private string \_Message;

public Categorys() {

\_Number = 0;

\_CategorysId = 0;

\_CategorysName = String.Empty;

\_Description = String.Empty;

\_Message = String.Empty;

}

public int Number {

set { \_Number = value; }

get { return \_Number; }

}

public int CategorysId {

set { \_CategorysId = value; }

get { return \_CategorysId; }

}

public string CategorysName {

set { \_CategorysName = value; }

get { return \_CategorysName; }

}

public string Description {

set { \_Description = value; }

get { return \_Description; }

}

public string Message {

set { \_Message = value; }

get { return \_Message; }

}

}

Лістинг 2. Код класу «RetailTradeBLL»

using RetailTradeEnterprise.DAO;

using RetailTradeEnterprise.Provider;

using System;

using System.Collections.Generic;

namespace RetailTradeEnterprise.BLL {

class RetailTradeBLL {

private ComingGoodsProvoder \_ComingGoodsProvider = new ComingGoodsProvoder();

private ProvidersProvider \_ProvidersProvider = new ProvidersProvider();

private CustomersProvider \_CustomersProvider = new CustomersProvider();

private HistoryComingGoodsProvider \_HistoryComingGoodsProviders = new HistoryComingGoodsProvider();

private DischargeGoodsProvider \_DischargeGoodsProvider = new DischargeGoodsProvider();

private WriteOffProvider \_WriteOffProvider = new WriteOffProvider();

public List<ComingGoods> GetAllSearchComingGoodsByComingGoodsName(string SearchComingGoodsName) {

List<ComingGoods> allComingGoods = new List<ComingGoods>();

List<ComingGoods> searchComingGoods = new List<ComingGoods>();

allComingGoods = \_ComingGoodsProvider.GetAllComingGoods();

for (int i = 0; i < allComingGoods.Count; i++) {

if (allComingGoods[i].GoodsName.ToLower().Contains(SearchComingGoodsName.ToLower())) {

searchComingGoods.Add(allComingGoods[i]);

}

}

return searchComingGoods;

}

public List<Customers> GetAllSearchCustomersByFIO(string SearchFIO) {

List<Customers> allCustomers = new List<Customers>();

List<Customers> searchCustomers = new List<Customers>();

allCustomers = \_CustomersProvider.GetAllCustomers();

for (int i = 0; i < allCustomers.Count; i++) {

if (allCustomers[i].FIO.ToLower().Contains(SearchFIO.ToLower())) {

searchCustomers.Add(allCustomers[i]);

}

}

return searchCustomers;

}

public List<Providers> GetAllSearchProvidersByFIO(string SearchFIO) {

List<Providers> allProviders = new List<Providers>();

List<Providers> searchProviders = new List<Providers>();

allProviders = \_ProvidersProvider.GetAllProviders();

for (int i = 0; i < allProviders.Count; i++) {

if (allProviders[i].FIO.ToLower().Contains(SearchFIO.ToLower())) {

searchProviders.Add(allProviders[i]);

}

}

return searchProviders;

}

public List<HistoryComingGoods> GetHistoryComingGoodsByPeriod(DateTime StartDeliveryDate, DateTime EndDeliveryDate) {

List<HistoryComingGoods> allByes = new List<HistoryComingGoods>();

List<HistoryComingGoods> selectedByesList = new List<HistoryComingGoods>();

allByes = \_HistoryComingGoodsProviders.GetAllHistoryComingGoods();

for (int i = 0; i < allByes.Count; i++) {

if (allByes[i].BuysDate > StartDeliveryDate && allByes[i].BuysDate < EndDeliveryDate) {

selectedByesList.Add(allByes[i]);

}

}

return selectedByesList;

}

public List<DischargeGoods> GetDischargeGoodsByPeriod(DateTime StartDeliveryDate, DateTime EndDeliveryDate) {

List<DischargeGoods> allDischargeGoods = new List<DischargeGoods>();

List<DischargeGoods> selectedDischargeGoodsList = new List<DischargeGoods>();

allDischargeGoods = \_DischargeGoodsProvider.GetAllDischargeGoods();

for (int i = 0; i < allDischargeGoods.Count; i++) {

if (allDischargeGoods[i].DateOfSale > StartDeliveryDate && allDischargeGoods[i].DateOfSale < EndDeliveryDate) {

selectedDischargeGoodsList.Add(allDischargeGoods[i]);

}

}

return selectedDischargeGoodsList;

}

public List<WriteOff> GetWriteOffByPeriod(DateTime StartDeliveryDate, DateTime EndDeliveryDate) {

List<WriteOff> allWriteOff = new List<WriteOff>();

List<WriteOff> selectedWriteOffList = new List<WriteOff>();

allWriteOff = \_WriteOffProvider.GetAllWriteOff();

for (int i = 0; i < allWriteOff.Count; i++) {

if (allWriteOff[i].DebitDate > StartDeliveryDate && allWriteOff[i].DebitDate < EndDeliveryDate) {

selectedWriteOffList.Add(allWriteOff[i]);

}

}

return selectedWriteOffList;

}

}

}

Лістинг 3. Код класу «ComingGoodsProvoder»

using RetailTradeEnterprise.AppCode;

using RetailTradeEnterprise.Provider;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.OleDb;

namespace RetailTradeEnterprise.DAO {

class ComingGoodsProvoder {

private string \_ConnString = System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

private CategorysProvider \_CategorysProvider = new CategorysProvider();

private List<Categorys> \_CategorysList = new List<Categorys>();

private ProvidersProvider \_ProvidersProvider = new ProvidersProvider();

private List<Providers> \_ProvidersList = new List<Providers>();

public void InsertComingGoods(int CategorysId, string GoodsName, string BarCode, int Quantity, string Measurements, string Producer,

DateTime BuysDate, DateTime ExpirationdDate, double PurchasePrice, double SellingPrice, int ProvidersId, string Description) {

string SqlString = "INSERT INTO ComingGoods (CategorysId, GoodsName, BarCode, Quantity, Measurements, Producer, " +

"BuysDate, ExpirationdDate, PurchasePrice, SellingPrice, ProvidersId, Description" +

