МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської підготовки

Кафедра Інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка алгоритму та програмної системи планування обсягів продажу аптечного закладу»

Виконав студент 2 курсу групи КН- 2 спеціальності 122 Комп'ютерні науки

Ількін Віктор Олегович

Керівник к.г.н., доц. Коваленко Людмила Борисівна

Рецензент д.ф.-м.н., проф. Ковальчук Володимир Володимирович

Одеса 2019

ЗМІСТ

ВСТУП

Як відомо, роздрібна торгівля являє собою кінцеву ланку від виробництва до кінцевого споживача. Це твердження справедливе для будь-якого товару і медикаменти тут не є винятком з правил. На ефективність торгівлі медикаментами впливає безліч факторів, серед яких можна відзначити такі параметри, як цінова політика, широта асортименту, рівень обслуговування, тип самої продукції і т.д. На деякі з цих параметрів вплинути ніяк не можна, проте є багато факторів, які істотно впливає на ефективність роботи аптечної організації і при цьому піддаються впливу. Одним із значних факторів являє собою задоволеність попиту покупців, яке досягається шляхом підтримки необхідного асортименту і ефективного управління запасами медикаментів даного аптечного закладу.

Управління товарними запасами – справжнє мистецтво, якому треба вчитися. Товари в асортименті аптеки поділяються на основні, які мають протягом року постійну модель продажів, і сезонні. Для товарів основного асортименту контроль і планування запасу може ґрунтуватися на замовленні, яке залежить від продажів. Для товарів сезонного асортименту, пов'язаних з великим ризиком і одночасно з великою віддачою, фахівець із закупівель, повинен завчасно планувати або прогнозувати продажу і величину товарного запасу.

Дане питання є досить актуальним як для невеликих одиничних аптек, так і для великих аптечних мереж. З ростом асортименту медикаментозних препаратів і кількості постачальників медикаментів стає все важче управляти медикаментозними запасами, і одного тільки застосування виключно класичних методів управління запасами стає вже недостатньо.

В наші дні статистичні методи прогнозування займають особливе місце у будь-якій економічній практиці. Прогнозування грає важливу роль для виявлення нових напрямків розвитку комерційної організації в умовах перманентного зміни і впливу на неї великого числа факторів як зовнішньої,

так і внутрішнього середовища, а також пошуку шляхів здійснення необхідних і достатніх дій з підтримки необхідного стійкого стану економічного становища даного підприємства. Область застосування подібних методів прогнозування для комерційних організацій досить широка. У поточній економічній ситуації зростає роль прогнозування і генерування подальшої попереджувальної інформації, яка безпосередньо сприяє прийняттю науково-обґрунтованих управлінських рішень. Однією з найважливіших проблем, яку можна і потрібно вирішити за допомогою сучасних інформаційних технологій, є завдання прогнозування збуту продукції (товару).

Актуальність обраної теми обумовлена тим, що в умовах глобальної економіки і постійно зростаючої конкуренції головним фактором, що визначає комерційний успіх організації, є грамотне стратегічне планування необхідних запасів медикаментів в даній організації.

Метою цієї дипломної роботи є розроблення автоматизованої інформаційної системи аптечної організації, а також реалізація функції прогнозування продажів для даної системи.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- розглянути і порівняти існуючі методи прогнозування;
- обґрунтувати вибір методу прогнозування;
- вибрати засоби розробки системи;
- розробити алгоритм побудови прогнозу;
- розробити алгоритм оцінки якості моделі;
- розробити інформаційну систему і реалізувати в ній функцію прогнозування продажів.

Структура магістерської роботи складається з вступу, трьох розділів, висновків, переліку посилань на 20 найменувань, додатків. Повний обсяг роботи становить 75 сторінок, містить 22 рисунки і 11 таблиць.

1 ОГЛЯД МЕТОДІВ ПРОГНОЗУВАННЯ

1.1 Сутність і методи прогнозування

Прогнозування можна розглядати, як важливий фактор вдосконалення системи управління господарюючого суб'єкта в сфері управління, оптимізації його діяльності та підвищення конкурентної стійкості на ринку, що зумовлює прояв великого інтересу до даної проблеми у вітчизняних організаціях, пов'язаних з бізнесом.

Прогнозування – це, перш за все, науково обґрунтована гіпотеза про ймовірний майбутній стан попиту на товари і послуги і характеризують цей стан показники [1][1)].

У ринкових умовах господарювання підприємства змушені займатися прогнозуванням. Зовнішнє середовище є визначальним фактором при складанні планів і прогнозів. Нестабільність і слабка передбачуваність зовнішнього середовища призводить до необхідності розробки методології та інструментарію прогнозування. Важливе значення набуває розвиток теорії і практики прогнозування при недостатній інформації про прогнозовані процесах з урахуванням інтересів різних учасників бізнес-процесів і зацікавлених сторін. Не дивлячись на значну мінливість зовнішнього середовища, є необхідність як довгострокового, так і короткострокового прогнозування. Це відноситься до різних господарюючих суб'єктів.

У перекладі з грецької слово «прогноз» означає передбачення, пророкування про розвиток чого-небудь, засноване на певних фактичних даних. У загальному вигляді під прогнозом слід розуміти науково обґрунтоване судження про можливі стани об'єкта в майбутньому, про альтернативні шляхи і терміни його здійснення.

Процес розробки прогнозів називається прогнозуванням.

[1)] [1] Дуброва Т.А. Статистические методы прогнозирования: Учеб, пособие для вузов. М.: ЮНИТИ-ДАНА, 2003. 206 с.

Прогнозування – це вид пізнавальної діяльності, спрямованої на встановлення майбутніх станів соціально-економічних систем на основі аналізу закономірностей її розвитку в ретроспективі. У проблемі прогнозування розрізняють два аспекти: теоретико-пізнавальний метод, що має на увазі опис можливих чи бажаних перспектив, станів, рішень проблем майбутнього, і управлінський, що передбачає використання інформації про майбутнє при прийнятті рішень. Розрізняють три форми передбачення: гіпотезу, прогноз і план.

Гіпотеза відображає наукове передбачення, здійснюване в рамках загальної теорії. Вихідною базою формування гіпотези є теорія і відкриті на її основі закономірності і причинно-наслідкові зв'язки функціонування і розвитку досліджуваних об'єктів. Гіпотеза здатна відобразити лише якісні характеристики досліджуваних об'єктів, що відображають загальні закономірності їх поведінки.

Прогноз в порівнянні з гіпотезою має більшу ступінь визначеності і достовірності, так як він базується на обліку не тільки якісних, а й певних кількісних параметрів об'єкта.

План являє собою ще більш конкретне, диференційоване і досить точне передбачення, пророкування результатів протікання конкретних подій і процесів.

Мета прогнозування полягає в створенні наукових передумов, що включають науковий аналіз тенденцій розвитку економіки; варіантне передбачення майбутнього розвитку суспільного відтворення, що враховує як сформовані тенденції, так і намічені цілі; оцінку можливих наслідків прийнятих рішень; обґрунтування напрямків соціально-економічного та науково-технічного розвитку для прийняття керуючих рішень.

Прогнозування - одна з основних складових управлінського процесу. Без прогнозування, без уявлення про очікуваний хід розвитку подій неможливо прийняття ефективного управлінського рішення. Існують різні різновиди прогнозів: економічні прогнози, прогнози розвитку технології,

прогнози розвитку конкуренції, прогнози на основі опитувань і досліджень, соціальне прогнозування. Всі типи прогнозів використовують різні методи прогнозування.

1.1.1Метод експертних оцінок

Сутність даного методу в проведенні інтуїтивно-логічного аналізу експертами (від лат. – досвідчений) стоїть проблеми з узгодженням суджень і винесенням результатів. Отримане спільну думку всіх експертів, які можуть бути отримані також на основі бальних оцінок, після їх обробки вважаються рішенням даного завдання [2][1)].

Для прогнозування в області обсягів продажів використовуються три форми:

– інтервального прогнозу;

– точкового прогнозу;

– прогнозу розподілу ймовірностей.

Інтервальний прогноз обсягу продажів полягає у встановленні меж, всередині яких знаходяться прогнозоване значення показника з заданим рівнем значущості.

Точковий прогноз обсягу продажів полягає в прогнозі конкретної цифри. Такий прогноз найбільш простий, так як містить найменше інформації. Існує думка, що такий прогноз може бути заздалегідь помилковим, але розрахунок помилки прогнозу і ймовірності точності прогнозування не передбачений методикою. Через це, частіше застосовуються методи інтервальних і імовірнісних прогнозів.

Прогноз розподілу ймовірностей полягає у виявленні ймовірності попадання фактичного значення показника в одну з груп з встановленим інтервалом груп. Наприклад, «Експерти виявили, що при складанні прогнозу

[1)] [2] Афанасьев, В.Н. Анализ временных рядов и прогнозирование: учебник/В.Н. Афанасьев, М.М. Юзбашев. М.: Финансы и статистика; ИНФРА-М, 2010

є ймовірність, що фактичний обсяг продажів не зможе розташуватися в заданому інтервалі, але при цьому, вони запевняють, що такий інтервал настільки малий, що може не враховуватися при плануванні».

Для знаходження спільної думки у експертів, часто потрібно отримати дані про прогнози від кожного з них, після чого, розрахувати значення прогнозів, користуючись системою зважування індивідуальних значень по якомусь критерію.

Все це змушує підприємця використовувати думки груп фахівців-експертів в різних областях. Це допомагає розглянути все різноманіття чинників і об'єднати знання і різні підходи кожного експерта в одну злагоджену робочу систему, за допомогою якої виникає пропозиція безлічі рішень проблеми і можливість для керівництва вибрати найбільш підходящий варіант для вирішення що стоїть перед ним проблеми.

При виконанні своєї ролі експерти виконають дві основні функції, це формування об'єктів, альтернатив, і вимірюють їх характеристики, тобто ймовірність скоєння тих чи інших подій. Об'єкти формуються шляхом інтуїції і логічного мислення, е експерта. А вимір характеристик вимагає чималих знань теорії вимірювань.

Важливими особливостями методу є рішення складних проблем шляхом наукових обґрунтувань всіх етапів експертизи, що дозволяє забезпечити високу ефективність роботи на будь-якому етапі, а також застосування кількісних методів при оцінці суджень. Така особливість відрізняє метод експертних оцінок від експертизи застосовуваної звичайним шляхом в будь-яких сферах життя.

Метод експертних оцінок застосовується при визначенні інтервалів часу здійснення подій, визначення ймовірності настання певних подій за певний проміжок часу, визначення альтернативи рішень, і можливості вибору з них найкращого, також альтернативу з розподілу ресурсів, альтернативу прийняття рішень в певній ситуації з оцінкою кращого.

Так, під поняттям експертна оцінка розуміється комплекс взаємопов'язаних логічно різних операцій і процедур, з використанням математики та наукового обґрунтування, спрямований на отримання інформації від експертів-фахівців, з метою вироблення та вибору оптимального рішення. Такі результати часто виражені як в кількісному, так і в якісній формі.

Методи експертних оцінок можна розділити на два підкласу [3][1)]:

– прямі методи полягають в отриманні незалежної думки колективу фахівців без впливу на їх думку інших членів комісії або всієї групи в цілому.

– експертні оцінки непрямі, тобто зі зворотним зв'язком містять в собі принцип впливу на оцінку групи або одного з експертів суджень, виявлених раніше від цієї ж групи або одного індивідуума.

Ще можна виділити колективні та індивідуальні експертні оцінки. Індивідуальні - це «інтерв'ю», яке проводиться в бесіді спеціаліста і експерта по схемі «питання - відповідь». Колективні методи – це методи за принципом «комісії», «мозковий штурм», метод «Дельфі». Головна відмінність даного підходу в тому, що в такому методі найбільш висока точність результату, так як при обробці оцінок висока ймовірність, що можуть бути отримані точні результати.

Також основними різновидами методів експертних оцінок, найбільш вживаних відносяться анкетування і інтерв'ювання, дискусії, мозкові штурми, наради і обговорення, сценарії, оперативні гри.

Будь-який з наведених видів має свої плюси і мінуси, в залежності від області їх застосування. Але часто найбільшу ефективність приносить комплексне їх застосування.

Методи сценаріїв і анкетування полягають в індивідуальній роботі фахівця. Але ось інтерв'ювання також може бути виконано і групою експертів. Всі інші методи експертизи мають на увазі колективність, так як

[1)] [3] Бутакова М.М. Экономическое прогнозирование: методы и приемы практических расчетов: учебное пособие/М.М. Бутакова, 2-ое изд., испр. М.: КНОРУС, 2010, 168 с.

незалежно від ступеня участі експертів в роботі кращим варіантом є отримання безлічі думок від різних експертів, що дозволяє отримати свідомо нову інформацію про події, факти і інших явищах з даного питання.

Загалом, прийняття рішень може бути засноване або на основі об'єктивних даних або на основі думки експертів.

Як висновок по результату розгляду методу експертних оцінок можна сказати, що їх застосовують у випадках, коли обґрунтування, оцінка і вибір можливого рішення неможливі тільки в результаті розрахунків. Такі проблеми часто виникають при прогнозуванні, довгостроковому плануванні розробці плану з управління виробництвом. Останнім часом також часто застосовуються в області соціально - політичної, науково-технічної, плануванні і прогнозі в цих сферах, в рішенні проблем управління.

Так як будь-яка сфера діяльності як виробництво, продажу постійно розвиваються, то вимоги до якості таких прогнозів також зростають. Тому рекомендується використовувати різні думки, засновані на судженнях експертів, обізнаних в даній області, які знають на досвіді про можливості розвитку ситуації, але і не забувати про розрахункову складової.

Метод експертних оцінок не заміни планових рішень, а дозволяє доповнити їх необхідною інформацією, думками експертів про те чи іншому питанні. І використання таких методів застосовується тільки там, де немає можливості застосувати розрахункові, точні методи.

Суть проблеми методу експертних оцінок у виборі компетентного фахівця та перевірки справжності його знань, виявлення прихованих причин поставлених оцінок, які занижують достовірність результатів.

1.1.2 Метод аналізу часових рядів

Наступні методи «Аналізу часових рядів», «Казуальний метод» засновані на розрахунку і аналізі кількісних показників, але кожен з цих двох методів помітно різниться.

Для початку слід визначити поняття «Тимчасових рядів» - це послідовність значень, що описує часовий процес, який вимірюється в послідовні моменти часу через рівні проміжки часу, що дозволяє передбачити майбутнє значення на основі минулих спостережень.

Метод аналізу часових рядів полягає в дослідженні взаємозв'язку між двома показниками, це прогноз випадкової компоненти і прогноз детермінованою компоненти [4][1)]. Прогнозування першого прогнозу, заснований на визначенні тенденцій розвитку і екстраполяції, тобто наявність тенденції в сьогоденні, що розповсюджується на майбутнє, але ось прогноз випадкової компоненти виявляється тільки з часткою деякої ймовірності, тому більш складний.

Таке прогнозування полягає у визначенні впливу змін обсягів продажів за минулий період з впливом на майбутній. Зазвичай тимчасові ряди використовуються для розрахунку детерменірованних компонент, таких як трендові, циклічні, випадкові і сезонні показники

Також важливим напрямком в дослідженні закономірності динаміки соціально-економічних процесів є вивчення загальної тенденції розвитку-тренду. Це можна здійснити, застосовуючи метод спеціального аналізу рядів динаміки. Конкретне їх використання залежить від характеру вихідної інформації і зумовлюється завданнями аналізу. Зміна рівнів рядів динаміки обумовлюється впливом на досліджувані явища ряду факторів, які, як правило, неоднорідні по силі, напрямку і часу їх дії.

Постійно діючі фактори роблять на досліджуване явище певний вплив і формують в рядах динаміки основну тенденцію розвитку або «тренд». Вплив інших факторів проявляється періодично. Різні результати дії постійних, періодичних і разових причин і факторів на рівні розвитку соціально-економічних явищ часом зумовлюють необхідність вивчення основних

[1)] [4] Айвазян, С. А. Прикладная статистика и основы эконометрики : учебник для вузов/С. А. Айвазян, В. С. Мхита- рян. М.: ЮНИТИ, 1998.

компонентів рядів динаміки, сезонних коливань, випадкових відхилень, трендів.

Тренд – це компонента яка змінюється та описує вплив довготривалих чинників, ефект від яких проступає поступово. При побудові економіко-математичної моделі прогнозу тренд виявляється основною складовою прогнозованого часового ряду, на яку вже накладаються інші складові, наприклад, сезонні коливання.

Сезонні коливання – це повторювані в кожному часовому періоді коливання, пов'язані зі зміною пори року. Такі зміни безпосередньо можуть бути можуть залежати від коливань інших факторів: політичних, економічних, соціальних, природних.

Якщо протягом року є тільки одне підвищення (зниження) рівня за кілька років, то кажуть про це сезонному циклі; якщо протягом періоду спостерігають кілька мінімумів і максимумів, то статистичну модель сезонної коливання вибирають відповідно до отриманого циклічного процесу.

Відзначимо, що часовий ряд не завжди містить сезонну (циклічну) складову. Перевірку на наявність або відсутність сезонних коливань проводять за допомогою будь-якого критерію або візуально при побудові графіка.

При підтвердженні наявності сезонного процесу виділяється сезонна складова. Процедури розрахунку значень сезонної компоненти залежать від прийнятої моделі часового ряду, що містить сезонність в адитивної або мультиплікативної формі.

Адитивну модель $y_t = u_t + \nu_t + \varepsilon_t$ застосовують в тому випадку, коли амплітуда сезонних коливань з часом не змінюється. Якщо відбуваються істотні сезонні зміни, то використовують мультиплікативну модель $y_t = u_t \cdot \nu_t \cdot \varepsilon_t$.

Часовий ряд, в якому спостерігаються і тренд, і сезонні коливання, будемо називати тренд-сезонним часовим рядом. Статистичні процедури оцінювання сезонності і коригування часових рядів становлять великий

інтерес в сфері економіки, причому оцінки сезонності є лише проміжний етап дослідження. Надалі вони використовуються для моделювання тренд-сезонних процесів. Для підвищення точності прогнозу необхідні дані за тривалий проміжок часу: в разі прогнозування обсягу продажів по місяцях - мінімум за два-три роки, причому кожному місяцю присвоюється номер періоду по порядку [5][1)].

Методика побудови прогнозування за допомогою тренд-сезонних моделей включає в себе наступні основні етапи:

1) Побудова графіка по вихідним даним. Побудова ліній тренду для отриманої кривої. Вибір найбільш точного тренду для прогнозної моделі. Даному етапу надається особливе значення, так як від вибору тренда залежить точність моделі і її достовірність.

2) Лінія тренду – це графічне представлення загальної закономірності зміни ряду даних. Вона може бути додана для будь-якого ряду даних на діаграмі з областями, лінійчатої діаграмі, гістограмі, графіку або точкової діаграмі. Лінія тренду дозволяє прогнозувати зміну ряду. Чим точніше вона буде обрана, тим точніше буде прогноз. Виділяють наступні види тренду:

– лінійний;

– логарифмічний;

– поліноміальний;

– статечної;

– експонентний.

3) Розрахунок значень тренду за фактичний період. Для того щоб розрахувати значення тренду, необхідно знайти його рівняння. Підставивши в рівняння номер періоду, виходить значення тренду.

4) Розрахунок помилки моделі і її точності.

5) Розрахунок прогнозних значень моделі з урахуванням коливань попиту. Отримані значення тренду коригуються на величину помилки і

[1)] [5] Кондратьев, Н. Д. Проблемы экономической динамики/ред. коллегия: Л. И. Абалкин [и др.]. М. : Экономика, 1989.

коливання попиту. Результат розрахунку - скориговані значення моделі, що дозволяють прогнозувати обсяги продажів.

6) Побудова довірчого інтервалу. Довірчий інтервал - це допустиме відхилення спостережуваних значень від справжніх. Він залежить від величини отриманої помилки і значень прогнозної моделі. Верхня межа інтервалу показує максимально можливе значення шуканого, нижня – мінімальне.

Вибір лінії тренду залежить від величини достовірності апроксимації. Найбільшою величиною мають поліноміальний і лінійний тренди.

Порядок виконання даних етапів і їх склад може змінюватися в залежності від постановки завдань, методів вирішення, використовуваних програмних засобів.

1.1.3 Казуальні методи

Наступним розглянутої групою методів є казуальне методи, або як їх називають причинно-наслідкові [6][1)].

Головним завданням таких методів є спроба виявити фактори поведінки прогнозованих показників. Часто, способом знайти такі фактори служить побудова економіко-математичних моделей і моделювання, тобто побудова моделі поведінки досліджуваного об'єкта, з урахуванням взаємозв'язку розвитків явищ і процесів. Але застосування такого методу ставить перед собою складне завдання вибору факторів, які не можуть бути визначені чисто статистичним шляхом, але вимагає вивчення самого процесу. У чому і полягає більший плюс економічного аналізу, на відміну від статистичного.

Розглянуті в даному розділі, останні групи методів, полягають, в основному, в використанні та розробці прогнозної моделі, де зміна в обсязі

[1)] [6] Лагутин, М. В. Наглядная математическая статистика : учеб, пособие / М. В. Лагутин. М. : БИНОМ ; Лаборатория знаний, 2009.

продажів є результат зміни декількох змінних. Такі методи вимагають для своєї реалізації визначення ознак і оцінки змін, установки між ними певних залежностей, що досить непросто.

Найбільш ефективними для прогнозування обсягів продажів є такі методи казуальної групи:

– метод провідних індикаторів;

– кореляційно-регресійний аналіз;

– метод обстеження намірів споживача.

Метод провідних індикаторів полягає в загальному прогнозуванні змін в певній сфері роботи схожих організацій, ситуації на ринку, ніж окремої з них. Тому даний метод дозволяє врахувати загальний рівень ринку для подальшого його обліку в прогнозі конкретної організації, що для деяких фірм є корисним фактором.

Сам термін «Провідні індикатори» можна визначити, як показники або тимчасові ряди, зміни яких відбувається в одному напрямку з досліджуваним показником, тільки випереджаючому його в часі. Наприклад, зростання добробуту випереджає зростання попиту. Так, вивчаючи показник і зміна добробуту можна припустити про можливу зміну попиту [7][1)].

Для того, щоб оцінити ринковий потенціал будь-якої зони, нерідко застосовують індикатори купівельної спроможності, з метою визначити привабливість даного ринку за середньозваженим значенням складових потенціал ринкової території. Це можуть бути показники кількості споживаних одиниць продукції, купівельної спроможності і готовності до витрат. Всі ці індикатори визначаються для потрібної території, і після відбувається виявлення і розрахунок середньозваженого індексу.

За допомогою обстежень намірів споживача можна отримати масу обґрунтувань отриманого прогнозу обсягів продажів. Адже, те, що хоче придбати споживач і коли він це зробить, знає тільки сам споживач, тому

[1)] [7] Лапыгин, Ю. Н. Экономическое прогнозирование : учеб, пособие / Ю. Н. Лапыгин. М.: Эксмо, 2009.

багато організацій вкладають свої кошти в вивчення думок споживачів про свій продукт. Найбільш популярні такі дослідження у організацій, які виробляють або продають великі товари і послуги, для покупки яких необхідно довгі міркування і планування.

Однак, істотний нездоланний мінус таких досліджень в тому, що думка покупця про товар часто піддається зміні під впливом зовнішніх факторів, і для його підтримки необхідно проводити процедури поліпшення продукції, акції та всілякого підтримки лояльності свого споживача, щоб якомога точніше отримати подальші прогнозні значення, які не суперечать з реальністю.

Так як для складання прогнозу можуть використовуватися і застосовуватися всі вищерозглянуті методи, то потрібно вирішити який саме метод найкраще використовувати в тій чи іншій ситуації. Це рішення може залежати від умов необхідної точності прогнозу, наявності даних, часу.

Якщо не потрібно проводити прогноз розрахункових даних, в якому частіше застосовуються методи часових рядів, то дуже вдалим для такого рішення буде застосування методу експертних оцінок або казуальних. Така потреба з'являється в терміновості отримання даних, при цьому не варто нехтувати часом, щоб ця терміновість не позначилася на результатах.

Якість зробленого прогнозу визначає спеціальний коефіцієнт, як відношення числа підтверджених прогнозів до числа загальних прогнозів. Такий розрахунок рекомендується проводити не в кінці прогнозних термінів, а при складанні прогнозу. Тому використовує такий метод як інверсна верифікація, суть якого полягає в тому, що правильність прогнозної моделі перевіряється її здатністю відтворювати фактичні дані в минулому.

До одним з найбільш популярних методів відноситься метод кореляційно-регресійного аналізу. Такий метод складається з двох компонент. Це кореляційний і регресійний аналіз. Кореляційний аналіз - це метод кількісно визначає напрямок і тісноту взаємозв'язку між досліджуваними об'єктами. Регресійний аналіз – це метод кількісно визначає

вид математичної функції, причинно-наслідкові залежності між досліджуваними об'єктами. Для їх оцінки застосовується шкала Чеддока, яка розділяє отримані результати на: слабкі – 0,1-0,3, помірні – 0,3-0,5, помітні – 0,5-0,7, високі–0,7-0 , 9 і сильні – 0,9-1,0.

Для прогнозування обсягів продажів може бути застосована регресійна модель, де розглядається такі факторні ознаки як дохід, ціна конкуренти, рівняння множинної регресії.

Такий метод регресійній моделі знаходить своє застосування в прогнозуванні попиту на товари споживчого призначення і виробничі кошти [8][1)].

1.2 Порівняльна характеристика методів прогнозування

Зробимо порівняльний аналіз методів між собою, шляхом складання короткої таблиці, яка характеризує розглянуті методи прогнозування (табл. 1).

Таблиця 1 – Порівняльна характеристика методів прогнозування

Характеристики	Метод експертних оцінок	Казуальний метод	Метод аналізу часових рядів
Термін	Довгостроковий,	Довгостроковий, середньостроко-вий	Коротко-строковий, середньо-строковий
Вимоги для реалізації методу	Експерти, кваліфіковані фахівці	Кількісні показники	Кількісні показники
Швидкість	+++	+++	+++
Складність методу	Висока	Висока	Середня

[1)] [8] Кошечкин С.А. Алгоритм прогнозирования объема продаж в MS Excol / С.А. Кошечкин // Маркетинг за рубежом. 2001. No 5. С. 34-42

Продовження табл.1

Характеристики	Метод експертних оцінок	Казуальний метод	Метод аналізу часових рядів
Достовірність і надійність	Залежить від компетентності експерта	Залежить від передбачення	Залежить від якості і кількості даних
Переваги	Системний підхід		Простота, наочність результатів, проста реалізація на ЕОМ
Недоліки	Достовірність і надійність результатів дослідження залежать від компетентності експерта, трудомісткість збору інформації, потреба у висококваліфікованих кадрах	Практичне застосування методу для деяких напрямків може виявитися помилковим	Прогнозування тільки кількісних показників, наявність даних за тривалий період часу, обмежений період прогнозування

Як ми бачимо з табл. 1, всі три розглянутих груп методів різноманітні за ступенем складності, періоду складання прогнозу і методу.

У першому випадку використовуються тільки якісні дані, але отриманий результат стоїть під загрозою достовірності через складність отримання даних опитування і необхідність наявності висококваліфікованих фахівців.

Другий метод дозволяє виявити причинно-наслідковий зв'язок з використанням кількісних даних, але не завжди може бути точним для деяких областей життя, можливість помилки узагальнення, необхідність глибинного аналізу.

Третій метод найбільш поширений і досить нескладний, також дозволяє за допомогою набору даних і ЕОМ зробити прогноз на, максимум, середньостроковий період, що і є його недоліком, але з огляду на постійно змінні фактори зовнішнього середовища і можливі нововведення в організації, розрахунок на більш тривалий термін не має сенсу, тому даний метод несе в собі переваги.

1.3 Обґрунтування вибору методу прогнозування

Провівши детальний аналіз кожного методу і з огляду на специфіку діяльності аптечних організацій, в даному дипломному проекте як метод прогнозування, був обраний метод аналізу часових рядів, з прорахунком тренду і сезонності. Даний метод використовується при складанні прогнозу на короткостроковий і середньостроковий період, що дозволяє отримувати достатню точність прогнозних значень.

Також, даний метод не тільки є найбільш популярним методом з усіх перерахованих, але і досить простим і ефективним. Його досить просто реалізувати за допомогою ЕОМ.

Використання даного методу дозволить враховувати вплив фактору сезонності та тренду, виявляти місяці найбільшою і найменшою активності в продажах і в зв'язку з цим правильно планувати запаси.

Грунтуючись на попередньому аналізі часових рядів, що відображають динаміку продажів в аптечній організації, в якості моделі тренду були обрані лінійна і поліноміальна функції.

Ці функції найбільш прості і часто використовуються в прогнозуванні обсягу продажів. Члени цих функцій можуть бути легко розраховані за допомогою методу найменших квадратів.

1.4 Вибір засобів розробки

Проектування концептуальної моделі предметної області доцільно проводити за допомогою спеціального засобу проектування Power Designer.

PowerDesigner - це інструмент для створення моделей бізнес-процесів, а також концептуальних, логічних і фізичних моделей даних для проектування баз даних, включаючи реляційні і розмірні моделі. PowerDesigner може координувати модель бізнес-процесу з дизайном бази даних, забезпечуючи, щоб кроки процесу, які створюють дані, мали уявлення даних в логічної моделі. PowerDesigner може створювати фактичну базу даних з фізичної моделі і створювати різні фізичні реалізації з однієї логічної моделі. PowerDesigner також може перетворити існуючі бази даних в діаграму моделі. Програма працює з багатьма системами управління базами даних (СКБД). Основні вихідні дані інструменту включають діаграми взаємозв'язку сутностей, звіти з аналізу впливу на зміни проекту, а також стандартні або призначені для користувача звіти по всіх об'єктах в проекті (таблиці, поля, відносини).

Його можна встановити за допомогою майстра установки, а необхідні оновлення виявляються і завантажуються монтажником автоматично. Складність установки програмного забезпечення зводиться до мінімуму завдяки автоматичній установці оновлень. Інші компоненти, такі як аналітичні служби і служби баз даних, можуть бути встановлені окремо згодом. Автоматичне оновлення також значно знижує витрати на обслуговування.

В якості мови програмування для розробки автоматизованої інформаційної системи була вибрана мова програмування високого рівня C#

(вимовляється як "сі Шарп"). Вибір на користь даного мови програмування можна обґрунтувати тим, що на автоматизованих робочих місцях користувачів в якості стандартної операційної системи встановлюється операційна Windows версії 7 і вище. Як компонент за замовчуванням, до складу даної операційної системи входить програмна платформа NetFramework, в результаті чого немає необхідності встановлювати додаткові компоненти.

Найбільш популярною мовою розробки програм на платформі NetFramework є мова програмування C#.

C# – простий, сучасний об'єктно-орієнтована і типобезопасна мова програмування. C# є об'єктно-орієнтованою мовою, але підтримує також і компонентно-орієнтоване програмування. Розробка сучасних додатків все більше тяжіє до створення програмних компонентів у формі автономних пакетів, що реалізують окремі функціональні можливості. Важлива особливість таких компонентів – це модель програмування на основі властивостей, методів і подій. Кожен компонент має атрибути, які надають декларативні відомості про компоненті, а також вбудовані елементи документації. C# надає мовні конструкції, безпосередньо підтримують таку концепцію роботи. Завдяки цьому C# відмінно підходить для створення і застосування програмних компонентів [9][1)].

Мова C# має в своєму арсеналі такі функції, що забезпечують надійність і стійкість додатків:

– прибирання сміття автоматично звільняє пам'ять, зайняту знищеними і невикористовуваними об'єктами;

– обробка виключень надає структурований і розширюваний спосіб виявляти і обробляти помилки;

[1)] [9] Троелсен, Эндрю. Язык программирования C# 5.0 и платформа .NET 4.5, 6–е изд. : Пер. с англ. М. : ООО “И.Д. Вильямс”, 2013. 1312 с., ил.

– сувора типізація мови не дозволяє звертатися до неініціалізованих змінним, виходити за межі індексованих масивів або виконувати неконтрольоване приведення типів.

У C# існує єдина система типів. Всі типи C #, включаючи типи-примітиви, такі як int і double, успадковують від одного кореневого типу object. Таким чином, всі типи використовують загальний набір операцій, і значення будь-якого типу можна зберігати, передавати і обробляти схожим чином. Крім того, C # підтримує призначені для користувача посилальні типи і типи значень, дозволяючи як динамічно виділяти пам'ять для об'єктів, так і зберігати спрощені структури в стеці [10][1)].

Під платформу .NET мовою C # на даний момент існує кілька середовищ розробки:

– microsoft Visual Studio;

– monoDevelop;

– sharpDevelop;

В якості інтегрованої середовища розробки було прийнято рішення використовувати Microsoft Visual Studio, як найбільш функціональну з наведених, яка надає потужний інструментарій по створенню широкого спектра програмного забезпечення. Дане середовище програмування являє собою функціонально насичену і найбільш пристосовану під використання в організаціях інтегроване середовище розробки. Одним з переваг Visual Studio є інтегрована довідкова система. Середовище розробки дає можливість отримувати доступ до документації безпосередньо з неї, за допомогою гарячої клавіші F1. Інтегрований відладчик дозволяє розробнику створювати потрібні точки зупину і відстежувати значення потрібних змінних.

[1)] [10] Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. СПб.: Питер, 2013. 896 с.: ил. (Серия «Мастер-класс»).

У редактор коду інтегровано засіб IntelliSense [11][1)]. В системі Visual Studio назву IntelliSense ставиться до великої кількості функцій, починаючи з візуальної реакції на неправильний код і інтелектуальних дескрипторів для проектування форм до комбінацій клавіш, натискання яких призводить до вставки цілих фрагментів коду. Ці властивості, разом узяті, забезпечують розробнику більш глибоке розуміння того, що відбувається, більш високу ефективність його роботи і дозволяють йому контролювати свою програму.

Для розробки інформаційної системи було прийнято рішення використовувати безкоштовну версію Express, можливостей якої в повній мірі достатньо для виконання поставленого завдання.

Для вибору системи управління базою даних (СКБД), яка буде використовуватися в інформаційній системі, розглянемо основні програмні продукти, які існують в даний момент часу на ринку програмного забезпечення. Порівняльні дані представлені в таблиці 2.

Таблиця 2 – Переваги та область застосування сучасних СУБД

Назва	Переваги	Галузь застосування
Microsoft SQL Server	1. Масштабованість і надійність; 2. Можливість одночасної роботи декількох користувачів з даними; 3. Забезпечення захисту даних у разі збою; 4. Можливість роботи з великими обсягами даних; 5. Можливість аналітичної обробки інформації, що зберігається.	Використовується для роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства

[1)] [11] IntelliSense в Visual Studio. URL: https://docs.microsoft.com/en-us/visualstudio/ide/using-intellisense?view =vs-2019. (Дата звернення 18.09.2019).