") Values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("CategorysId", CategorysId);

cmd.Parameters.AddWithValue("GoodsName", GoodsName);

cmd.Parameters.AddWithValue("BarCode", BarCode);

cmd.Parameters.AddWithValue("Quantity", Quantity);

cmd.Parameters.AddWithValue("Measurements", Measurements);

cmd.Parameters.AddWithValue("Producer", Producer);

cmd.Parameters.AddWithValue("BuysDate", BuysDate.ToString());

cmd.Parameters.AddWithValue("ExpirationdDate", ExpirationdDate.ToString());

cmd.Parameters.AddWithValue("PurchasePrice", PurchasePrice);

cmd.Parameters.AddWithValue("SellingPrice", SellingPrice);

cmd.Parameters.AddWithValue("ProvidersId", ProvidersId);

cmd.Parameters.AddWithValue("Description", Description);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

public List<ComingGoods> GetComingGoodsByCategorysId(int CategorysId) {

\_ProvidersList = \_ProvidersProvider.GetAllProviders();

\_CategorysList = \_CategorysProvider.GetAllCategorys();

int i = 0;

string SqlString = "SELECT \* FROM ComingGoods WHERE CategorysId=" + CategorysId;

List<ComingGoods> listComingGoods = new List<ComingGoods>();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

using (OleDbDataReader reader = cmd.ExecuteReader()) {

while (reader.Read()) {

ComingGoods oneComingGoods = new ComingGoods();

oneComingGoods.Number = ++i;

oneComingGoods.ComingGoodsId = Convert.ToInt32(reader["ComingGoodsId"]);

oneComingGoods.CategorysId = Convert.ToInt32(reader["CategorysId"]);

oneComingGoods.GoodsName = reader["GoodsName"].ToString();

oneComingGoods.BarCode = reader["BarCode"].ToString();

oneComingGoods.Quantity = Convert.ToInt32(reader["Quantity"]);

oneComingGoods.Measurements = reader["Measurements"].ToString();

oneComingGoods.Producer = reader["Producer"].ToString();

oneComingGoods.BuysDate = Convert.ToDateTime(reader["BuysDate"]);

oneComingGoods.ExpirationdDate = Convert.ToDateTime(reader["ExpirationdDate"]);

oneComingGoods.PurchasePrice = Convert.ToDouble(reader["PurchasePrice"]);

oneComingGoods.SellingPrice = Convert.ToDouble(reader["SellingPrice"]);

oneComingGoods.ProvidersId = Convert.ToInt32(reader["ProvidersId"]);

oneComingGoods.Description = reader["Description"].ToString();

listComingGoods.Add(oneComingGoods);

}

}

conn.Close();

}

}

if (listComingGoods.Count == 0) {

ComingGoods noComingGoods = new ComingGoods();

noComingGoods.ComingGoodsId = 0;

noComingGoods.CategorysName = NamesMy.NoDataNames.NoDataInComingGoods;

listComingGoods.Add(noComingGoods);

} else {

for (int j = 0; j < listComingGoods.Count; j++) {

listComingGoods[j].ProvidersFIO = GetProvidersFIO(listComingGoods[j].ProvidersId, \_ProvidersList);

listComingGoods[j].CategorysName = GetCategorysName(listComingGoods[j].CategorysId, \_CategorysList);

}

}

return listComingGoods;

}

public List<ComingGoods> GetAllComingGoods() {

\_ProvidersList = \_ProvidersProvider.GetAllProviders();

\_CategorysList = \_CategorysProvider.GetAllCategorys();

int i = 0;

string SqlString = "SELECT \* FROM ComingGoods";

List<ComingGoods> listComingGoods = new List<ComingGoods>();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

using (OleDbDataReader reader = cmd.ExecuteReader()) {

while (reader.Read()) {

ComingGoods oneComingGoods = new ComingGoods();

oneComingGoods.Number = ++i;

oneComingGoods.ComingGoodsId = Convert.ToInt32(reader["ComingGoodsId"]);

oneComingGoods.CategorysId = Convert.ToInt32(reader["CategorysId"]);

oneComingGoods.GoodsName = reader["GoodsName"].ToString();

oneComingGoods.BarCode = reader["BarCode"].ToString();

oneComingGoods.Quantity = Convert.ToInt32(reader["Quantity"]);

oneComingGoods.Measurements = reader["Measurements"].ToString();

oneComingGoods.Producer = reader["Producer"].ToString();

oneComingGoods.BuysDate = Convert.ToDateTime(reader["BuysDate"]);

oneComingGoods.ExpirationdDate = Convert.ToDateTime(reader["ExpirationdDate"]);

oneComingGoods.PurchasePrice = Convert.ToDouble(reader["PurchasePrice"]);

oneComingGoods.SellingPrice = Convert.ToDouble(reader["SellingPrice"]);

oneComingGoods.ProvidersId = Convert.ToInt32(reader["ProvidersId"]);

oneComingGoods.Description = reader["Description"].ToString();

listComingGoods.Add(oneComingGoods);

}

}

conn.Close();

}

}

if (listComingGoods.Count == 0) {

ComingGoods noComingGoods = new ComingGoods();

noComingGoods.ComingGoodsId = 0;

noComingGoods.Message = NamesMy.NoDataNames.NoDataInComingGoods;

listComingGoods.Add(noComingGoods);

} else {

for (int j = 0; j < listComingGoods.Count; j++) {

listComingGoods[j].ProvidersFIO = GetProvidersFIO(listComingGoods[j].ProvidersId, \_ProvidersList);

listComingGoods[j].CategorysName = GetCategorysName(listComingGoods[j].CategorysId, \_CategorysList);

}

}

return listComingGoods;

}

private string GetProvidersFIO(int ProvidersId, List<Providers> ProvidersList) {

for (int i = 0; i < ProvidersList.Count; i++) {

if (ProvidersId == ProvidersList[i].ProvidersId) {

return ProvidersList[i].FIO;

}

}

return "";

}

private string GetCategorysName(int CategorysId, List<Categorys> CategorysList) {

for (int i = 0; i < CategorysList.Count; i++) {

if (CategorysId == CategorysList[i].CategorysId) {

return CategorysList[i].CategorysName;

}

}

return "";

}

public ComingGoods SelectedComingGoodsByComingGoodsId(int ComingGoodsId) {

string SqlString = "SELECT \* " +

"FROM ComingGoods Where ComingGoodsId=" + ComingGoodsId.ToString();

ComingGoods oneComingGoods = new ComingGoods();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

using (OleDbDataReader reader = cmd.ExecuteReader()) {

while (reader.Read()) {

oneComingGoods.ComingGoodsId = Convert.ToInt32(reader["ComingGoodsId"]);

oneComingGoods.CategorysId = Convert.ToInt32(reader["CategorysId"]);

oneComingGoods.GoodsName = reader["GoodsName"].ToString();

oneComingGoods.BarCode = reader["BarCode"].ToString();

oneComingGoods.Quantity = Convert.ToInt32(reader["Quantity"]);

oneComingGoods.Measurements = reader["Measurements"].ToString();

oneComingGoods.Producer = reader["Producer"].ToString();

oneComingGoods.BuysDate = Convert.ToDateTime(reader["BuysDate"]);

oneComingGoods.ExpirationdDate = Convert.ToDateTime(reader["ExpirationdDate"]);

oneComingGoods.PurchasePrice = Convert.ToDouble(reader["PurchasePrice"]);

oneComingGoods.SellingPrice = Convert.ToDouble(reader["SellingPrice"]);

oneComingGoods.ProvidersId = Convert.ToInt32(reader["ProvidersId"]);

oneComingGoods.Description = reader["Description"].ToString();