Продовження табл.2

Назва	Переваги	Галузь застосування
Microsoft Access	1. Легкість в освоєнні; 2. Потужні засоби підготовки звітів; 3. Швидке і легке створення баз даних (присутній конструктор, немає необхідності використовувати при створенні БД запити); 4. Використання засобів автоматизації і додавання складних виразів без написання коду; 5. Невимогливий до ресурсів комп'ютера.	Створення звітів довільної форми на підставі різних даних. Розробка некомерційних додатків.
Oracle Database	1. Безліч підтримуваних платформ; 2. Швидкість роботи; 3. Легкість в управлінні; 4. Масштабованість; 5. Високий рівень захисту даних (як від апаратних збоїв, так і від атак ззовні); 6. Дозволяє оперувати величезними обсягами інформації.	Бази даних з великим об'ємом інформації.

СУБД Oracle використовується, як правило, в основному для створення великих високонавантажених проектів, є однією з найбільш потужних сучасних СУБД, призначених для реалізації баз даних рівня корпорації, що пред'являє серйозні вимоги до сервера. Вибір даної системи в якості використовуваної в розроблюваної системі недоцільний.

Використання СУБД Access також недоцільно, в силу того, що ця СУБД є файл-серверною. Дана СУБД має в своїй основі файловий принцип організації даних, при її використанні в проекті можуть виникнути помилки, які виникають при використанні загального ресурсу.

Використання Microsoft SQL Server найдоцільніше в даному випадку, оскільки ця СУБД є серверної і повністю масштабується. Ця СУБД перш за все виділяється високою надійністю. Це досягається за рахунок застосування різних базових технологій, таких як створення відмово стійких кластерів, віддзеркалення, надання різноманітних засобів для роботи з журналами.

Для розробки інформаційної системи було прийнято рішення використовувати безкоштовну версію Microsoft SQL Server версії Express, якої в повній мірі достатньо для виконання поставленого завдання.

Основні переваги даної системи:

- використовує управління на основі політик для виявлення невідповідних політик безпеки. Ця функція дозволяє тільки авторизованому персоналу доступ до бази даних. Аудит безпеки і події можуть бути автоматично записані в файли журналу.

- має вбудовану прозору процедуру стиснення даних поряд з шифруванням. Користувачам не потрібно змінювати програмну реалізацію для шифрування даних. Сервер MS SQL має контроль доступу в поєднанні з ефективними інструментами управління дозволами. Крім того, він забезпечує підвищену продуктивність при зборі даних.

- сервер включає в себе ефективні інструменти управління та інтелектуального аналізу даних, а також розбиття диска. Оптимальне обслуговування сервера може бути забезпечено при дотриманні ефективних методів управління даними. Ці методи також забезпечують доступність і можливість відновлення даних.

SQL Server Express – це безкоштовна версія основної системи керування базами даних Microsoft – SQL Server. По суті, SQL Server – це система управління базами даних, яку можна використовувати для зберігання і доступу до інформації, що зберігається в багатьох різних базах даних. SQL Server має вражаючий набором функцій, таких як бізнес-аналітика, звітність і поглиблена аналітика.

SQL Server Express – це найпростіше з доступних пропозицій. Це повний механізм бази даних, який ви можете розгорнути на сервері або вбудувати в додаток. Express безкоштовний і поставляється з багатьма з тих же функцій, що і корпоративна версія. SQL Server Express, ймовірно, найбільш підходить для підтримки виробничих програм для підприємств малого та середнього бізнесу. Типовий варіант використання – SQL Server Express – це розгортання розробниками, які не хочуть створювати додатки з базою даних, розміщеної на сервері. Використовуючи Express, вони зможуть розробляти програми зі своєю базою даних SQL Server.

Величезна перевага SQL Server Express полягає в тому, що він безкоштовний. Єдині витрати – це витрати часу на завантаження і налаштування системи. SQL Server Express є ідеальної відправною точкою для невеликих незалежних постачальників програмного забезпечення, оскільки його можна використовувати з будь-яким невеликим додатком. Ліцензування дозволяє включати Express як частину програми або продукту. Хоча існують обмеження на використання пам'яті і сокетів, вони не так обмежувальні, як деякі можуть подумати. Експрес не обмежується одним користувачем, що є поширеною помилкою. Існує обмеження бази даних 10 ГБ, але це максимальний розмір для кожної бази даних, що означає, що ви можете мати кілька баз даних, які зберігають до 10 ГБ даних. Якщо ви є незалежним розробником ПЗ, і ваша компанія відчуває високі темпи зростання, що призводить до збільшення вимог до бази даних, ви можете перейти тільки на платну версію SQL Server.

В SQL Server Express є можливість безкоштовного резервного копіювання в онлайн-сховище, яке допоможе захистити ваші цінні бізнес-дані, якщо щось піде не так. Адміністратори як і раніше повинні дотримуватись рекомендацій з безпеки, таким як обмеження доступу до папок резервного копіювання та дотримання політик паролів Windows.

Хоча Express є «полегшеною» версією SQL Server, все ще існує вражаючий набір функцій, за які доведеться платити з іншими системами.

Express підтримує повнотекстовий пошук, власний XML і середу виконання спільної мови SQL. Інші ключові функції включають в себе компонент звітності і конструктор звітів, що дозволяє створювати призначені для користувача звіти.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Коротка характеристика архітектури системи

Розроблена інформаційна система має архітектуру клієнт-серверної системи, яка представлена на рис. 2.1.

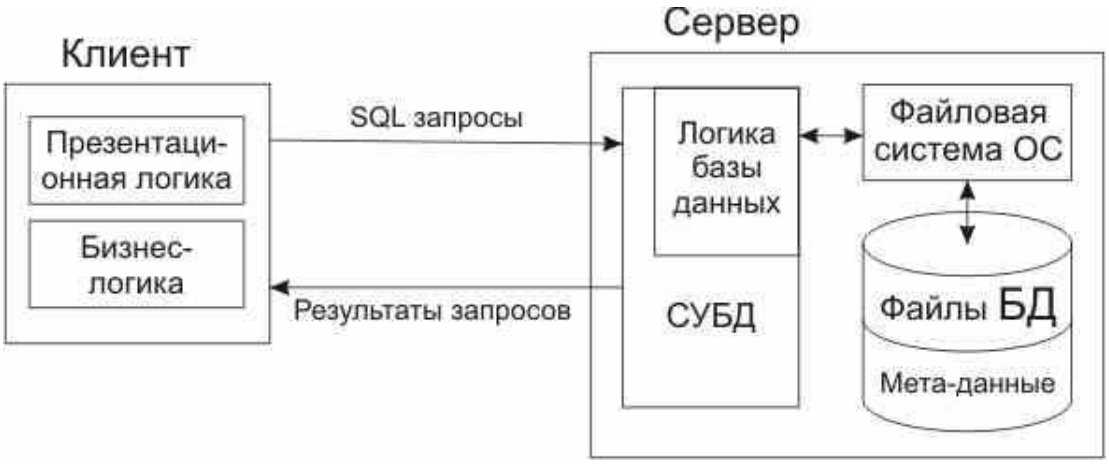


Рисунок 2.1 – Архітектура клієнт-сервер

Дана архітектура побудована на використанні так званого сервера бази даних, що має на увазі управління інформаційними ресурсами бази даних, виконання інформаційних запитів, розмежування доступу до даних і їх захист покладені на систему управлінням базою даних (СКБД). У такій системі будь-який доступ до даних можливий тільки через спеціальні запити до сервера БД через СУБД, яка вже сама взаємодіє з операційною системою сервера для отримання інформації. Безпосередній доступ до файлів даних засобами операційної системи, минаючи СУБД, блокується.

На програмне забезпечення клієнтської сторони (робочих станцій) покладаються в цьому випадку функції з реалізації обробки та інтерпретації даних відповідно до алгоритмів бізнес-логіки вирішення конкретних користувальницьких задач і все інтерфейсні функції введення і відображення інформації (презентаційна логіка).

Взаємодія клієнтської і серверної сторін здійснюється в цьому випадку шляхом формування клієнтом і передачі по мережі сервера бази даних

відповідних інформаційних запитів (в сучасних системах на мові SQL). Ці запити інтерпретуються і виконуються програмним забезпеченням сервера бази даних (СКБД), який потім, якщо це необхідно, повертає клієнту по мережі результат виконання запиту, тобто тільки корисну, затребувану інформацію.

Представлена архітектура дозволяє забезпечити ефективне використання мережевих комунікацій при великій кількості одночасно працюючих із загальним сервером бази даних клієнтських робочих станцій, дає можливість використання одночасної паралельної обробки запитів до бази даних, тим самим підвищуючи продуктивність системи. Ця архітектура на якісно іншому рівні вирішує питання управління доступу користувачів до даних, проблеми захисту даних як від несанкціонованого доступу, так і від можливих апаратно-програмних збоїв системи, дозволяє реалізувати в сучасних СУБД потужний комплекс декларативних засобів контролю і забезпечення цілісності даних.

Конфігурації апаратних засобів комп'ютерів, що використовуються в якості сервера бази даних і робочих станцій, їх системне програмне забезпечення може бути оптимізовано з урахуванням виконуваних ними специфічних функцій.

Говорячи про оптимізацію розподілу функцій між серверною і клієнтськими компонентами інформаційної системи з базами даних, слід зауважити, що наведений вище варіант не є єдиним. Зокрема, сучасні СУБД дозволяють перекладати на серверну сторону і частина бізнес-функцій прикладних задач. Для цього в СУБД реалізовані механізми так званих збережених процедур і процедур обробки подій (тригери).

2.2 Проектування алгоритму побудови прогнозу і розрахунок помилки

Для вирішення поставленого завдання скористаємося тренд-сезонною адитивною моделлю прогнозування, по якій аналізовані рівні часового ряду

можуть бути представлені у вигляді суми трендової, сезонної і випадкової складової [12]:

$$Y_t = T_t + S_t + E_t$$

де, Y_t – спостережуване значення показника продажів в момент часу t;

T_t – значення тренду в момент часу t;

S_t – значення сезонної складової в момент t;

E_t – значення випадкової (нерегулярної) складової в момент t.

Етапи побудови прогнозу наступні.

1. Визначення рівняння тренду.

В якості рівнянь тренду були обрані лінійна (1) і поліноміальна функція (2):

$$T_{\text{л}} = a_0 t + a_1 \quad (1)$$

$$T_{\text{п}} = a_0 t^2 + a_1 t + a_2 \quad (2)$$

де, a_0, a_1, a_2 – коефіцієнти функцій;

T – значення тренду в період часу t.

2. Визначення величини сезонної компоненти

Знаходимо оцінки сезонної компоненти як різницю між фактичними рівнями ряду і значеннями, отриманими за допомогою рівнянь тренду [12]:

$$S_t + E_t = Y_t - T_t$$

Використовуємо оцінки сезонної компоненти $(S_t + E_t)$ для розрахунку значень скоригованої сезонної компоненти S_t на період прогнозування (12 місяців). Для цього знайдемо середні значення за кожен інтервал оцінки сезонної компоненти S_t. У моделей з сезонною компонентою зазвичай передбачається, що сезонні впливи за період взаємо погашаються. У

адитивної моделі це виражається в тому, що сума значень сезонної компоненти за всіма інтервалах повинна бути дорівнює нулю. Тому знайшовши значення випадкової складової, поділивши суму середніх оцінок сезонної складової на 12, ми віднімаємо її значення з кожної середньої оцінки і отримуємо скориговану сезонну компоненту S_t.

3. Визначення помилок моделі.

За методикою побудови адитивної моделі розрахунок абсолютної помилки для кожного періоду здійснюється за формулою [12]:

$$E_t = Y_t - (T_t + S_t)$$

Середнім показником точності прогнозу є середня абсолютна помилка прогнозу $\bar{E}$, яка визначається як середня арифметична проста з абсолютних помилок прогнозу за формулою виду [13]:

$$\bar{E} = \frac{1}{n}\sum_{t=1}^{n}\frac{E_t}{Y_t}$$

де n - довжина часового ряду.

Середня абсолютна помилка прогнозу показує узагальнену характеристику ступеня відхилення фактичних і прогнозних значень ознаки і має ту ж розмірність, що і розмірність досліджуваного ознаки.

Недоліком середньої абсолютної помилки прогнозу є істотна залежність від масштабу виміру рівнів досліджуваних економічних явищ. Тому на практиці, як характеристики точності прогнозу, визначають середньоквадратичну помилку апроксимації, яка виражається у відсотках щодо фактичних значень ознаки. Іншими словами, це середньоквадратичне відхилення розрахункових значень від фактичних, яке обчислюється за формулою [13]:

$$\bar{E}^* = \frac{1}{n}\sum_{t=1}^{n}\left(\frac{E_t}{Y_t}\right)^2 * 100\%$$

Даний показник є відносним показником точності прогнозу і не відображає розмірність досліджуваних ознак, виражається у відсотках і на практиці використовується для порівняння точності прогнозів отриманих як за різними моделями, так і по різних об'єктах. При значенні $\bar{E}^* < 10$ точність прогнозу вважається високою.

4. Визначення точності всієї моделі.

Точність прогнозування є поняття прямо протилежне помилку прогнозування. Якщо помилка прогнозування велика, то точність мала і навпаки, якщо помилка прогнозування мала, то точність велика. По суті справи оцінка помилки прогнозу є зворотна величина для точності прогнозування. Визначити точність всієї моделі можна за допомогою виразу $100\% - \bar{E}^*$.

5. Визначення довірчого інтервалу можливих відхилень.

Для побудови довірчого інтервалу скористаємося даними середньоквадратичної помилки для моделі.Формула довірчого інтервал має вигляд [12]:

$$(F_t \cdot (1 - \bar{E}^*); F_t \cdot (1 + \bar{E}^*))$$

де F_t – прогнозоване значення моделі в момент часу t.

Довірчий інтервал - це допустиме відхилення спостережуваних значень від справжніх. Він залежить від величини отриманої помилки і значень прогнозної моделі. Верхня межа інтервалу показує максимально можливе значення шуканого, нижня - мінімальне.

6. Побудова прогнозу.

Проводиться розрахунок прогнозних значень по тренду. Отримані значення коригуються на величину сезонної компоненти, і з урахуванням помилки моделі знаходиться довірчий інтервал.

2.3 Інфологіческое проектування

2.2.1 Концептуальна модель даних

Концептуальне моделювання - це процес аналізу інформаційних потреб кінцевих користувачів системи [14]. Модель «сутність-зв'язок» наочно і точно відображає уявлення автора про вимоги до даних. Тому вона є хорошим джерелом інформації для проектувальника логічної моделі даних.

На підставі аналізу предметної області була спроектована концептуальна модель даних (ER-діаграма), представлена на рисунку 2.7.

Концептуальна модель описує структуру досліджуваної предметної області. Вона покликана виявити логіко-семантичні зв'язки між даними. З іншого боку, діаграми виразні, наочні і легко інтерпретуються кінцевими користувачами. Логіко-семантичні зв'язки потрібні для визначення обмежень цілісності майбутньої бази даних. Концептуальна модель служить джерелом інформації для фази логічного моделювання.

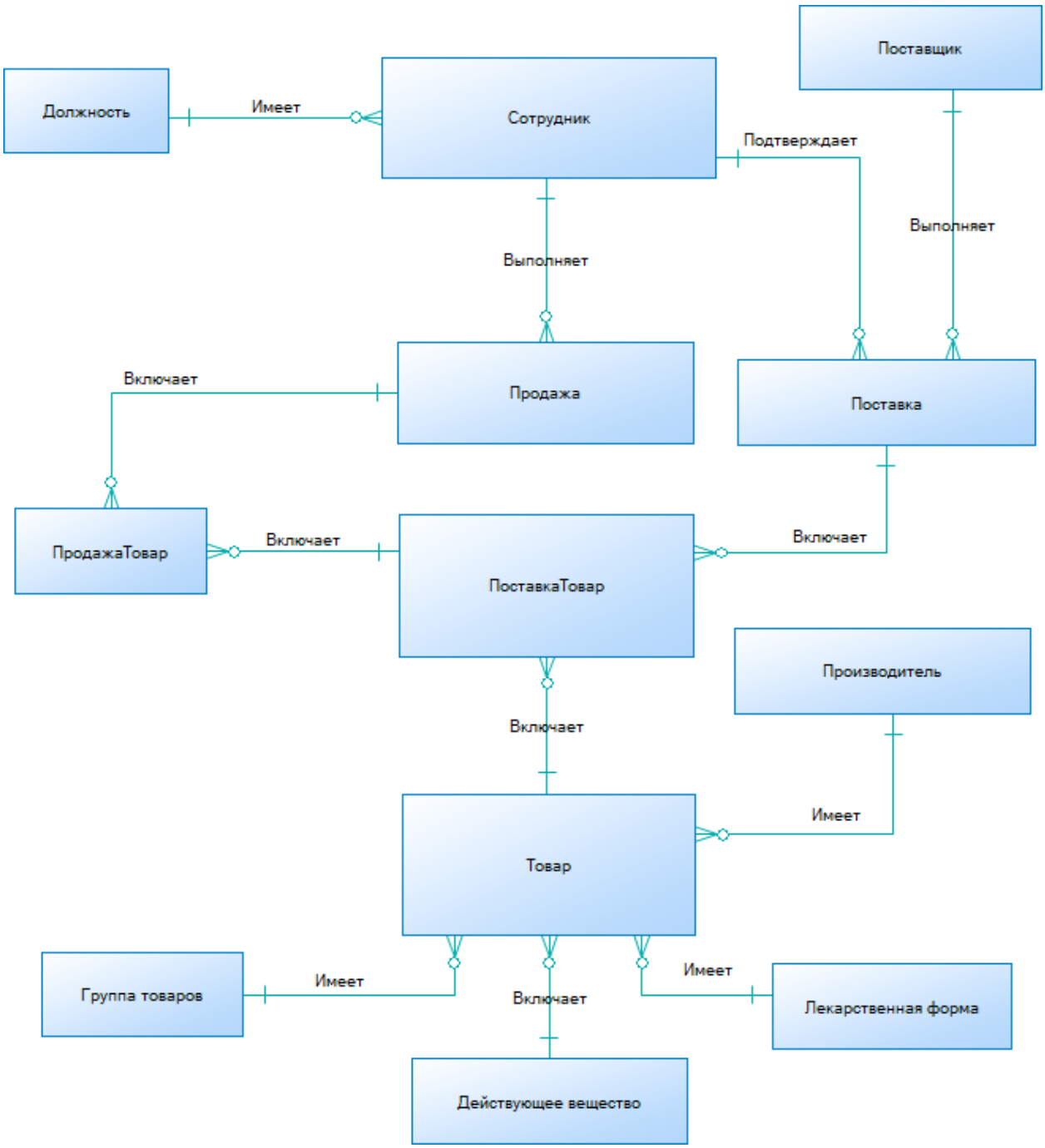


Рисунок 2.7 – Концептуальна модель даних предметної області ER-рівня

2.2.2 Логічна модель даних

В процесі логічного моделювання концептуальна модель уточняється і перетворюється в логічну з урахуванням базової моделі даних цільової СУБД. Логічна модель даних представлена на рисунку 2.8. Діаграма FA-

рівня деталізує уявлення про інформаційні потреби предметної області до рівня атрибутів сутностей.

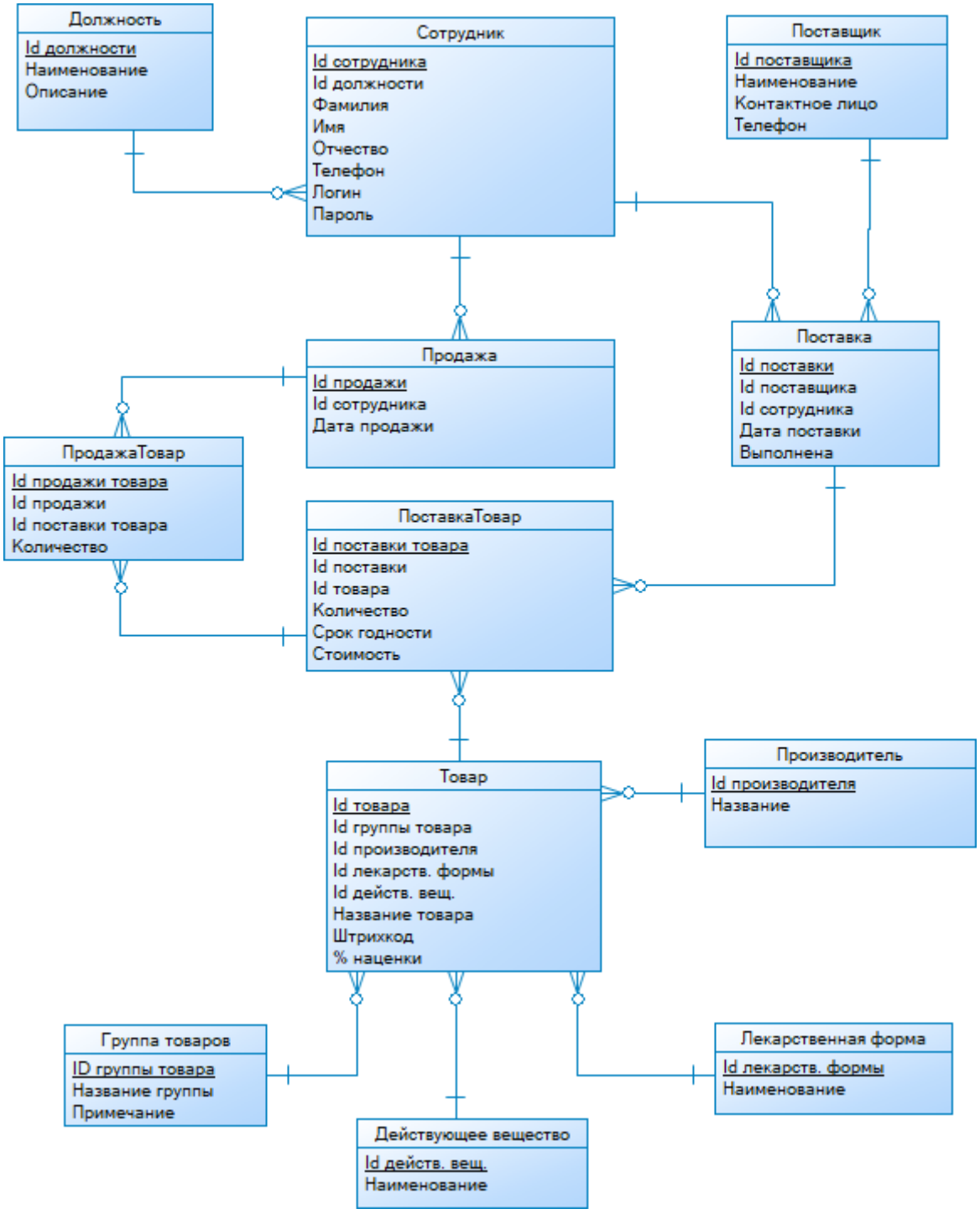


Рисунок 2.8 – Діаграма логічної моделі даних FA-рівня

Коректність логічної моделі перевіряється за допомогою правил нормалізації [15]:

- - атрибути є простими, всі використовувані домени містять тільки скалярні значення, в рядках таблиць немає повторень, отже, все таблиці перебувають в 1НФ;

- - для наборів значень, що відносяться до декількох записів, створені окремі таблиці, і зв'язок цих таблиць здійснюється за допомогою зовнішнього ключа, отже, все таблиці перебувають у 2НФ;

- - все таблиці перебувають в 3НФ, так як всі атрибути з транзитивними залежностями виділені в окремі таблиці.

Таким чином, можна зробити висновок, що модель даних нормалізована і відповідає трьом нормальним формам.

Логічна модель є джерелом інформації для фази фізичного проектування БД і для проектування програми.

2.2.3 Глосарій сутностей і атрибутів

Суті моделі наведені в таблиці 2. Ця таблиця містить імена і визначення всіх сутностей моделі.

Таблиця 2 - Сутності моделі спроектованої бази даних

Ім'я	**Визначення**
Група товарів	Класифікації асортименту товарів, що об'єднує товари з широкого ознакою використання або виробництва
Діюча речовина	Хімічна речовина або унікальна біологічна субстанція в складі лікарського засобу, з фізіологічним дією якої на організм пов'язують лікувальні властивості даного препарату

Кінець таблиці №2

Ім'я	Визначення
Посада	Правова освіта, первинна неподільна структурна одиниця в організації або поза нею, займана фізичною особою, що відповідає встановленим кваліфікаційним вимогам
Лікарська форма	Що надається лікарського засобу або лікарській рослинній сировині зручний для застосування стан, при якому досягається необхідний лікувальний ефект
Поставка	Заключний етап виконання домовленості купівлі-продажу, на якому відбувається оплатне передача товару продавцем покупцеві
Поставка Товар	Поставка конкретного товару покупцеві
Постачальник	Будь-яка юридична або фізична особа, які постачають товари або послуги замовникам
Продаж	Обмін товару на гроші, підтверджений чеком продажу
Продаж Товар	Продаж конкретного товару покупцеві
Виробник	Набір повноважень, призначений користувачеві, який дозволяє виконувати певні операції в системі
Співробітник	Особа, яка працює в аптечній організації
Товар	Медикаменти, що продаються в аптеці

Імена та визначення всіх атрибутів моделі представлені в таблиці 3.

Таблиця 3 - Атрибути моделі проектованої бази даних

Ім'я атрибута	Сенс	Сутність
% наценки	Відсоток націнки на поставлений товар	Товар

Продовження таблиці №3

Ім'я атрибута	Сенс	Сутність
Id группы товара	Ідентифікатор групи товарів	Группа товаров
Id группы товара	Ідентифікатор групи товару	Товар
Id действ. вещ.	Ідентифікатор діючих речовина	Действующее вещество
Id действ. вещ.	Ідентифікатор діючої речовини товару	Товар
Id должности	Ідентифікатор посади співробітника	Сотрудник
Id должности	ідентифікатор посади	Должность
Id лекарств. формы	Ідентифікатор лікарської форми товару	Товар
Id лекарств. формы	Ідентифікатор лікарської форми	Лекарственная форма
Id поставки	Ідентифікатор поставки для товара	Поставка Товар
Id поставки	ідентифікатор поставки	Поставка
Id поставки товара	Ідентифікатор поставки товару	Поставка Товар
Id поставки товара	Ідентифікатор поставки товару в таблиці продажу	Продажа Товар
Id поставщика	Ідентифікатор постачальника в таблиці поставок	Поставка
Id поставщика	ідентифікатор постачальника	Поставщик
Id продажи	ідентифікатор продажу	Продажа
Id продажи	Ідентифікатор продажу в таблиці продажів товару	ПродажаТовар
Id продажи товара	Ідентифікатор продажу товару	ПродажаТовар
Id производителя	Ідентифікатор виробника в таблиці товарів	Товар
Id производителя	ідентифікатор виробника	Производитель
Id сотрудника	Ідентифікатор співробітника в таблиці поставок (оформлення поставки)	Поставка
Id сотрудника	ідентифікатор співробітника	Сотрудник
Id сотрудника	Ідентифікатор співробітника в таблиці продажів (продаж товару)	Продажа
Id товара	ідентифікатор товару	Товар
Id товара	Ідентифікатор товару в таблиці поставок	Поставка Товар

Кінець таблиці №3

Ім'я атрибута	Сенс	Сутність
Выполнена	Маркер виконання факту поставки медикаментів в організацію	Поставка
Дата поставки	Дата поставки медикаментів	Поставка
Дата продажи	Дата продажу медикаментів клієнту	Продажа
Имя	Ім'я співробітника організації	Сотрудник
Количество	Кількість проданих одиниць товару	Продажа Товар
Количество	Кількість поставлених одиниць товару	Поставка Товар
Контактное лицо	Контактна особа постачальника	Поставщик
Логин	Логін для входу в програму	Сотрудник
Название	Найменування виробника товару	Производитель
Название группы	Найменування групи товарів	Группа товаров
Название товара	Найменування аптечного товару	Товар
Наименование	Найменування діючої речовини	Действующее вещество
Наименование	Найменування посади співробітника	Должность
Наименование	Найменування постачальника товарів	Поставщик
Наименование	Найменування лікарської форми	Лекарственная форма
Описание	Опис посади співробітника	Должность
Отчество	По батькові співробітника аптеки	Сотрудник
Пароль	Пароль для входу в програму	Сотрудник
Примечание	Примітка до групи товарів	Группа товаров
Срок годности	Термін придатності поставленого товару	Поставка Товар
Стоимость	Оптова вартість товару	Поставка Товар
Телефон	Телефон співробітника	Сотрудник
Телефон	Телефон постачальника	Поставщик
Фамилия	Прізвище співробітника аптеки	Сотрудник
Штрихкод	Штрихкод товару в форматі EAN13	Товар

2.3 Діаграма класів

Для спрощення роботи з базою даних було вирішено застосувати технологію об'єктно-реляційного відображення (ORM). ORM дозволяє

абстрагуватися від способу зберігання даних і маніпулювати даними на рівні об'єктів, що дозволяє з легкістю переходити від використання однієї СУБД до іншої.

Найбільш популярною ORM для .NET є Entity Framework [16]. Entity Framework має вбудований в Visual Studio графічний редактор і підтримує три підходи до створення об'єктної моделі і бази даних: «Code First», «Database First» і «Model First».

«Code First» дозволяє визначити модель за допомогою класів C # або VB.Net, створює базу даних і додає в неї таблиці на основі коду.

«Database First» дозволяє реконструювати модель на основі існуючої бази даних.

«Model First» дозволяє створити нову модель за допомогою конструктора Entity Framework, а потім сформувати схему бази даних на основі моделі. Модель зберігається в EDMX-файлі. Її можна переглядати і змінювати в конструкторі Entity Framework. На основі EDMX-файлу автоматично формуються класи в додатку, з якими відбувається взаємодія.

Для розробки інформаційної системи найбільш зручний підхід DataBaseFirst, так як база даних вже спроектована.

Діаграма класів, що представляють об'єктну модель бази даних, представлена на рисунку 2.9.

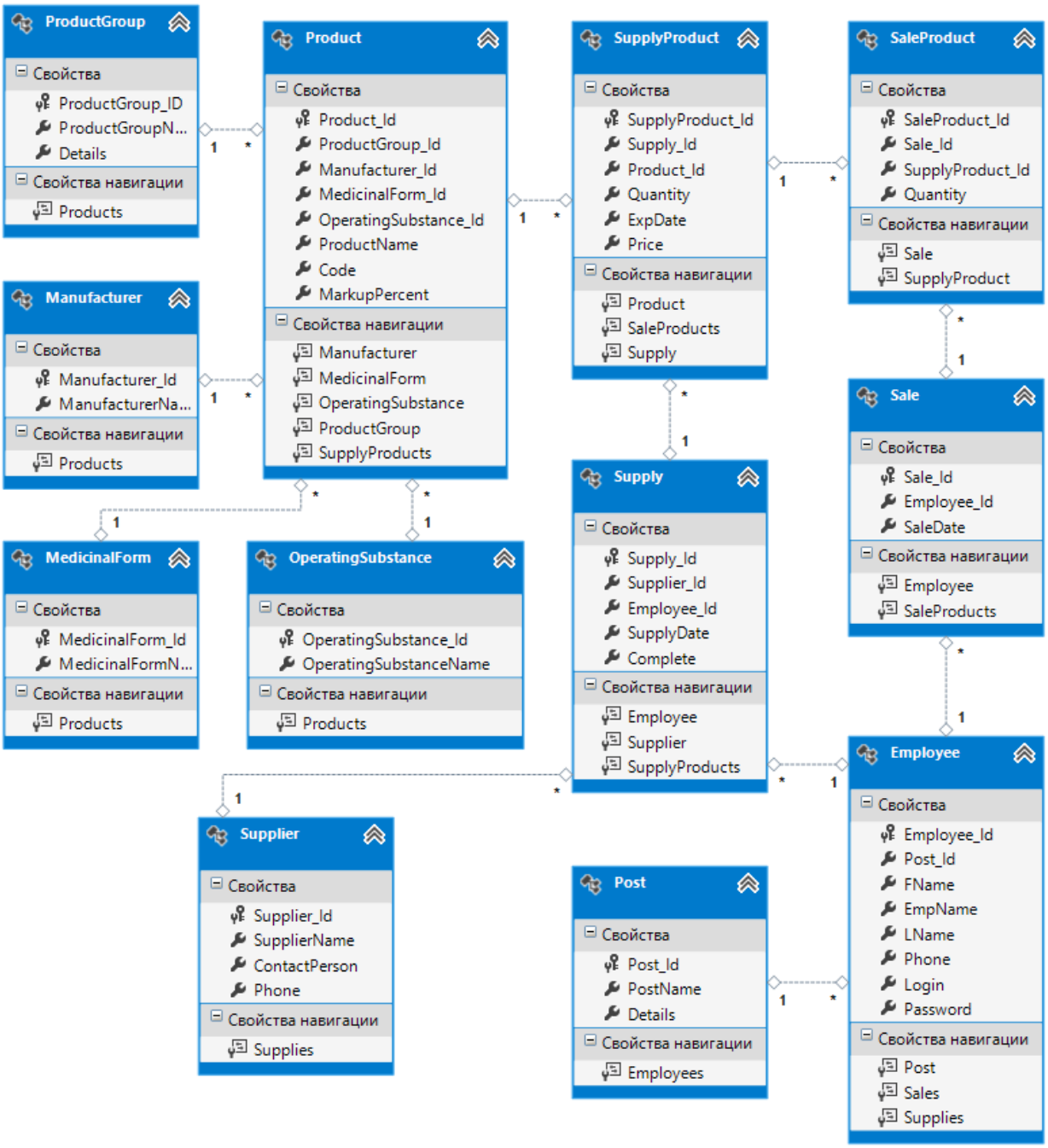


Рисунок 2.9 – Діаграма класів, що представляють об'єктну модель бази даних

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Фізична реалізація бази даних

Фізична модель даних для MS SQL Server представлена на рисунку 3.1. Фізична модель - логічна модель бази даних, виражена в термінах мови опису даних конкретної СУБД.