}

}

}

conn.Close();

}

return oneComingGoods;

}

public void UpdateComingGoods(int CategorysId, string GoodsName, string BarCode, int Quantity, string Measurements, string Producer,

DateTime BuysDate, DateTime ExpirationdDate, double PurchasePrice, double SellingPrice, int ProvidersId, string Description, int ComingGoodsId) {

string SqlString = "UPDATE ComingGoods SET CategorysId=?, GoodsName=?, BarCode=?, Quantity=?, Measurements=?, Producer=?," +

"BuysDate=?, ExpirationdDate=?, PurchasePrice=?, SellingPrice=?, ProvidersId=?, Description=? " +

"WHERE ComingGoodsId=?";

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("CategorysId", CategorysId);

cmd.Parameters.AddWithValue("GoodsName", GoodsName);

cmd.Parameters.AddWithValue("BarCode", BarCode);

cmd.Parameters.AddWithValue("Quantity", Quantity);

cmd.Parameters.AddWithValue("Measurements", Measurements);

cmd.Parameters.AddWithValue("Producer", Producer);

cmd.Parameters.AddWithValue("BuysDate", BuysDate.ToString());

cmd.Parameters.AddWithValue("ExpirationdDate", ExpirationdDate.ToString());

cmd.Parameters.AddWithValue("PurchasePrice", PurchasePrice);

cmd.Parameters.AddWithValue("SellingPrice", SellingPrice);

cmd.Parameters.AddWithValue("ProvidersId", ProvidersId);

cmd.Parameters.AddWithValue("Description", Description);

cmd.Parameters.AddWithValue("ComingGoodsId", ComingGoodsId);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

public void DeleteComingGoodsByComingGoodsId(int ComingGoodsId) {

string SqlString = "DELETE FROM ComingGoods WHERE ComingGoodsId=" + ComingGoodsId.ToString();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

public void PlusGoods(int ComingGoodsId, int Quantity) {

string SqlString = "UPDATE ComingGoods SET Quantity=? WHERE ComingGoodsId=?";

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("Quantity", Quantity);

cmd.Parameters.AddWithValue("ComingGoodsId", ComingGoodsId);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

public void MinusGoods(int ComingGoodsId, int Quantity) {

int allQuantity = SelectedComingGoodsByComingGoodsId(ComingGoodsId).Quantity - Quantity;

string SqlString = "UPDATE ComingGoods SET Quantity=? WHERE ComingGoodsId=?";

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("Quantity", allQuantity);

cmd.Parameters.AddWithValue("GoodsId", ComingGoodsId);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

}

}

public class ComingGoods {

private int \_Number;

private int \_ComingGoodsId;

private int \_CategorysId; //Категорія

private string \_GoodsName; //Назва

private string \_BarCode; //Штрих-код

private int \_Quantity; //Кількість

private string \_Measurements; //Од. виміру

private string \_Producer; //Виробник

private DateTime \_BuysDate; //Дата постачання

private DateTime \_ExpirationdDate; //Термін придатності

private double \_PurchasePrice; //Ціна закупівлі

private double \_SellingPrice; //Ціна продажу

private int \_ProvidersId; //Постачальник

private string \_Description; //Опис

private string \_ProvidersFIO;

private string \_CategorysName;

private string \_Message;

public ComingGoods() {

\_Number = 0;

\_ComingGoodsId = 0;

\_CategorysId = 0;

\_GoodsName = String.Empty;

\_BarCode = String.Empty;

\_Quantity = 0;

\_Measurements = String.Empty;

\_Producer = String.Empty;

\_BuysDate = new DateTime();

\_ExpirationdDate = new DateTime();

\_PurchasePrice = 0.0;

\_SellingPrice = 0.0;

\_ProvidersId = 0;

\_Description = String.Empty;

\_ProvidersFIO = String.Empty;

\_CategorysName = String.Empty;

\_Message = String.Empty;

}

public int Number {

set { \_Number = value; }

get { return \_Number; }

}

public int ComingGoodsId {

set { \_ComingGoodsId = value; }

get { return \_ComingGoodsId; }

}

public int CategorysId {

set { \_CategorysId = value; }

get { return \_CategorysId; }

}

public string GoodsName {

set { \_GoodsName = value; }

get { return \_GoodsName; }

}

public string BarCode {

set { \_BarCode = value; }

get { return \_BarCode; }

}

public int Quantity {

set { \_Quantity = value; }

get { return \_Quantity; }

}

public string Measurements {

set { \_Measurements = value; }

get { return \_Measurements; }

}

public string Producer {

set { \_Producer = value; }

get { return \_Producer; }

}

public DateTime BuysDate {

set { \_BuysDate = value; }

get { return \_BuysDate; }

}

public DateTime ExpirationdDate {

set { \_ExpirationdDate = value; }

get { return \_ExpirationdDate; }

}

public double PurchasePrice {

set { \_PurchasePrice = value; }

get { return \_PurchasePrice; }

}

public double SellingPrice {

set { \_SellingPrice = value; }

get { return \_SellingPrice; }

}

public string Description {

set { \_Description = value; }

get { return \_Description; }

}

public string ProvidersFIO {

set { \_ProvidersFIO = value; }

get { return \_ProvidersFIO; }

}

public string CategorysName {

set { \_CategorysName = value; }

get { return \_CategorysName; }

}

public int ProvidersId {

set { \_ProvidersId = value; }

get { return \_ProvidersId; }

}

public string Message {

set { \_Message = value; }

get { return \_Message; }

}

}

Лістинг 4. Код класу «CustomersProvider»

using RetailTradeEnterprise.AppCode;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.OleDb;

namespace RetailTradeEnterprise.Provider {

class CustomersProvider {

private string \_ConnString = System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

public void InsertCustomers(string FirstName, string LastName, string Phone, string Address, string Email) {

string SqlString = "INSERT INTO Customers (FirstName, LastName, Phone, Address, " +

"Email) Values(?, ?, ?, ?, ?)";

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("FirstName", FirstName);

cmd.Parameters.AddWithValue("LastName", LastName);

cmd.Parameters.AddWithValue("Phone", Phone);

cmd.Parameters.AddWithValue("Address", Address);

cmd.Parameters.AddWithValue("Email", Email);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

public List<Customers> GetAllCustomers() {

int i = 0;

string SqlString = "SELECT \* FROM Customers";

List<Customers> listAllCustomers = new List<Customers>();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

using (OleDbDataReader reader = cmd.ExecuteReader()) {

while (reader.Read()) {

Customers oneCustomers = new Customers();

oneCustomers.Number = ++i;

oneCustomers.CustomersId = Convert.ToInt32(reader["CustomersId"].ToString());

oneCustomers.FirstName = reader["FirstName"].ToString();

oneCustomers.LastName = reader["LastName"].ToString();

oneCustomers.FIO = oneCustomers.LastName + " " + oneCustomers.FirstName;

oneCustomers.Phone = reader["Phone"].ToString();

oneCustomers.Address = reader["Address"].ToString();

oneCustomers.Email = reader["Email"].ToString();

listAllCustomers.Add(oneCustomers);

}

}

conn.Close();

}

}

if (listAllCustomers.Count == 0) {

Customers noDataInCustomers = new Customers();

noDataInCustomers.CustomersId = 0;

noDataInCustomers.Message = NamesMy.NoDataNames.NoDataInCustomers;

listAllCustomers.Add(noDataInCustomers);

}

return listAllCustomers;

}

public Customers SelectedCustomersByCustomersId(int CustomersId) {

string SqlString = "SELECT \* FROM Customers Where CustomersId=" + CustomersId.ToString();

Customers oneCustomers = new Customers();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

using (OleDbDataReader reader = cmd.ExecuteReader()) {

while (reader.Read()) {

oneCustomers.CustomersId = Convert.ToInt32(reader["CustomersId"].ToString());

oneCustomers.FirstName = reader["FirstName"].ToString();

oneCustomers.LastName = reader["LastName"].ToString();

oneCustomers.FIO = oneCustomers.LastName + " " + oneCustomers.FirstName;

oneCustomers.Phone = reader["Phone"].ToString();

oneCustomers.Address = reader["Address"].ToString();

oneCustomers.Email = reader["Email"].ToString();

}

}

}

conn.Close();

}

return oneCustomers;

}

public void UpdateCustomers(string FirstName, string LastName, string Phone,

string Address, string Email, int CustomersId) {

string SqlString = "UPDATE Customers SET FirstName=?, LastName=?, Phone=?, Address=?, " +

"Email=? WHERE CustomersId=?";

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("FirstName", FirstName);

cmd.Parameters.AddWithValue("LastName", LastName);

cmd.Parameters.AddWithValue("Phone", Phone);

cmd.Parameters.AddWithValue("Address", Address);

cmd.Parameters.AddWithValue("Email", Email);

cmd.Parameters.AddWithValue("CustomersId", CustomersId);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

public void DeleteCustomersByCustomersId(int CustomersId) {

string SqlString = "DELETE FROM Customers WHERE CustomersId=" + CustomersId.ToString();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

}

}

public class Customers {

private int \_Number;

private int \_CustomersId;

private string \_FirstName;

private string \_LastName;

public string \_FIO;

private string \_Phone;

private string \_Address;

private string \_Email;

private string \_Message;

public Customers() {

\_Number = 0;

\_CustomersId = 0;

\_FirstName = String.Empty;

\_LastName = String.Empty;

\_FIO = String.Empty;

\_Phone = String.Empty;

\_Address = String.Empty;

\_Email = String.Empty;

\_Message = String.Empty;

}

public int Number {

set { \_Number = value; }

get { return \_Number; }

}

public int CustomersId {

set { \_CustomersId = value; }

get { return \_CustomersId; }

}

public string FirstName {

set { \_FirstName = value; }

get { return \_FirstName; }

}

public string LastName {

set { \_LastName = value; }

get { return \_LastName; }

}

public string FIO {

set { \_FIO = value; }

get { return \_FIO; }

}

public string Phone {

set { \_Phone = value; }

get { return \_Phone; }

}

public string Address {

set { \_Address = value; }

get { return \_Address; }

}

public string Email {

set { \_Email = value; }

get { return \_Email; }

}

public string Message {

set { \_Message = value; }

get { return \_Message; }

}

}

Лістинг 5. Код класу «DischargeGoodsProvider»

using RetailTradeEnterprise.AppCode;

using RetailTradeEnterprise.Provider;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.OleDb;

namespace RetailTradeEnterprise.DAO {

class DischargeGoodsProvider {

private string \_ConnString = System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

private CategorysProvider \_CategorysProvider = new CategorysProvider();

private List<Categorys> \_CategorysList = new List<Categorys>();

private CustomersProvider \_CustomersProvider = new CustomersProvider();

private List<Customers> \_CustomersList = new List<Customers>();

public void InsertDischargeGoods(int CategorysId, string GoodsName, string BarCode, int Quantity, string Measurements, string Producer,

DateTime DateOfSale, DateTime ExpirationdDate, double SellingPrice, int CustomersId) {

string SqlString = "INSERT INTO DischargeGoods (CategorysId, GoodsName, BarCode, Quantity, Measurements, Producer, " +

"DateOfSale, ExpirationdDate, SellingPrice, CustomersId" +

") Values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("CategorysId", CategorysId);

cmd.Parameters.AddWithValue("GoodsName", GoodsName);

cmd.Parameters.AddWithValue("BarCode", BarCode);

cmd.Parameters.AddWithValue("Quantity", Quantity);

cmd.Parameters.AddWithValue("Measurements", Measurements);

cmd.Parameters.AddWithValue("Producer", Producer);

cmd.Parameters.AddWithValue("DateOfSale", DateOfSale.ToString());

cmd.Parameters.AddWithValue("ExpirationdDate", ExpirationdDate.ToString());

cmd.Parameters.AddWithValue("SellingPrice", SellingPrice);

cmd.Parameters.AddWithValue("CustomersId", CustomersId);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

public List<DischargeGoods> GetDischargeGoodsByCategorysId(int CategorysId) {

\_CustomersList = \_CustomersProvider.GetAllCustomers();

\_CategorysList = \_CategorysProvider.GetAllCategorys();

int i = 0;

string SqlString = "SELECT \* FROM DischargeGoods WHERE CategorysId=" + CategorysId;