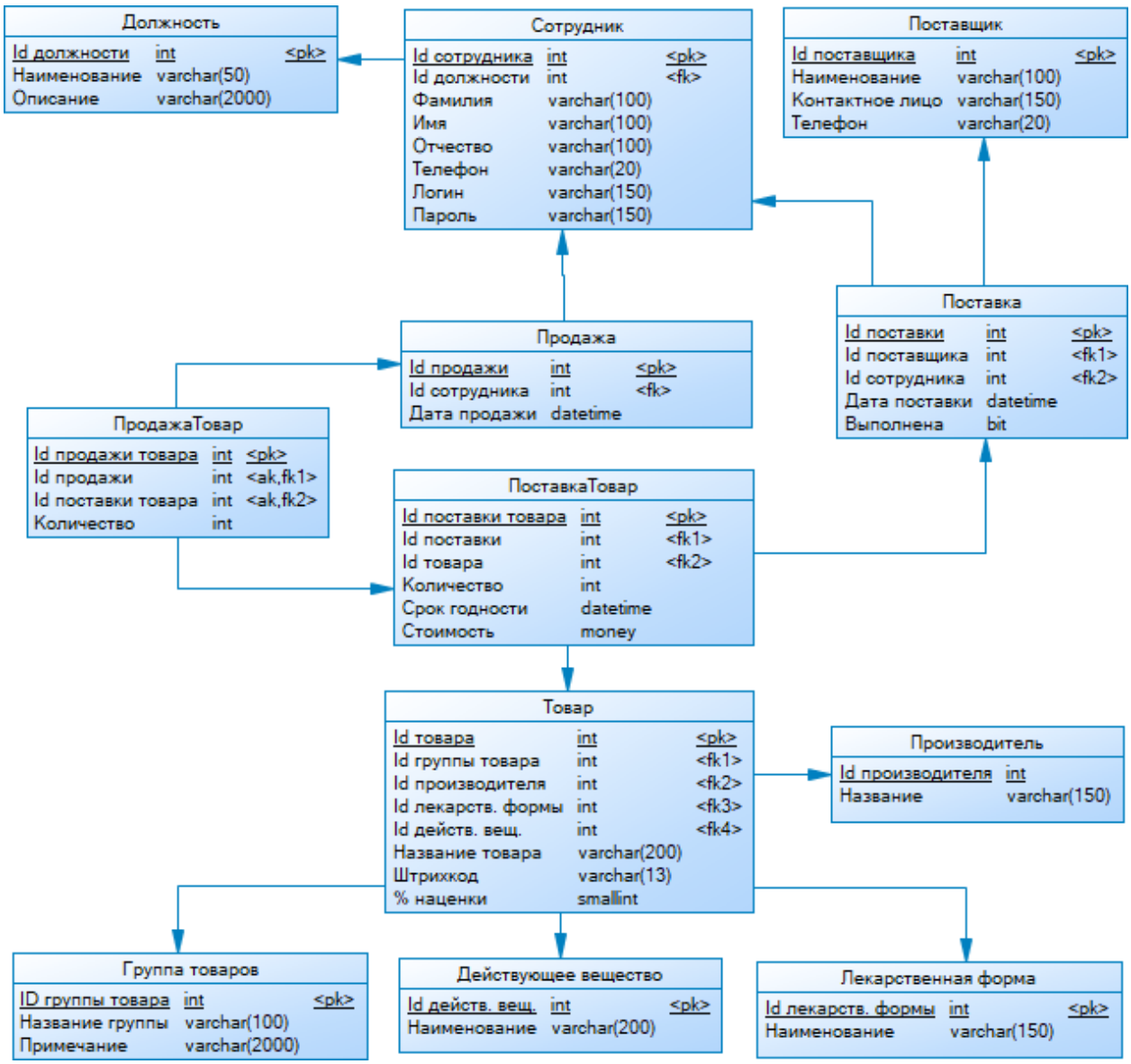


Рисунок 3.1 – Фізична модель бази даних MS SQL Server

Фізична модель бази даних містить всі деталі, необхідні конкретної СУБД для створення бази: найменування таблиць і стовпців, типи даних, визначення первинних і зовнішніх ключів і т.п.

Забезпечення унікальності значень в таблицях досягнуто шляхом визначення обмежень або індексів для полів, які потребують унікальність.

Одним з необхідних шляхів досягнення високої продуктивності сервера бази даних є використання індексів. Індекс прискорює процес запиту, надаючи швидкий доступ до рядків даних в таблиці [17]. Для кожної таблиці були задані необхідні індекси.

3.2 Опис програмної реалізації

3.2.1 Структура проекту

В основу реалізованої автоматизованої інформаційної системи покладена архітектура Model View Controller (MVC). MVC - це архітектура програмного забезпечення, в якій модель даних програми, графічний інтерфейс і логіка управління розділені на три окремих компонента таким чином, що зміна одного з компонентів мінімально впливають на інші компоненти.

Шаблон MVC дозволяє розділити дані, подання та обробку дій користувача на три окремих компонента:

- модель (Model), надає дані (зазвичай для View), а також реагує на запити (зазвичай від контролера), змінюючи свій стан;
- подання (View), відповідає за відображення інформації (призначений для користувача інтерфейс);
- контролер (Controller), інтерпретує дані, введені користувачем, і інформує модель і уявлення про необхідність відповідної реакції.

Важливо відзначити, що як уявлення, так і контролер залежать від моделі. Однак модель не залежить ні від уявлення, ні від контролера. Це одне

з ключових переваг подібного поділу. Воно дозволяє будувати модель незалежно від візуального представлення [18, 19].

Структура проекту програми представлена на рисунку 3.2.

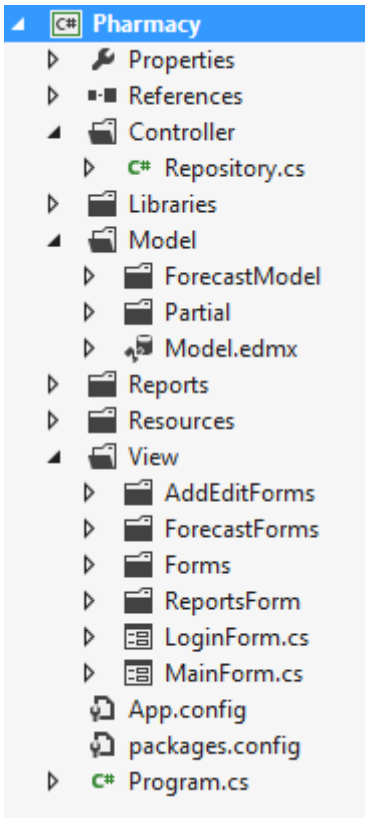


Рисунок 3.2 – Структура проекту інформаційної системи

В папці Controller міститься клас Repository, який надає методи для обміну даними з базою даних MS SQL Server. Схема класу Repository і його опис представлено в розділі 3.2.2.

Папка Model містить файл моделі даних Model.edmx, згенерувала за допомогою конструктора Entity Framework, а також папку Partial. В папці Partial знаходяться класи, помічені ключовим словом partial. Даний модифікатор дозволяє розділяти визначення для класів, структур і інтерфейсів на кілька файлів [10]. Компілятор об'єднує всі часткові файли класу під час компіляції. Папка ForecastModel містить реалізацію класів для побудови прогнозу продажів.

Використання механізму часткових класів дозволяє розширити функціональність класів моделі, не зачіпаючи автоматично згенерували Entity Framework сутнісні класи. Прикладом використання часткових класів в проекті є перевизначення методів порівняння об'єктів моделі даних - Equals () і GetHashCode() (порівняння по первинному ключу сутності), а також додавання необхідних властивостей і методів.

В папці Reports містяться файли звітів з розширенням .rdlc. Дані файли використовуються при побудові звітів і виведення їх на друк.

Папка Resources містить ресурси програми (іконки для елементів управління).

В папці View розташовані класи з формами додатки. В окремих підпапках містяться форми для додавання і редагування сутностей, форми з таблицями для виведення наборів сутностей, а також форми для виведення звітів. У корені даної папки розташована головна форма програми, а також форма для входу в додаток.

Файл App.config - конфігураційний файл програми. В даному файлі знаходиться строка підключення до бази даних.

Файл Program.cs містить в собі точку входу в додаток (метод Main).

3.2.2 Опис класу доступу до бази даних

Розглянемо методи класу Repository в проекті програми. Схема класу представлена на рисунку 3.3.

Метод AddEntity <T> () додає новий запис в базу даних. Параметр типу T вказується при виклику методу. Таким чином даний метод є універсальним для додавання в базу об'єктів сутностей. Після виконання даний метод повертає об'єкт доданої суті з унікальним ID, згенерований базою даних.

Метод CreateBarcode () створює зображення штрих коду по 13-значному коду товару. Для генерації зображення використовується безкоштовна бібліотека BarcodeLib.dll.

Рисунок 3.3 – Схема класса Repository

Статичний метод FirstStart() використовується при першому запуску програми. Якщо в базі даних немає адміністратора, то скрипт буде працювати без введення логіна і пароля, дозволяючи додати необхідні довідкові дані і адміністратора системи.

Метод GetEntity() дозволяє виконувати пошук в базі даних одиничної суті по заданій умові.

Метод GetEntityes<T> дозволяє отримати колекцію об'єктів заданої сутності. Параметр типу T вказується при виклику методу. Так само даний метод має перевантажену версію, яка на вході приймає метод-предикат - умова вибору об'єктів суті.

Метод GetForecastProductData() готує дані для прогнозування по заданому товару. У методі виконується наступний метод з SQL-запитом до бази даних:

```
public List<ForecastData> GetForecastProductData(SupplyProduct supplyProduct)
        {
            using (var context = new PharmacyDBEntities())
            {
                var query = "" +
                    "SELECT YEAR(Sale.SaleDate) AS Year," +
                    " MONTH(Sale.SaleDate) AS Month," +
                    " SUM(SaleProduct.Quantity) AS Count," +
                    " Product.ProductName AS Assortment " +
                    " FROM dbo.SaleProduct INNER JOIN dbo.Sale ON
SaleProduct.Sale_Id = Sale.Sale_Id " +
                    " INNER JOIN dbo.SupplyProduct ON
SaleProduct.SupplyProduct_Id = SupplyProduct.SupplyProduct_Id " +
                    " INNER JOIN dbo.Product ON SupplyProduct.Product_Id =
Product.Product_Id " +
                    " WHERE SupplyProduct.Product_Id =
"+supplyProduct.Product_Id+
                    " GROUP BY YEAR(Sale.SaleDate) ,MONTH(Sale.SaleDate)
,Product.ProductName ORDER BY Year";

                return
context.Database.SqlQuery<ForecastData>(query).ToList();
            }
              }
```

Таким чином, метод дозволяє отримати сумарні продажі обраного товару, згрупованого по місяцях і року.

Метод HandleException () використовується для обробки винятків, пов'язаних із запитами до бази даних. Даний метод витягує повідомлення з вкладених винятків і виводить повідомлення з помилкою на екран.

Метод RemoveEntity <T> реалізує видалення об'єкта з бази даних. Метод є універсальним для всіх сутностей, тип сутності вказується при виклику методу за допомогою параметра T.

Метод SearchProducts () використовується для пошуку товару по заданих параметрах. На вхід даний метод приймає рядок з назвою товару або його частиною, штрих код товару, групу і діюча речовина. Якщо який-небудь з параметрів вказано з нульовим ідентифікатором, то в пошуку він не використовується.

Метод SearchSupply () використовується для пошуку поставки товару за заданими параметрами. На вхід даний метод приймає об'єкт постачальника і дату поставки. Якщо який-небудь з параметрів вказано з нульовим ідентифікатором, то в пошуку він не використовується.

Метод UpdateEntity <T> () використовується для поновлення сутностей в базі даних. Метод є універсальним, параметр типу T вказується при виклику методу.

Метод UserLogin() перевіряє наявність в базі даних співробітника із зазначеним логіном і паролем. Якщо збіг знайдено метод повертає значення true, в іншому випадку - значення false.

Публічна статична властивість CurrentEmployee містить посилання на об'єкт співробітника, який пройшов авторизацію в програмі за допомогою логіна і пароля.

Код класу Repository.cs наведено в додатку Б.

3.2.3 Реализация алгоритма прогнозирования

Схема класів використовуваних для роботи алгоритму прогнозування представлена на рисунку 2.4.

Класс ErrorData використовується для розрахунку і зберігання даних з помилками прогнозування. Клас має наступні публічні властивості:

public int Num – номер періоду;

public double Fact – фактичне значення продажів за даний період;

public double Model – розрахункове значення моделі за даний період;

public double Error – помилка моделі (фактичне значення мінус значення моделі);

public double SqError – середньоквадратична помилка.

Розрахунок значення середньоквадратичної помилки здійснюється за допомогою методу, представленого в лістингу 3.1

```
/// <summary>
/// Расчет среднеквадратичной ошибки для периода
/// </summary>
private double GetSkoError()
{
   //(факт-модель)^2/модель^2
   return Math.Pow(Error, 2) / Math.Pow(Model, 2);
}
```

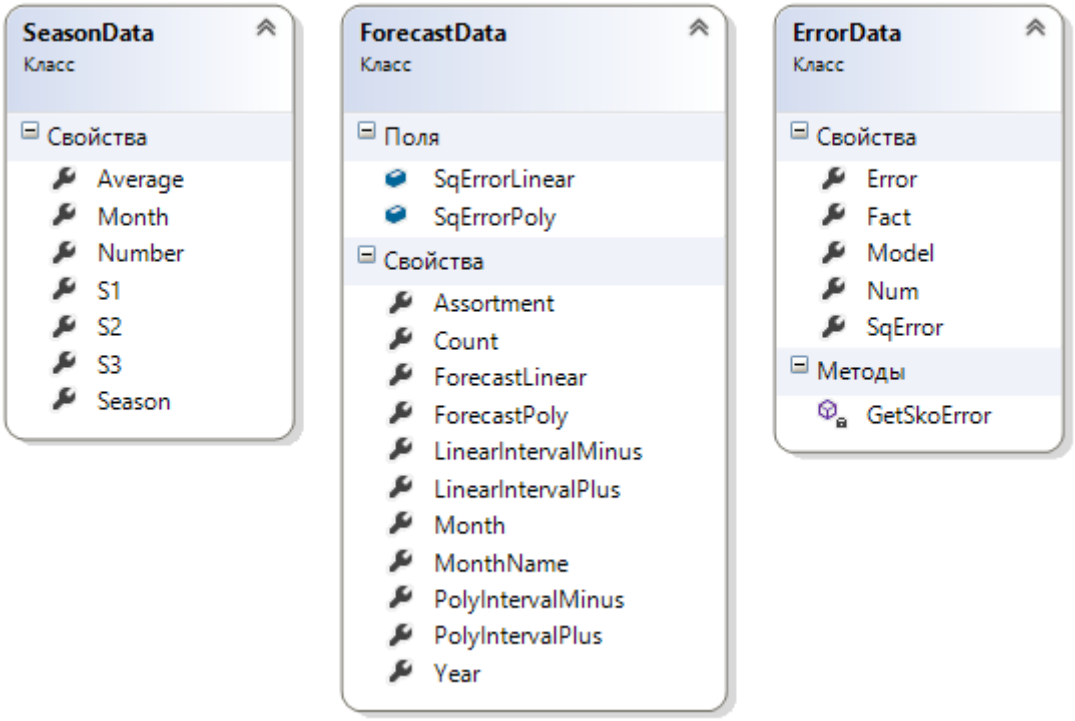


Рисунок 3.4 – Схема класів використовуваних для роботи алгоритму прогнозування

Клас ForecastData містить результати прогнозування. Клас містить наступні властивості:

public string MonthName – назва місяця;

public int Month – номер місяця;

public int Year – рік;

public string Assortment – назва товару;

public int Count – фактичне значення продажів за місяць;

public double ForecastLinear – прогнозне значення лінійної моделі;

public double ForecastPoly – прогнозне значення поліноміальної моделі;

public static double SqErrorLinear – квадратична помилка лінійної моделі;

public static double SqErrorPoly – квадратична помилка поліноміальної моделі.

Для розрахунку довірчих інтервалів реалізовані наступні автосвойства:

public double LinearIntervalPlus { get {return ForecastLinear * (1 + SqErrorLinear); } }

public double LinearIntervalMinus { get { return ForecastLinear * (1- SqErrorLinear); } }

public double PolyIntervalPlus { get { return ForecastPoly * (1 + SqErrorPoly); } }

public double PolyIntervalMinus { get { return ForecastPoly * (1 - SqErrorPoly); } }

Клас SeasonData містить дані розрахунку сезонної компоненти. У класі реалізовані наступні властивості:

public int Number – номер періоду;

public string Month – назва місяця;

public double S1 – сезонна компонента першого року;

public double S2 – сезонна компонента другого року;

public double S3 – сезонна компонента третього року;

public double Average – середнє значення сезонної компоненти за період;

public double Season – обчислене значення сезонної компоненти.

Для роботи алгоритму прогнозування необхідні дані за минулі три роки. При виборі товару в випадаючому списку здійснюється вибірка з бази даних за допомогою методу GetForecastData(), поданого в п. 3.2.2

В результаті виконання запиту виходить список сумарних продажів, згрупований по році і місяця продажу. Якщо обсягу даних буде недостатньо для роботи алгоритму прогнозування, то програма покаже вікно з попередженням.