List<DischargeGoods> listDischargeGoods = new List<DischargeGoods>();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

using (OleDbDataReader reader = cmd.ExecuteReader()) {

while (reader.Read()) {

DischargeGoods oneDischargeGoods = new DischargeGoods();

oneDischargeGoods.Number = ++i;

oneDischargeGoods.DischargeGoodsId = Convert.ToInt32(reader["DischargeGoodsId"]);

oneDischargeGoods.CategorysId = Convert.ToInt32(reader["CategorysId"]);

oneDischargeGoods.GoodsName = reader["GoodsName"].ToString();

oneDischargeGoods.BarCode = reader["BarCode"].ToString();

oneDischargeGoods.Quantity = Convert.ToInt32(reader["Quantity"]);

oneDischargeGoods.Measurements = reader["Measurements"].ToString();

oneDischargeGoods.Producer = reader["Producer"].ToString();

oneDischargeGoods.DateOfSale = Convert.ToDateTime(reader["DateOfSale"]);

oneDischargeGoods.ExpirationdDate = Convert.ToDateTime(reader["ExpirationdDate"]);

oneDischargeGoods.SellingPrice = Convert.ToDouble(reader["SellingPrice"]);

oneDischargeGoods.CustomersId = Convert.ToInt32(reader["CustomersId"]);

listDischargeGoods.Add(oneDischargeGoods);

}

}

conn.Close();

}

}

if (listDischargeGoods.Count == 0) {

DischargeGoods noDischargeGoods = new DischargeGoods();

noDischargeGoods.DischargeGoodsId = 0;

noDischargeGoods.Message = NamesMy.NoDataNames.NoDataInDischargeGoods;

listDischargeGoods.Add(noDischargeGoods);

} else {

for (int j = 0; j < listDischargeGoods.Count; j++) {

listDischargeGoods[j].CustomersFIO = GetCustomersFIO(listDischargeGoods[j].CustomersId, \_CustomersList);

listDischargeGoods[j].CategorysName = GetCategorysName(listDischargeGoods[j].CategorysId, \_CategorysList);

}

}

return listDischargeGoods;

}

public List<DischargeGoods> GetAllDischargeGoods() {

\_CustomersList = \_CustomersProvider.GetAllCustomers();

\_CategorysList = \_CategorysProvider.GetAllCategorys();

int i = 0;

string SqlString = "SELECT \* FROM DischargeGoods";

List<DischargeGoods> listDischargeGoods = new List<DischargeGoods>();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

using (OleDbDataReader reader = cmd.ExecuteReader()) {

while (reader.Read()) {

DischargeGoods oneDischargeGoods = new DischargeGoods();

oneDischargeGoods.Number = ++i;

oneDischargeGoods.DischargeGoodsId = Convert.ToInt32(reader["DischargeGoodsId"]);

oneDischargeGoods.CategorysId = Convert.ToInt32(reader["CategorysId"]);

oneDischargeGoods.GoodsName = reader["GoodsName"].ToString();

oneDischargeGoods.BarCode = reader["BarCode"].ToString();

oneDischargeGoods.Quantity = Convert.ToInt32(reader["Quantity"]);

oneDischargeGoods.Measurements = reader["Measurements"].ToString();

oneDischargeGoods.Producer = reader["Producer"].ToString();

oneDischargeGoods.DateOfSale = Convert.ToDateTime(reader["DateOfSale"]);

oneDischargeGoods.ExpirationdDate = Convert.ToDateTime(reader["ExpirationdDate"]);

oneDischargeGoods.SellingPrice = Convert.ToDouble(reader["SellingPrice"]);

oneDischargeGoods.CustomersId = Convert.ToInt32(reader["CustomersId"]);

listDischargeGoods.Add(oneDischargeGoods);

}

}

conn.Close();

}

}

if (listDischargeGoods.Count == 0) {

DischargeGoods noDischargeGoods = new DischargeGoods();

noDischargeGoods.DischargeGoodsId = 0;

noDischargeGoods.Message = NamesMy.NoDataNames.NoDataInDischargeGoods;

listDischargeGoods.Add(noDischargeGoods);

} else {

for (int j = 0; j < listDischargeGoods.Count; j++) {

listDischargeGoods[j].CustomersFIO = GetCustomersFIO(listDischargeGoods[j].CustomersId, \_CustomersList);

listDischargeGoods[j].CategorysName = GetCategorysName(listDischargeGoods[j].CategorysId, \_CategorysList);

}

}

return listDischargeGoods;

}

private string GetCustomersFIO(int CustomersId, List<Customers> CustomersList) {

for (int i = 0; i < CustomersList.Count; i++) {

if (CustomersId == CustomersList[i].CustomersId) {

return CustomersList[i].FIO;

}

}

return "";

}

private string GetCategorysName(int CategorysId, List<Categorys> CategorysList) {

for (int i = 0; i < CategorysList.Count; i++) {

if (CategorysId == CategorysList[i].CategorysId) {

return CategorysList[i].CategorysName;

}

}

return "";

}

public DischargeGoods SelectedDischargeGoodsByDischargeGoodsId(int DischargeGoodsId) {

string SqlString = "SELECT \* " +

"FROM DischargeGoods Where DischargeGoodsId=" + DischargeGoodsId.ToString();

DischargeGoods oneDischargeGoods = new DischargeGoods();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

using (OleDbDataReader reader = cmd.ExecuteReader()) {

while (reader.Read()) {

oneDischargeGoods.DischargeGoodsId = Convert.ToInt32(reader["DischargeGoodsId"]);

oneDischargeGoods.CategorysId = Convert.ToInt32(reader["CategorysId"]);

oneDischargeGoods.GoodsName = reader["GoodsName"].ToString();

oneDischargeGoods.BarCode = reader["BarCode"].ToString();

oneDischargeGoods.Quantity = Convert.ToInt32(reader["Quantity"]);

oneDischargeGoods.Measurements = reader["Measurements"].ToString();

oneDischargeGoods.Producer = reader["Producer"].ToString();

oneDischargeGoods.DateOfSale = Convert.ToDateTime(reader["DateOfSale"]);

oneDischargeGoods.ExpirationdDate = Convert.ToDateTime(reader["ExpirationdDate"]);

oneDischargeGoods.SellingPrice = Convert.ToDouble(reader["SellingPrice"]);

oneDischargeGoods.CustomersId = Convert.ToInt32(reader["CustomersId"]);

}

}

}

conn.Close();

}

return oneDischargeGoods;

}

public void UpdateDischargeGoods(int CategorysId, string GoodsName, string BarCode, int Quantity, string Measurements, string Producer,

DateTime DateOfSale, DateTime ExpirationdDate, double SellingPrice, int CustomersId, int DischargeGoodsId) {

string SqlString = "UPDATE DischargeGoods SET CategorysId=?, GoodsName=?, BarCode=?, Quantity=?, Measurements=?, Producer=?," +