Наступний етап роботи алгоритму прогнозування передбачає отримання рівняння лінійного і поліноміального тренда. Для цих цілей в програмі використовуються класи безкоштовної бібліотеки для наукових обчислень Accord.NET. Код отримання рівнянь тренда за допомогою даної

бібліотеки представлений нижче. Для отримання рівнянь на вхід методу подаються значення періода від 1 до 36 (3 роки), а так само фактичні значення продажів в моменти часу.

```
// первый параметр для нахождения уравнения тренда
// от 1 до 36 - 3 года
var inputs = Enumerable.Range(1, 36).Select(i => (double)i).ToArray();
// второй параметр - помесячные значения продаж для 36 месяцев
var outputs = _data.Select(data => (double)data.Count).ToArray();
// вычисление уравнения линейного тренда
var ols = new OrdinaryLeastSquares();
linearRegression = ols.Learn(inputs, outputs);
//вычисление уравнения полиноминального тренда
var ls = new PolynomialLeastSquares()
{
  Degree = 2 // степень полинома
};
// расчёт уравнения
_polynomialRegression = ls.Learn(inputs, outputs);
```

На наступному кроці алгоритму виробляється обчислення сезонної компоненти моделі. Код методу, що реалізує даний крок, представлений нижче.

```
private void CalculateSeason(out List<SeasonData> seasonDataList, Func<double,
double> transformMethod)
        {
            seasonDataList = new List<SeasonData>();
            for (int i = 1; i <= 12; i++)
            {
                var season = new SeasonData
                {
                    Number = i, // номер периода
                    // фактическое значение - минус значение по уравнению тренда
                    S1 = _data[i - 1].Count - transformMethod(i), // первый год
                    S2 = _data[i + 12 - 1].Count - transformMethod(i + 12), //
второй год
                    S3 = _data[i + 24 - 1].Count - transformMethod(i + 24), //
третий год
                    Month =
CultureInfo.CurrentCulture.DateTimeFormat.GetMonthName(i) // присваиваем месяцу
название
                };
                // получаем коллекцию, с расчитанными сезонными компонентами
                // за каждый месяц Average=(S1 + S2 + S3) / 3;
                // например, средняя сезонная компонента для января за три года
                seasonDataList.Add(season);
            }
            // вычисляем среднее значение сезонной компоненты за 12 месяцев
            var sumSeasonAverage = seasonDataList.Sum(data => data.Average);
            // вычитаем из каждого месяца сезонной компоненты полученное среднее
            // получаем скорректированную сезонную компоненту St
            foreach (var seasonData in seasonDataList)
            {
                seasonData.Season = seasonData.Average - sumSeasonAverage;
            }
                                        }
```

З значення продажів за місяць віднімаються значення, обчислені за допомогою рівняння тренду. Сезонні компоненти обчислюються для кожного року лінійної і поліноміальної моделі.

На наступному кроці роботи алгоритму обчислюються дані для розрахунку помилки моделі. Код, який реалізує даний крок представлений нижче. Метод обчислює місячні значення продажів отримані за допомогою рівняння тренда, скориговані сезонною компонентою за даний період. Далі за допомогою методу LINQ проводиться розрахунок середньоквадратичної помилки кожної моделі.

```
private void CalculateErrors(Func<double, double> transformMethod, List<SeasonData>
seasonData, out List<ErrorData> errorData)
        {
            errorData = new List<ErrorData>();
            int j = 1;
            for (int i = 0; i < _data.Count; i++)
            {
                var error = new ErrorData
                {
                    Num = i + 1, // номер периода
                    Fact = _data[i].Count, // фактическое значение
                    // значение по уравнению тренда + скорректир. значение сезонной
компоненты
                    Model = transformMethod(i) + seasonData[j - 1].Average
                    // в классе вычисляется ошибка модели - Error
                    // фактическое значение - значение по модели прогноза
                };
                errorData.Add(error);
                j++;
                // выборка скор. сезонной компоненты за 12 месяцев - период прогноза
                if (j == 12)
                {
                    j = 1;
                }
            }
        }
```

Після обчислення помилки алгоритм реалізує прогнозування значень продажів на наступний рік. Код, який реалізує фінальний крок алгоритму представлений нижче. Для обчислення прогнозного значення дані, обчислені за допомогою рівняння тренда за прогнозований період, складаються зі

значенням сезонної компоненти в даний період. У класі ForecastData, після заповнення полів даними, на підставі значень середньоквадратичної помилки обчислюються довірчі інтервали кожної моделі.

```
private void ForecastResult()
        {
            // данные текущего года
            var currentYearData = _fullData.Where(data => data.Year ==
DateTime.Now.Year).ToList();
            var first = currentYearData.First(); // берем цифру текущего года
            _forecastData = new List<ForecastData>(12);
            for (int i = 1; i <= 12; i++)
            {
                var dat = new ForecastData
                {
                    Year = first.Year,
                    Month = i,
                    Assortment = first.Assortment
                };
                if (i < currentYearData.Count)
                {
                    dat.Count = currentYearData[i - 1].Count;
                }
                dat.ForecastLinear = _linearRegression.Transform(i + 36) +
_seasonDataLinear[i - 1].Season;
                dat.ForecastPoly = _polynomialRegression.Transform(i + 36) +
_seasonDataPolynominal[i - 1].Season;
                _forecastData.Add(dat);
            }
            dataGridViewForecast.DataSource = _forecastData;
        }
```

3.3 Опис графічного інтерфейсу системи

Вікно введення логіна і пароля для входу в програму представлено на рисунку 3.5.

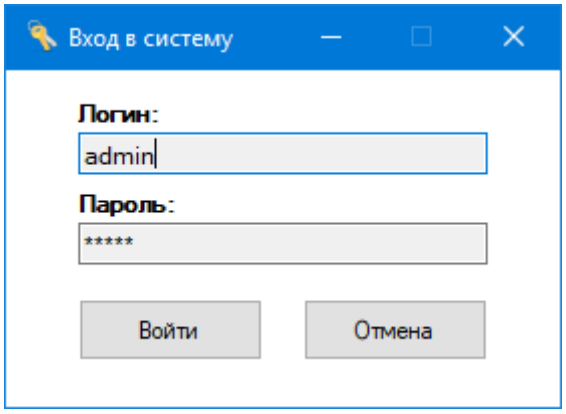


Рисунок 3.5 – Вікно входу в програму

Головне вікно програми для ролі адміністратор представлено на рисунку 3.6. ПІБ користувача і його роль в системі відображається в нижній частині форми. На головне формі, в таблиці, відображаються продажу увійшов співробітника. Адміністратору доступні всі продажі в системі.

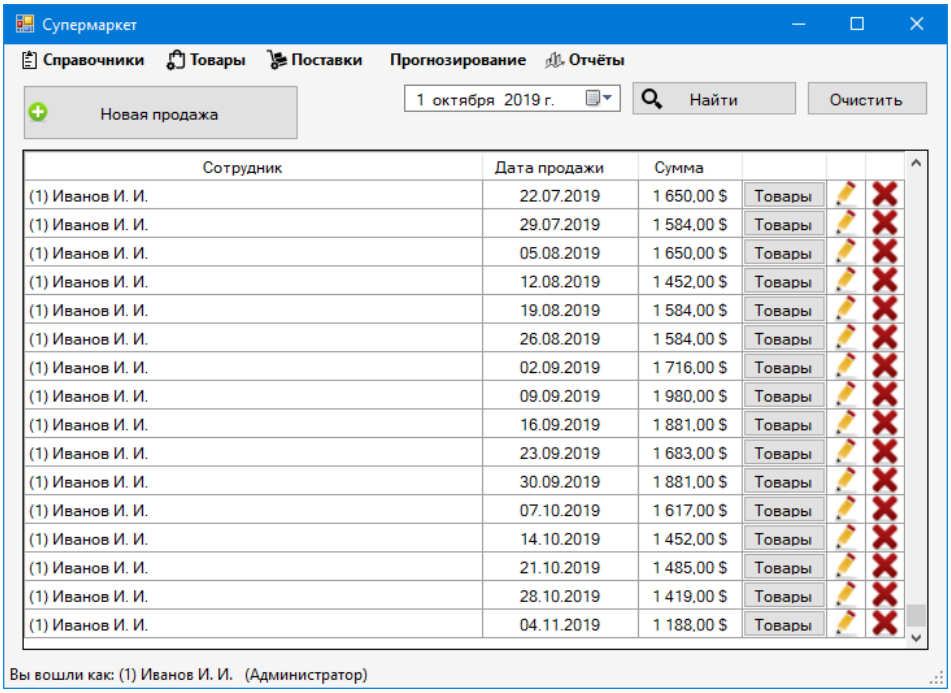


Рисунок 3.6 – Головне вікно інформаційної системи

Пункт меню «Довідники» представлено на рисунку 3.7.

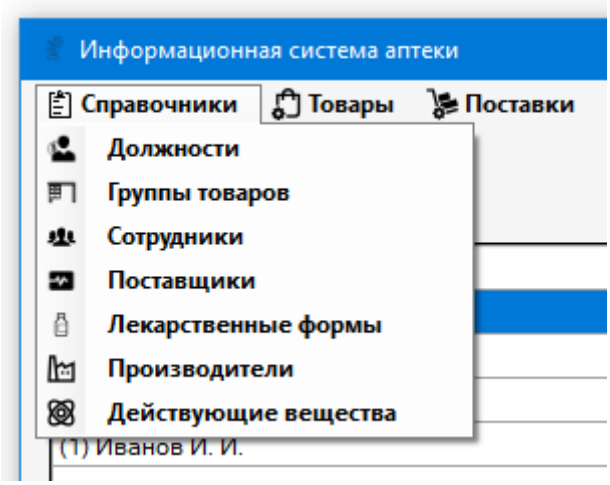


Рисунок 3.7 – Розгорнутий пункт меню «Довідники»

Приклад довідкової форми «Співробітники» представлено на рисунку 3.8. Для кожного запису в таблиці, в правій частині, доступні кнопки для редагування і видалення записів. Всі форми з довідковими даними оформлені в одному виді.

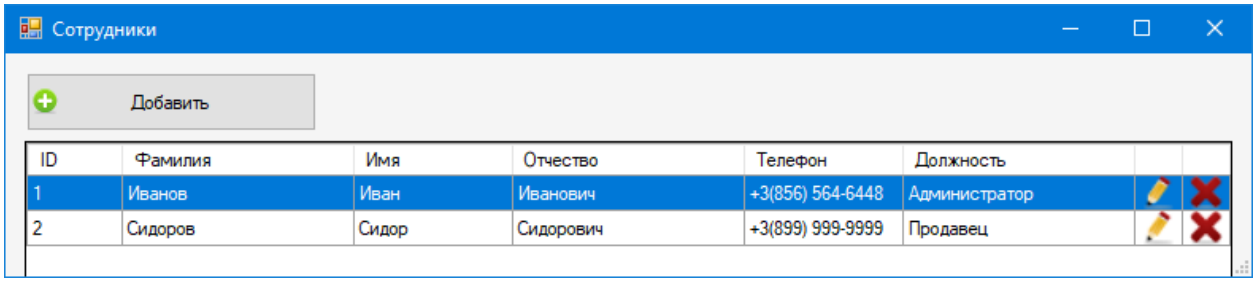


Рисунок 3.8 – Зовнішній вигляд форми «Співробітники»

Форма Додавання и Редагування співробітників представлена на рисунку 3.9. При завантаженні даної форми автоматично підключаються дані зі довідника посад співробітників. При Першому запуску системи вікно Введення логіна и паролі не Відображається. Таким чином дерло в систему додається обліковій Запис адміністратора. Перший запуск програми визначається відсутністю записів в таблиці співробітників в базі даних.

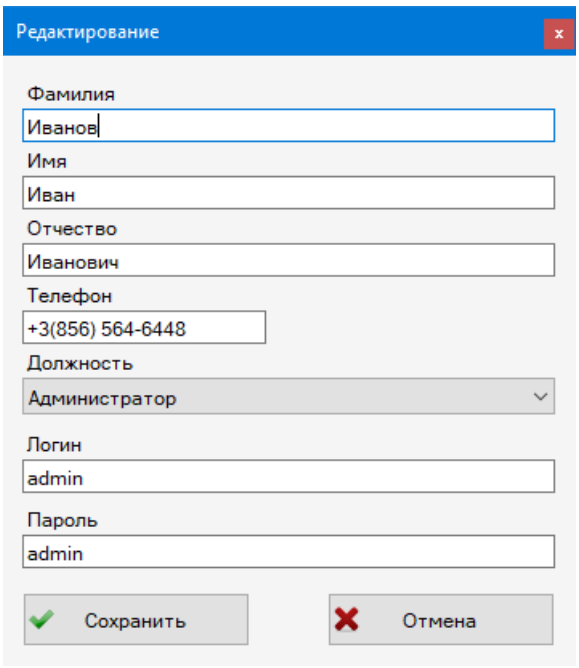


Рисунок 3.9 – Форма редагування і додавання співробітника

Пункт меню «Товари» представлено на рисунку 3.10. За допомогою даної форми здійснюється додавання, видалення і редагування номенклатури аптечних товарів. У формі реалізований пошук необхідних товарів по заданих параметрах.

Так само на цій формі відображається наявність товару на складі. Залишок розраховується як різниця між проданим і поставленою кількістю. При відсутності товару на складі продаж неможлива.

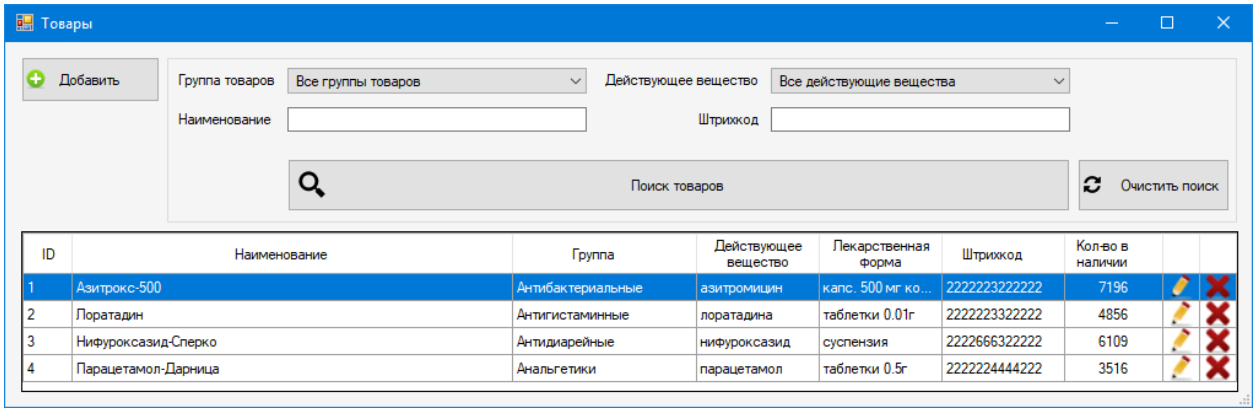


Рисунок 3.10 – Зовнішній вигляд форми «Товари»

Форма додавання і редагування товару представлена на рисунку 3.11

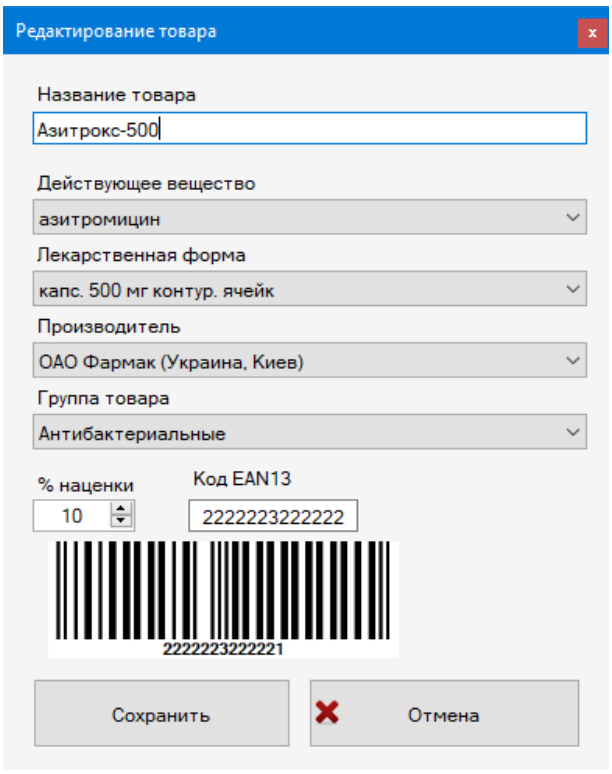


Рисунок 3.11 – Форма додавання і редагування товару

Форма «Поставки» представлена на рисунку 3.12. На даній формі відображаються всі поставки товарів в аптеку. Доступний пошук позиції по найменуванню постачальника і дати поставки. Форма додавання і редагування поставки представлена на рисунку 3.13. У кожному рядку таблиці розташована кнопка «Товари», після натискання на яку відкривається вікно з позиціями товарів в даній поставці (рис. 3.14).

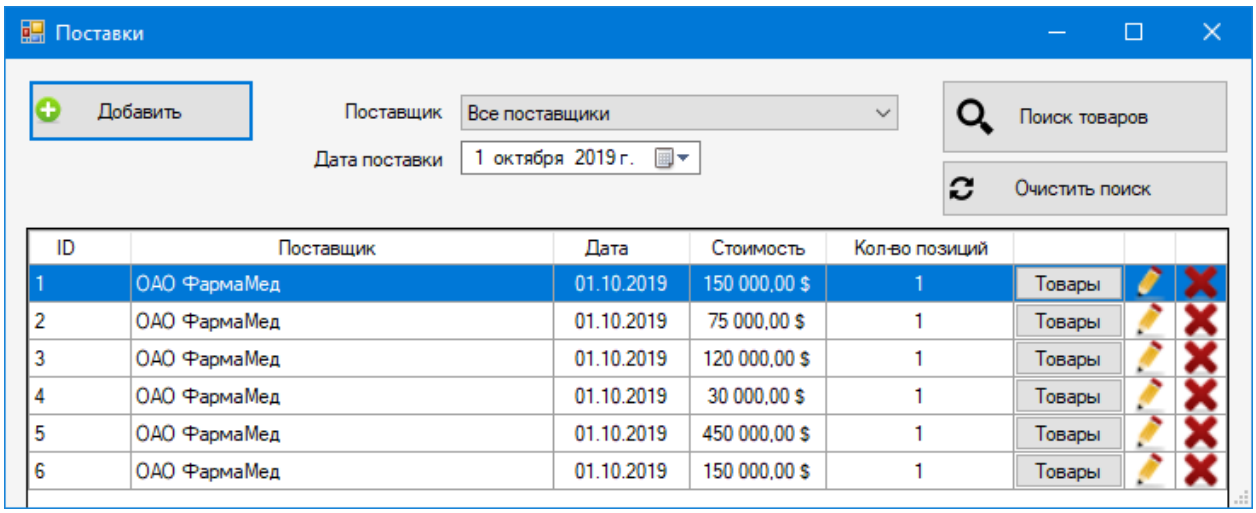


Рисунок 3.12 – Зовнішній вигляд вікна «Поставки»

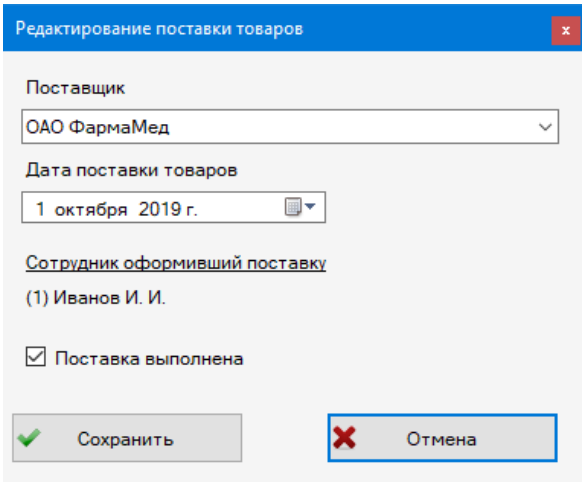


Рисунок 3.13 – Форма додавання і редагування поставки товару

При додаванні нового товару в поставку відкривається форма «Товари» в режимі вибору (доступний тільки пошук товарів і можливість вибору).

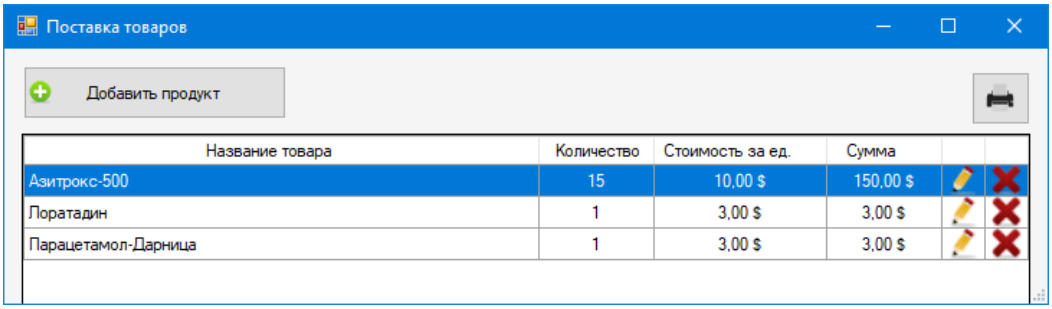


Рисунок 3.14 – Форма редагування товарів в постачанні

З форми редагування номенклатури товарів в постачанні можливий друк звіту (рис. 3.15).

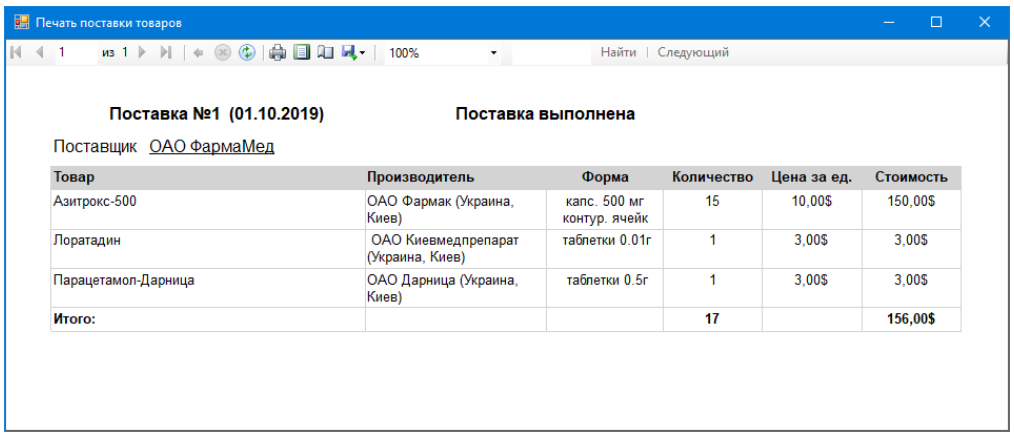


Рисунок 3.15 – Форма звіту щодо постачання товару

При створенні нового продажу аптечних товарів або редагування вже створеної відображається форма з вибором дати (рис. 3.16).

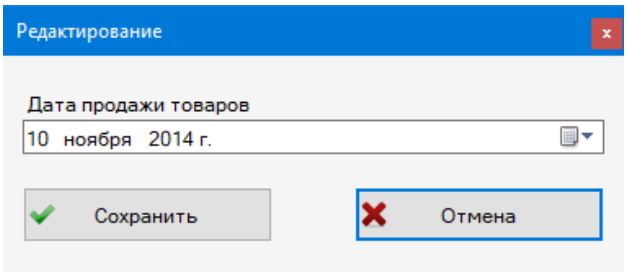


Рисунок 3.16 – Вікно створення і редагування продажу товару

З головної форми програми, в таблиці продажів, за допомогою натискання кнопки «Товари» відображається вікно редагування проданих товарів. При додаванні або редагування товарів продажу відкривається форма «Товари» в режимі вибору (можливий пошук і вибір). Так само з даної форми можливий друк чека (рис. 3.18)

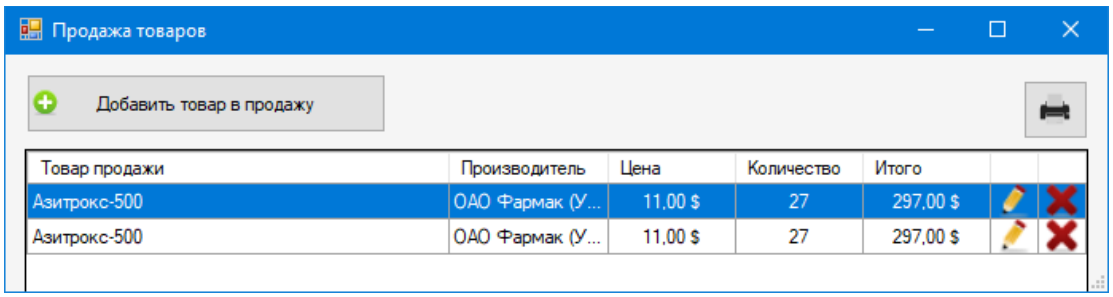


Рисунок 3.17 – Зовнішній вигляд форми продажу товарів

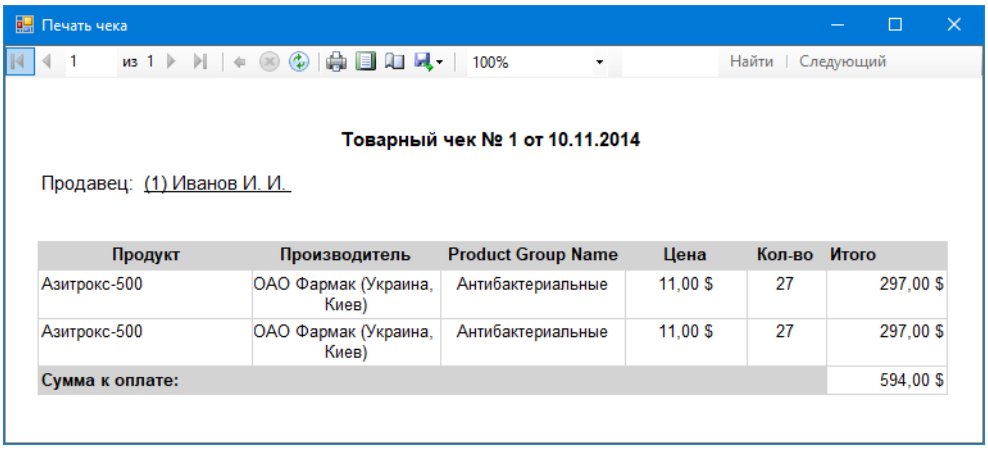


Рисунок 3.18 – Форма друку чека продажу

Пункт меню «Звіти» дозволяє переглянути і вивести на друк діаграму продажів по групі товару (рис. 3.19), а також звіт за обсягами продажів поточного співробітника (рис. 3.20). Перед висновком звіту користувач повинен вказати необхідний діапазон дат (місяць за замовчуванням).

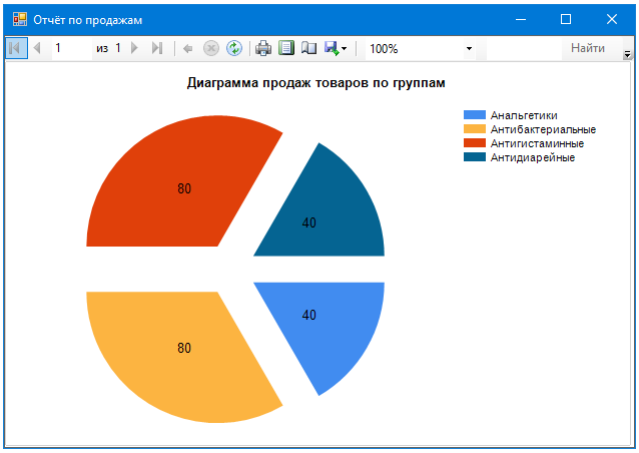


Рисунок 3.19 – Діаграма продажів товарів по групам

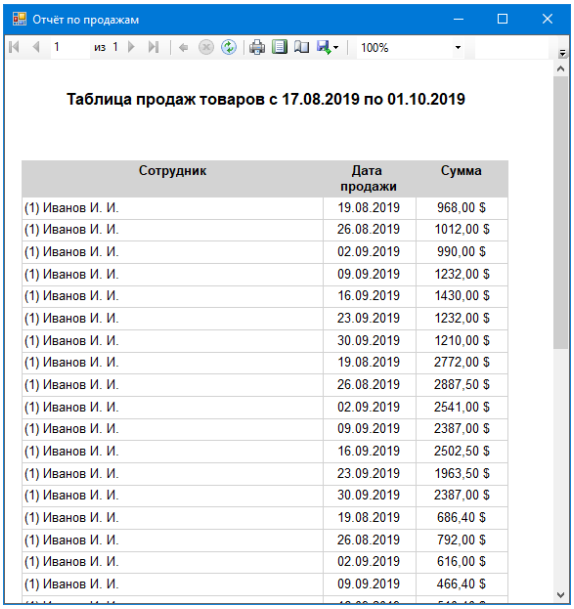


Рисунок 3.20 – Таблиця продажів за обраний період

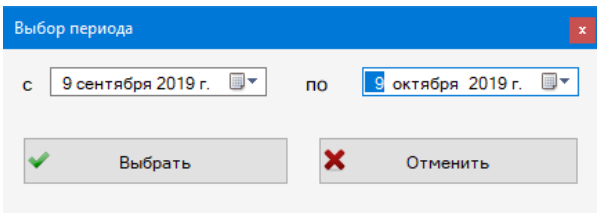
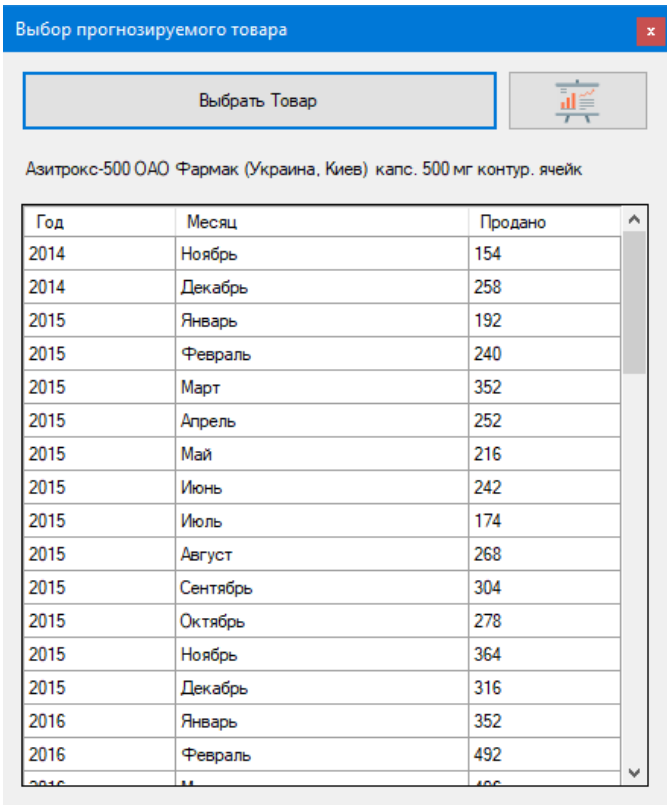


Рисунок 3.21 – Форма вибору діапазону дат для формування звіту

3.3 Приклад роботи алгоритму прогнозування

Форма вибору товару для прогнозування продажів представлена на рисунку 3.22. Після вибору товару в таблиці відображаються сумарні продажі згруповані по році і місяця. Для роботи алгоритму потрібні дані продажів товару не менш ніж за три роки.



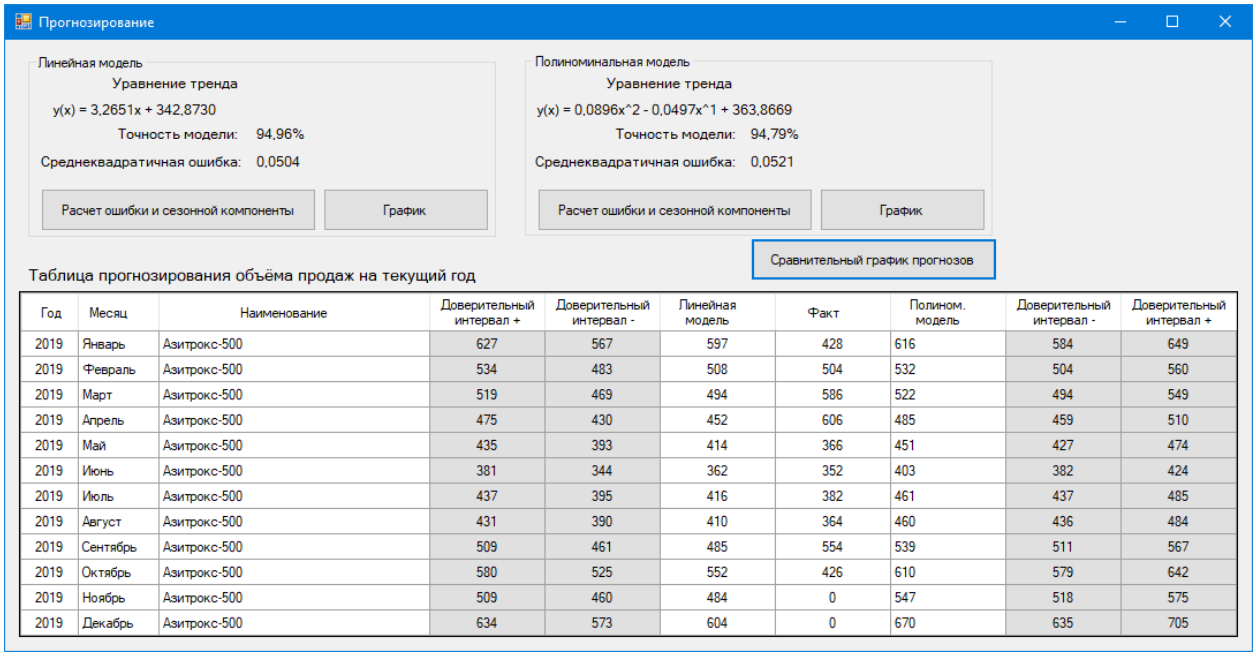
Год	Месяц	Продано
2014	Ноябрь	154
2014	Декабрь	258
2015	Январь	192
2015	Февраль	240
2015	Март	352
2015	Апрель	252
2015	Май	216
2015	Июнь	242
2015	Июль	174
2015	Август	268
2015	Сентябрь	304
2015	Октябрь	278
2015	Ноябрь	364
2015	Декабрь	316
2016	Январь	352
2016	Февраль	492

Рисунок 3.22 – Форма вибору товару для прогнозування продажів

Головна форма прогнозування продажів представлена на рисунку 3.23. У лівій верхній частині форми представлено розрахункове рівняння тренду для лінійної моделі, і, в правій частині - рівняння поліноміальної моделі.