"DateOfSale=?, ExpirationdDate=?, SellingPrice=?, CustomersId=? " +

"WHERE DischargeGoodsId=?";

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("CategorysId", CategorysId);

cmd.Parameters.AddWithValue("GoodsName", GoodsName);

cmd.Parameters.AddWithValue("BarCode", BarCode);

cmd.Parameters.AddWithValue("Quantity", Quantity);

cmd.Parameters.AddWithValue("Measurements", Measurements);

cmd.Parameters.AddWithValue("Producer", Producer);

cmd.Parameters.AddWithValue("DateOfSale", DateOfSale.ToString());

cmd.Parameters.AddWithValue("ExpirationdDate", ExpirationdDate.ToString());

cmd.Parameters.AddWithValue("SellingPrice", SellingPrice);

cmd.Parameters.AddWithValue("CustomersId", CustomersId);

cmd.Parameters.AddWithValue("DischargeGoodsId", DischargeGoodsId);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

public void DeleteDischargeGoodsByDischargeGoodsId(int DischargeGoodsId) {

string SqlString = "DELETE FROM DischargeGoods WHERE DischargeGoodsId=" + DischargeGoodsId.ToString();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

}

}

public class DischargeGoods {

private int \_Number;

private int \_DischargeGoodsId;

private int \_CategorysId; //Категорія

private string \_GoodsName; //Назва

private string \_BarCode; //Штрих-код

private int \_Quantity; //Кількість

private string \_Measurements; //Од. виміру

private string \_Producer; //Виробник

private DateTime \_DateOfSale; //Дата постачання

private DateTime \_ExpirationdDate; //Термін придатності

private double \_SellingPrice; //Ціна продажу

private int \_CustomersId; //Постачальник

private string \_CustomersFIO;

private string \_CategorysName;

private string \_Message;

public DischargeGoods() {

\_Number = 0;

\_DischargeGoodsId = 0;

\_CategorysId = 0;

\_GoodsName = String.Empty;

\_BarCode = String.Empty;

\_Quantity = 0;

\_Measurements = String.Empty;

\_Producer = String.Empty;

\_DateOfSale = new DateTime();

\_ExpirationdDate = new DateTime();

\_SellingPrice = 0.0;

\_CustomersId = 0;

\_CustomersFIO = String.Empty;

\_CategorysName = String.Empty;

\_Message = String.Empty;

}

public int Number {

set { \_Number = value; }

get { return \_Number; }

}

public int DischargeGoodsId {

set { \_DischargeGoodsId = value; }

get { return \_DischargeGoodsId; }

}

public int CategorysId {

set { \_CategorysId = value; }

get { return \_CategorysId; }

}

public string GoodsName {

set { \_GoodsName = value; }

get { return \_GoodsName; }

}

public string BarCode {

set { \_BarCode = value; }

get { return \_BarCode; }

}

public int Quantity {

set { \_Quantity = value; }

get { return \_Quantity; }

}

public string Measurements {

set { \_Measurements = value; }

get { return \_Measurements; }

}

public string Producer {

set { \_Producer = value; }

get { return \_Producer; }

}

public DateTime DateOfSale {

set { \_DateOfSale = value; }

get { return \_DateOfSale; }

}

public DateTime ExpirationdDate {

set { \_ExpirationdDate = value; }

get { return \_ExpirationdDate; }

}

public double SellingPrice {

set { \_SellingPrice = value; }

get { return \_SellingPrice; }

}

public string CustomersFIO {

set { \_CustomersFIO = value; }

get { return \_CustomersFIO; }

}

public string CategorysName {

set { \_CategorysName = value; }

get { return \_CategorysName; }

}

public int CustomersId {

set { \_CustomersId = value; }

get { return \_CustomersId; }

}

public string Message {

set { \_Message = value; }

get { return \_Message; }

}

}

Лістинг 6. Код класу «HistoryComingGoodsProvider»

using RetailTradeEnterprise.AppCode;

using RetailTradeEnterprise.Provider;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.OleDb;

namespace RetailTradeEnterprise.DAO {

class HistoryComingGoodsProvider {

private string \_ConnString = System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

private CategorysProvider \_CategorysProvider = new CategorysProvider();

private List<Categorys> \_CategorysList = new List<Categorys>();

private ProvidersProvider \_ProvidersProvider = new ProvidersProvider();

private List<Providers> \_ProvidersList = new List<Providers>();

public void InsertHistoryComingGoods(int CategorysId, string GoodsName, string BarCode, int Quantity, string Measurements, string Producer,

DateTime BuysDate, DateTime ExpirationdDate, double PurchasePrice, double SellingPrice, int ProvidersId, string Description) {

string SqlString = "INSERT INTO HistoryComingGoods (CategorysId, GoodsName, BarCode, Quantity, Measurements, Producer, " +

"BuysDate, ExpirationdDate, PurchasePrice, SellingPrice, ProvidersId, Description" +

") Values(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("CategorysId", CategorysId);

cmd.Parameters.AddWithValue("GoodsName", GoodsName);

cmd.Parameters.AddWithValue("BarCode", BarCode);

cmd.Parameters.AddWithValue("Quantity", Quantity);

cmd.Parameters.AddWithValue("Measurements", Measurements);

cmd.Parameters.AddWithValue("Producer", Producer);

cmd.Parameters.AddWithValue("BuysDate", BuysDate.ToString());

cmd.Parameters.AddWithValue("ExpirationdDate", ExpirationdDate.ToString());

cmd.Parameters.AddWithValue("PurchasePrice", PurchasePrice);

cmd.Parameters.AddWithValue("SellingPrice", SellingPrice);

cmd.Parameters.AddWithValue("ProvidersId", ProvidersId);

cmd.Parameters.AddWithValue("Description", Description);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

public HistoryComingGoods SelectedHistoryComingGoodsByComingGoodsId(int HistoryComingGoodsId) {

string SqlString = "SELECT \* " +

"FROM HistoryComingGoods Where HistoryComingGoodsId=" + HistoryComingGoodsId.ToString();

HistoryComingGoods oneHistoryComingGoods = new HistoryComingGoods();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

using (OleDbDataReader reader = cmd.ExecuteReader()) {

while (reader.Read()) {

oneHistoryComingGoods.HistoryComingGoodsId = Convert.ToInt32(reader["HistoryComingGoodsId"]);

oneHistoryComingGoods.CategorysId = Convert.ToInt32(reader["CategorysId"]);

oneHistoryComingGoods.GoodsName = reader["GoodsName"].ToString();

oneHistoryComingGoods.BarCode = reader["BarCode"].ToString();

oneHistoryComingGoods.Quantity = Convert.ToInt32(reader["Quantity"]);

oneHistoryComingGoods.Measurements = reader["Measurements"].ToString();

oneHistoryComingGoods.Producer = reader["Producer"].ToString();

oneHistoryComingGoods.BuysDate = Convert.ToDateTime(reader["BuysDate"]);

oneHistoryComingGoods.ExpirationdDate = Convert.ToDateTime(reader["ExpirationdDate"]);

oneHistoryComingGoods.PurchasePrice = Convert.ToDouble(reader["PurchasePrice"]);

oneHistoryComingGoods.SellingPrice = Convert.ToDouble(reader["SellingPrice"]);

oneHistoryComingGoods.ProvidersId = Convert.ToInt32(reader["ProvidersId"]);

oneHistoryComingGoods.Description = reader["Description"].ToString();

}

}

}

conn.Close();

}

return oneHistoryComingGoods;

}

public List<HistoryComingGoods> GetAllHistoryComingGoods() {

\_ProvidersList = \_ProvidersProvider.GetAllProviders();

\_CategorysList = \_CategorysProvider.GetAllCategorys();

int i = 0;

string SqlString = "SELECT \* FROM HistoryComingGoods";

List<HistoryComingGoods> listHistoryComingGoods = new List<HistoryComingGoods>();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

using (OleDbDataReader reader = cmd.ExecuteReader()) {

while (reader.Read()) {

HistoryComingGoods oneHistoryComingGoods = new HistoryComingGoods();

oneHistoryComingGoods.Number = ++i;

oneHistoryComingGoods.HistoryComingGoodsId = Convert.ToInt32(reader["HistoryComingGoodsId"]);

oneHistoryComingGoods.CategorysId = Convert.ToInt32(reader["CategorysId"]);

oneHistoryComingGoods.GoodsName = reader["GoodsName"].ToString();

oneHistoryComingGoods.BarCode = reader["BarCode"].ToString();

oneHistoryComingGoods.Quantity = Convert.ToInt32(reader["Quantity"]);

oneHistoryComingGoods.Measurements = reader["Measurements"].ToString();

oneHistoryComingGoods.Producer = reader["Producer"].ToString();

oneHistoryComingGoods.BuysDate = Convert.ToDateTime(reader["BuysDate"]);

oneHistoryComingGoods.ExpirationdDate = Convert.ToDateTime(reader["ExpirationdDate"]);

oneHistoryComingGoods.PurchasePrice = Convert.ToDouble(reader["PurchasePrice"]);

oneHistoryComingGoods.SellingPrice = Convert.ToDouble(reader["SellingPrice"]);

oneHistoryComingGoods.ProvidersId = Convert.ToInt32(reader["ProvidersId"]);

oneHistoryComingGoods.Description = reader["Description"].ToString();

listHistoryComingGoods.Add(oneHistoryComingGoods);

}

}

conn.Close();

}

}

if (listHistoryComingGoods.Count == 0) {

HistoryComingGoods noHistoryComingGoods = new HistoryComingGoods();

noHistoryComingGoods.HistoryComingGoodsId = 0;

noHistoryComingGoods.Message = NamesMy.NoDataNames.NoDataInHistoryComingGoods;

listHistoryComingGoods.Add(noHistoryComingGoods);

} else {

for (int j = 0; j < listHistoryComingGoods.Count; j++) {

listHistoryComingGoods[j].ProvidersFIO = GetProvidersFIO(listHistoryComingGoods[j].ProvidersId, \_ProvidersList);

listHistoryComingGoods[j].CategorysName = GetCategorysName(listHistoryComingGoods[j].CategorysId, \_CategorysList);

}

}

return listHistoryComingGoods;

}

private string GetProvidersFIO(int ProvidersId, List<Providers> ProvidersList) {

for (int i = 0; i < ProvidersList.Count; i++) {

if (ProvidersId == ProvidersList[i].ProvidersId) {

return ProvidersList[i].FIO;

}

}

return "";

}

private string GetCategorysName(int CategorysId, List<Categorys> CategorysList) {

for (int i = 0; i < CategorysList.Count; i++) {

if (CategorysId == CategorysList[i].CategorysId) {

return CategorysList[i].CategorysName;

}

}

return "";

}

public void UpdateHistoryComingGoods(int CategorysId, string GoodsName, string BarCode, int Quantity, string Measurements, string Producer,

DateTime BuysDate, DateTime ExpirationdDate, double PurchasePrice, double SellingPrice, int ProvidersId, string Description, int HistoryComingGoodsId) {

string SqlString = "UPDATE HistoryComingGoods SET CategorysId=?, GoodsName=?, BarCode=?, Quantity=?, Measurements=?, Producer=?," +

"BuysDate=?, ExpirationdDate=?, PurchasePrice=?, SellingPrice=?, ProvidersId=?, Description=? " +

"WHERE HistoryComingGoodsId=?";

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("CategorysId", CategorysId);

cmd.Parameters.AddWithValue("GoodsName", GoodsName);

cmd.Parameters.AddWithValue("BarCode", BarCode);

cmd.Parameters.AddWithValue("Quantity", Quantity);

cmd.Parameters.AddWithValue("Measurements", Measurements);

cmd.Parameters.AddWithValue("Producer", Producer);

cmd.Parameters.AddWithValue("BuysDate", BuysDate.ToString());

cmd.Parameters.AddWithValue("ExpirationdDate", ExpirationdDate.ToString());

cmd.Parameters.AddWithValue("PurchasePrice", PurchasePrice);

cmd.Parameters.AddWithValue("SellingPrice", SellingPrice);

cmd.Parameters.AddWithValue("ProvidersId", ProvidersId);

cmd.Parameters.AddWithValue("Description", Description);

cmd.Parameters.AddWithValue("HistoryComingGoodsId", HistoryComingGoodsId);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

public void DeleteHistoryComingGoodsByHistoryComingGoodsId(int HistoryComingGoodsId) {

string SqlString = "DELETE FROM HistoryComingGoods WHERE HistoryComingGoodsId=" + HistoryComingGoodsId.ToString();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

}

}

public class HistoryComingGoods {

private int \_Number;

private int \_HistoryComingGoodsId;

private int \_CategorysId; //Категорія

private string \_GoodsName; //Назва

private string \_BarCode; //Штрих-код

private int \_Quantity; //Кількість

private string \_Measurements; //Од. виміру

private string \_Producer; //Виробник

private DateTime \_BuysDate; //Дата постачання

private DateTime \_ExpirationdDate; //Термін придатності

private double \_PurchasePrice; //Ціна закупівлі

private double \_SellingPrice; //Ціна продажу

private int \_ProvidersId; //Постачальник

private string \_Description; //Опис

private string \_ProvidersFIO;

private string \_CategorysName;

private string \_Message;

public HistoryComingGoods() {

\_Number = 0;

\_HistoryComingGoodsId = 0;

\_CategorysId = 0;

\_GoodsName = String.Empty;

\_BarCode = String.Empty;

\_Quantity = 0;

\_Measurements = String.Empty;

\_Producer = String.Empty;

\_BuysDate = new DateTime();

\_ExpirationdDate = new DateTime();

\_PurchasePrice = 0.0;

\_SellingPrice = 0.0;

\_ProvidersId = 0;

\_Description = String.Empty;

\_ProvidersFIO = String.Empty;

\_CategorysName = String.Empty;

\_Message = String.Empty;

}

public int Number {

set { \_Number = value; }

get { return \_Number; }

}

public int HistoryComingGoodsId {

set { \_HistoryComingGoodsId = value; }

get { return \_HistoryComingGoodsId; }

}

public int CategorysId {

set { \_CategorysId = value; }

get { return \_CategorysId; }

}

public string GoodsName {

set { \_GoodsName = value; }

get { return \_GoodsName; }

}

public string BarCode {

set { \_BarCode = value; }

get { return \_BarCode; }

}

public int Quantity {

set { \_Quantity = value; }

get { return \_Quantity; }

}

public string Measurements {

set { \_Measurements = value; }

get { return \_Measurements; }

}

public string Producer {

set { \_Producer = value; }

get { return \_Producer; }

}

public DateTime BuysDate {

set { \_BuysDate = value; }

get { return \_BuysDate; }

}

public DateTime ExpirationdDate {

set { \_ExpirationdDate = value; }

get { return \_ExpirationdDate; }

}

public double PurchasePrice {

set { \_PurchasePrice = value; }

get { return \_PurchasePrice; }

}

public double SellingPrice {

set { \_SellingPrice = value; }

get { return \_SellingPrice; }

}

public string Description {

set { \_Description = value; }

get { return \_Description; }

}

public string ProvidersFIO {

set { \_ProvidersFIO = value; }

get { return \_ProvidersFIO; }

}

public string CategorysName {

set { \_CategorysName = value; }

get { return \_CategorysName; }

}

public int ProvidersId {

set { \_ProvidersId = value; }

get { return \_ProvidersId; }

}

public string Message {

set { \_Message = value; }

get { return \_Message; }

}

}

Лістинг 7. Код класу «LogsProvider»

using RetailTradeEnterprise.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 RetailTradeEnterprise.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; }

}

}

Лістинг 8. Код класу «ProvidersProvider»

using RetailTradeEnterprise.AppCode;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.OleDb;

namespace RetailTradeEnterprise.Provider {

class ProvidersProvider {

private string \_ConnString = System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

public void InsertProviders(string FirstName, string LastName, string Phone, string Address, string Email) {

string SqlString = "INSERT INTO Providers (FirstName, LastName, Phone, Address, " +

"Email) Values(?, ?, ?, ?, ?)";

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("FirstName", FirstName);

cmd.Parameters.AddWithValue("LastName", LastName);

cmd.Parameters.AddWithValue("Phone", Phone);

cmd.Parameters.AddWithValue("Address", Address);

cmd.Parameters.AddWithValue("Email", Email);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

public List<Providers> GetAllProviders() {

int i = 0;

string SqlString = "SELECT \* FROM Providers";

List<Providers> listAllProviders = new List<Providers>();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

using (OleDbDataReader reader = cmd.ExecuteReader()) {

while (reader.Read()) {

Providers oneProviders = new Providers();

oneProviders.Number = ++i;

oneProviders.ProvidersId = Convert.ToInt32(reader["ProvidersId"].ToString());

oneProviders.FirstName = reader["FirstName"].ToString();

oneProviders.LastName = reader["LastName"].ToString();

oneProviders.FIO = oneProviders.LastName + " " + oneProviders.FirstName;

oneProviders.Phone = reader["Phone"].ToString();

oneProviders.Address = reader["Address"].ToString();

oneProviders.Email = reader["Email"].ToString();

listAllProviders.Add(oneProviders);

}

}

conn.Close();

}

}

if (listAllProviders.Count == 0) {

Providers noDataInProviders = new Providers();

noDataInProviders.ProvidersId = 0;

noDataInProviders.Message = NamesMy.NoDataNames.NoDataInProviders;

listAllProviders.Add(noDataInProviders);

}

return listAllProviders;

}

public Providers SelectedProvidersByProvidersId(int ProvidersId) {

string SqlString = "SELECT \* FROM Providers Where ProvidersId=" + ProvidersId.ToString();

Providers oneProviders = new Providers();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

using (OleDbDataReader reader = cmd.ExecuteReader()) {

while (reader.Read()) {

oneProviders.ProvidersId = Convert.ToInt32(reader["ProvidersId"].ToString());

oneProviders.FirstName = reader["FirstName"].ToString();

oneProviders.LastName = reader["LastName"].ToString();

oneProviders.FIO = oneProviders.LastName + " " + oneProviders.FirstName;

oneProviders.Phone = reader["Phone"].ToString();

oneProviders.Address = reader["Address"].ToString();

oneProviders.Email = reader["Email"].ToString();

}

}

}

conn.Close();

}

return oneProviders;

}

public void UpdateProviders(string FirstName, string LastName, string Phone, string Address, string Email, int ProvidersId) {

string SqlString = "UPDATE Providers SET FirstName=?, LastName=?, Phone=?, Address=?, " +

"Email=? WHERE ProvidersId=?";

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

cmd.CommandType = CommandType.Text;

cmd.Parameters.AddWithValue("FirstName", FirstName);

cmd.Parameters.AddWithValue("LastName", LastName);

cmd.Parameters.AddWithValue("Phone", Phone);

cmd.Parameters.AddWithValue("Address", Address);

cmd.Parameters.AddWithValue("Email", Email);

cmd.Parameters.AddWithValue("ProvidersId", ProvidersId);

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

public void DeleteProvidersByProvidersId(int ProvidersId) {

string SqlString = "DELETE FROM Providers WHERE ProvidersId=" + ProvidersId.ToString();

using (OleDbConnection conn = new OleDbConnection(\_ConnString)) {

using (OleDbCommand cmd = new OleDbCommand(SqlString, conn)) {

conn.Open();

cmd.ExecuteNonQuery();

conn.Close();

}

}

}

}

}