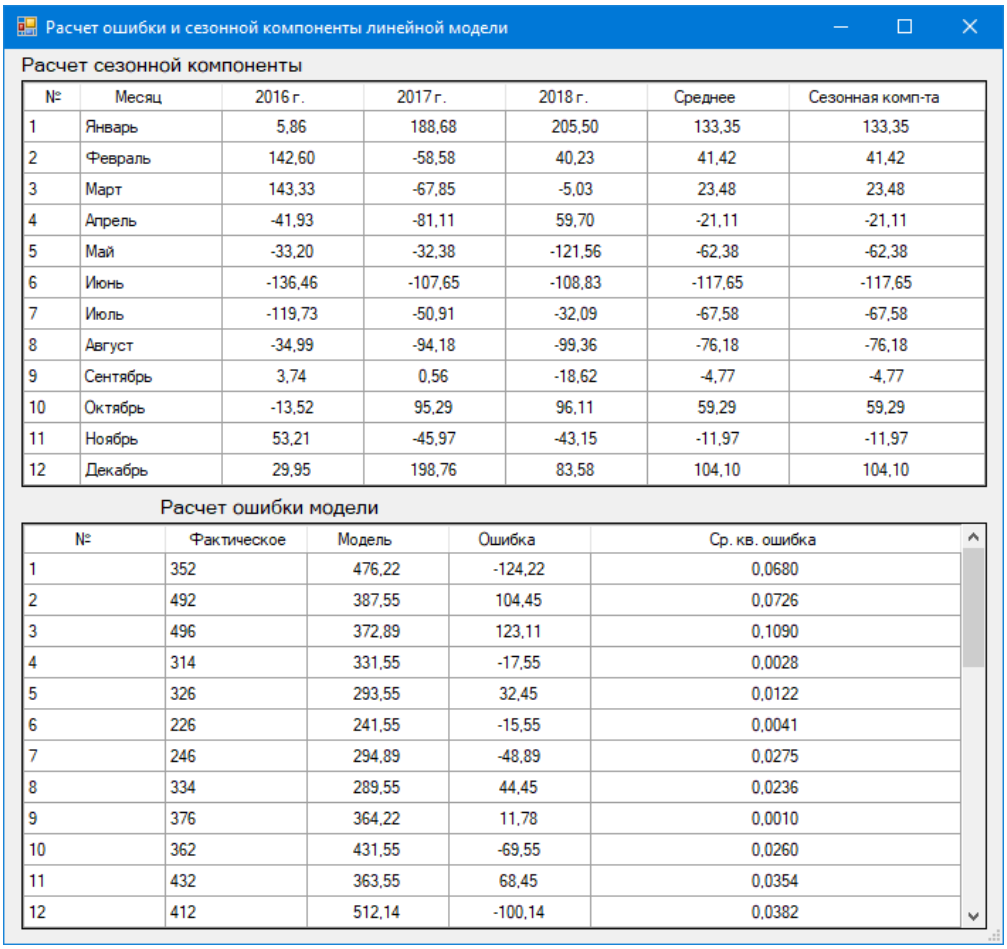
Моделі прогнозування були розраховані на підставі продажів обраного препарату за 2016 - 2018 р.р. У таблиці представлені спрогнозовані значення обраного препарату на поточний рік за двома моделями, реальні значення продажів для порівняння, а так само довірчі інтервали моделей. Розрахункові

значення, а так само порівняльні графіки представлені на рисунках 3.24 - 3.27.



Год	Месяц	Наименование	Доверительный интервал +	Доверительный интервал -	Линейная модель	Факт	Полином. модель	Доверительный интервал -	Доверительный интервал +
2019	Январь	Азитрокс-500	627	567	597	428	616	584	649
2019	Февраль	Азитрокс-500	534	483	508	504	532	504	560
2019	Март	Азитрокс-500	519	469	494	586	522	494	549
2019	Апрель	Азитрокс-500	475	430	452	606	485	459	510
2019	Май	Азитрокс-500	435	393	414	366	451	427	474
2019	Июнь	Азитрокс-500	381	344	362	352	403	382	424
2019	Июль	Азитрокс-500	437	395	416	382	461	437	485
2019	Август	Азитрокс-500	431	390	410	364	460	436	484
2019	Сентябрь	Азитрокс-500	509	461	485	554	539	511	567
2019	Октябрь	Азитрокс-500	580	525	552	426	610	579	642
2019	Ноябрь	Азитрокс-500	509	460	484	0	547	518	575
2019	Декабрь	Азитрокс-500	634	573	604	0	670	635	705

Рисунок 3.23 – Головна форма відображення результатів роботи алгоритму прогнозування продажів



№	Месяц	2016 г.	2017 г.	2018 г.	Среднее	Сезонная комп-та
1	Январь	5,86	188,68	205,50	133,35	133,35
2	Февраль	142,60	-58,58	40,23	41,42	41,42
3	Март	143,33	-67,85	-5,03	23,48	23,48
4	Апрель	-41,93	-81,11	59,70	-21,11	-21,11
5	Май	-33,20	-32,38	-121,56	-62,38	-62,38
6	Июнь	-136,46	-107,65	-108,83	-117,65	-117,65
7	Июль	-119,73	-50,91	-32,09	-67,58	-67,58
8	Август	-34,99	-94,18	-99,36	-76,18	-76,18
9	Сентябрь	3,74	0,56	-18,62	-4,77	-4,77
10	Октябрь	-13,52	95,29	96,11	59,29	59,29
11	Ноябрь	53,21	-45,97	-43,15	-11,97	-11,97
12	Декабрь	29,95	198,76	83,58	104,10	104,10

Расчет ошибки модели

№	Фактическое	Модель	Ошибка	Ср. кв. ошибка
1	352	476,22	-124,22	0,0680
2	492	387,55	104,45	0,0726
3	496	372,89	123,11	0,1090
4	314	331,55	-17,55	0,0028
5	326	293,55	32,45	0,0122
6	226	241,55	-15,55	0,0041
7	246	294,89	-48,89	0,0275
8	334	289,55	44,45	0,0236
9	376	364,22	11,78	0,0010
10	362	431,55	-69,55	0,0260
11	432	363,55	68,45	0,0354
12	412	512,14	-100,14	0,0382

Рисунок 3.24 – Розрахунок помилки і сезонної компоненти лінійної моделі

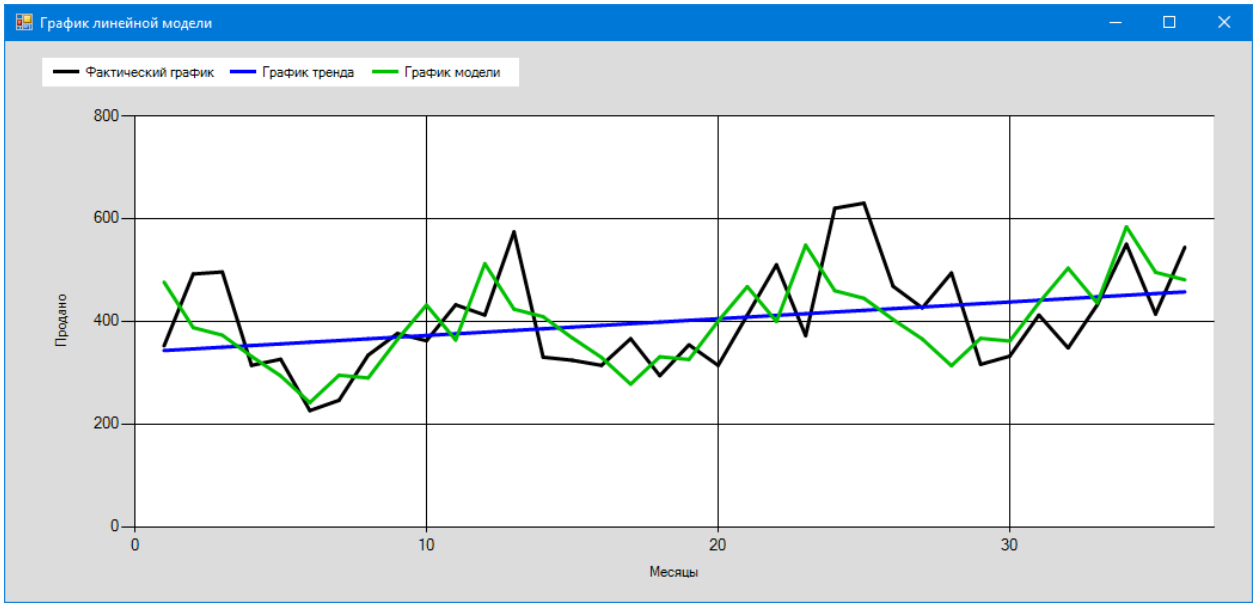


Рисунок 3.25 – Порівняльний графік лінійної моделі

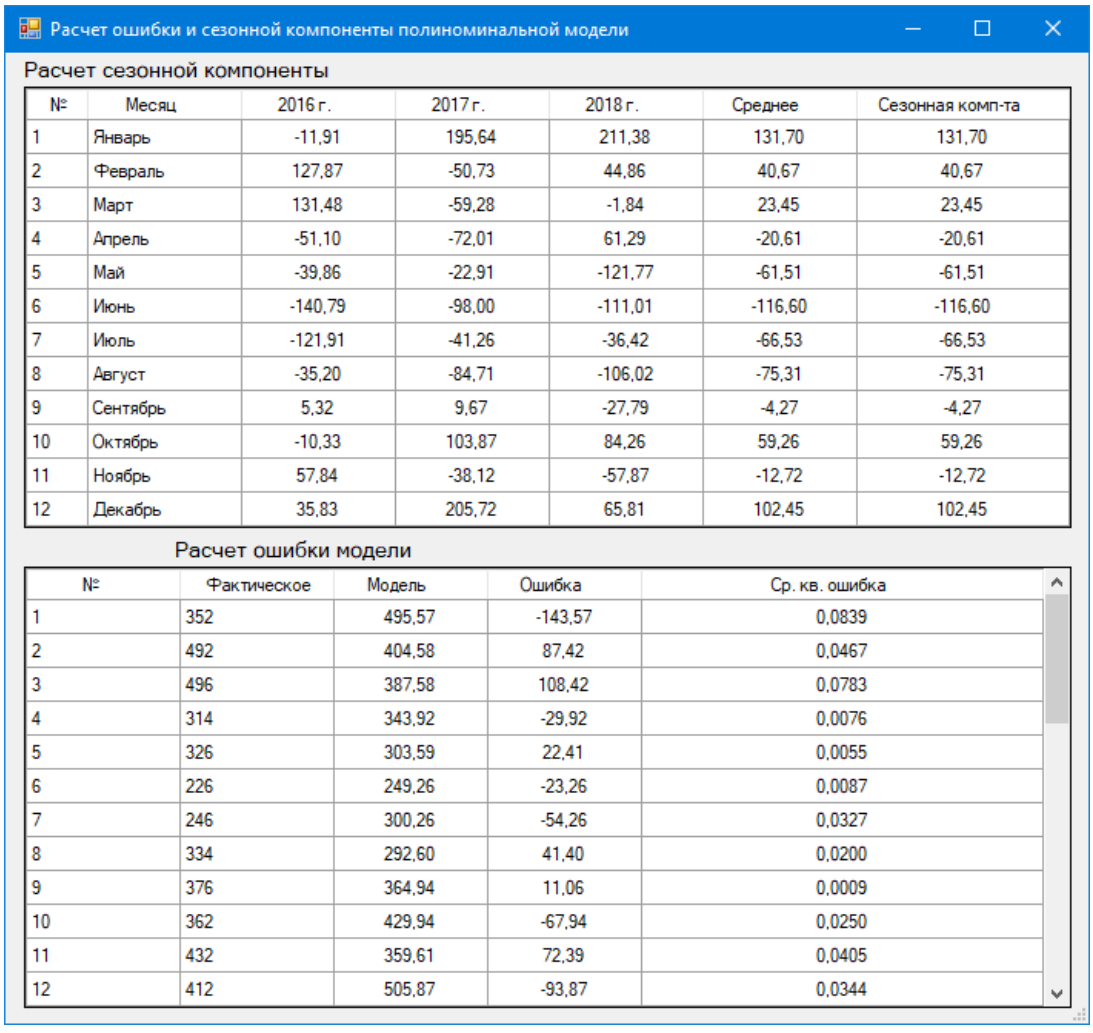


Рисунок 3.26 – Розрахункові значення помилки і сезонної компоненти поліноміальної моделі

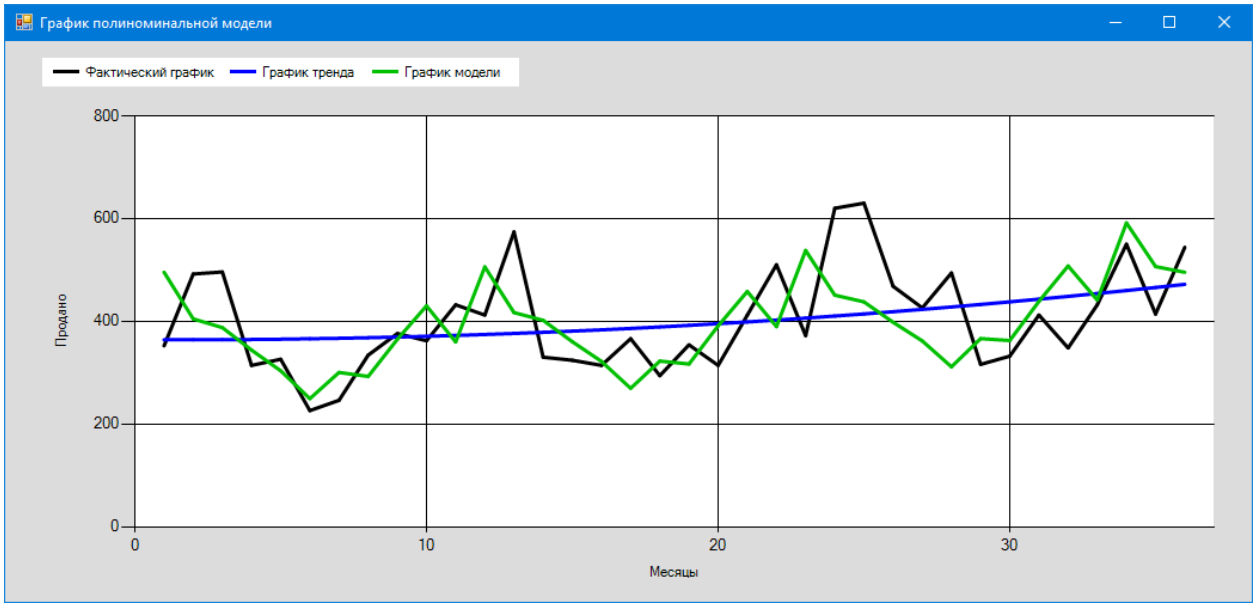


Рисунок 3.27 – Порівняльний графік поліноміальної моделі

На рисунку 3.28 представлений порівняльний графік прогнозованих значень, а також графік реальних продажів за поточний рік.

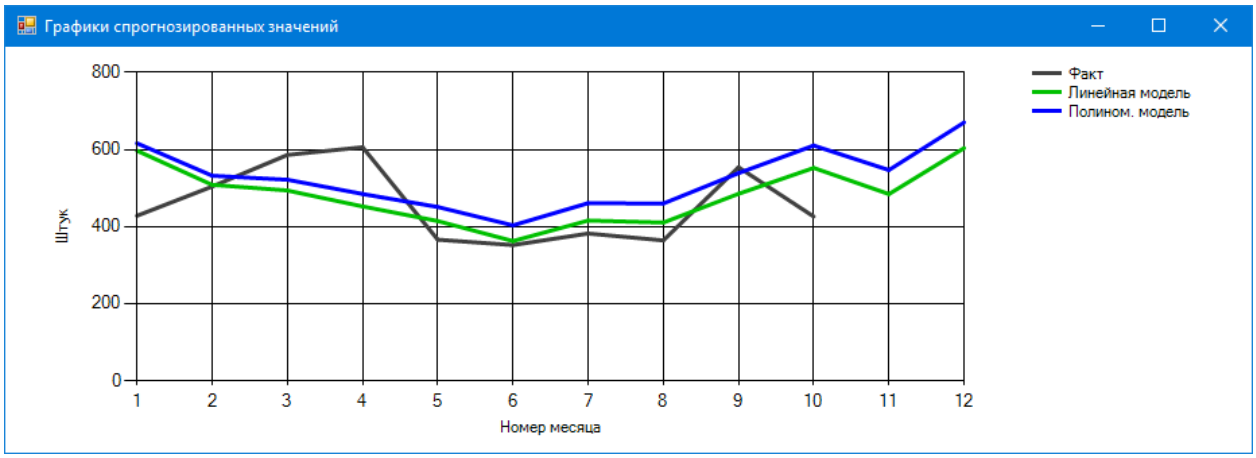


Рисунок 3.28 – Порівняльний графік прогнозованих значень, а також графік реальних продажів за поточний рік

3.4 Визначення показників метрик коду

Якість програмного забезпечення - це здатність програмного продукту при заданих умовах задовольняти встановленим або передбачуваним потребам. Якість коду оцінюється по набору критеріїв. Методами, які дозволяють оцінити якість програмного коду, є метрики програмного забезпечення (або метрики коду) [21].

Підвищена складність сучасних програмних додатків також підвищує складність забезпечення надійності та супроводу коду. Метрики коду представляють собою набір оцінок програмного забезпечення, які дають розробникам більш глибоке уявлення про розроблюваний коді. Використовуючи переваги метрик коду, розробники можуть зрозуміти, які типи і методи повинні бути перероблені або ретельно протестовані. Групи розробників можуть виявляти потенційні ризики, розуміння поточного стану проекту і відстеження ходу виконання під час розробки програмного забезпечення.Метрика програмного забезпечення – міра, що дозволяє отримати чисельне значення деякої властивості програмного забезпечення або його специфікацій.

Середовище розробки Microsoft Visual Studio має вбудовані інструменти для виконання автоматичного розрахунку метрик коду. Розглянемо метрики, які обчислює середовище розробки.

Індекс зручності підтримки - обчислює значення індексу від 0 до 100, яке представляє відносну простоту обслуговування коду. Високе значення означає кращу підтримку. Рейтинги з колірним кодуванням можна використовувати для швидкого виявлення проблемних місць в коді. Зелений колір становить від 20 до 100 і вказує на те, що код має гарну підтримкою. Жовтий рейтинг становить від 10 до 19 і вказує на те, що код є помірно підтримуваним. Червоний рейтинг - це рейтинг від 0 до 9 і вказує на низьку підтримку коду.

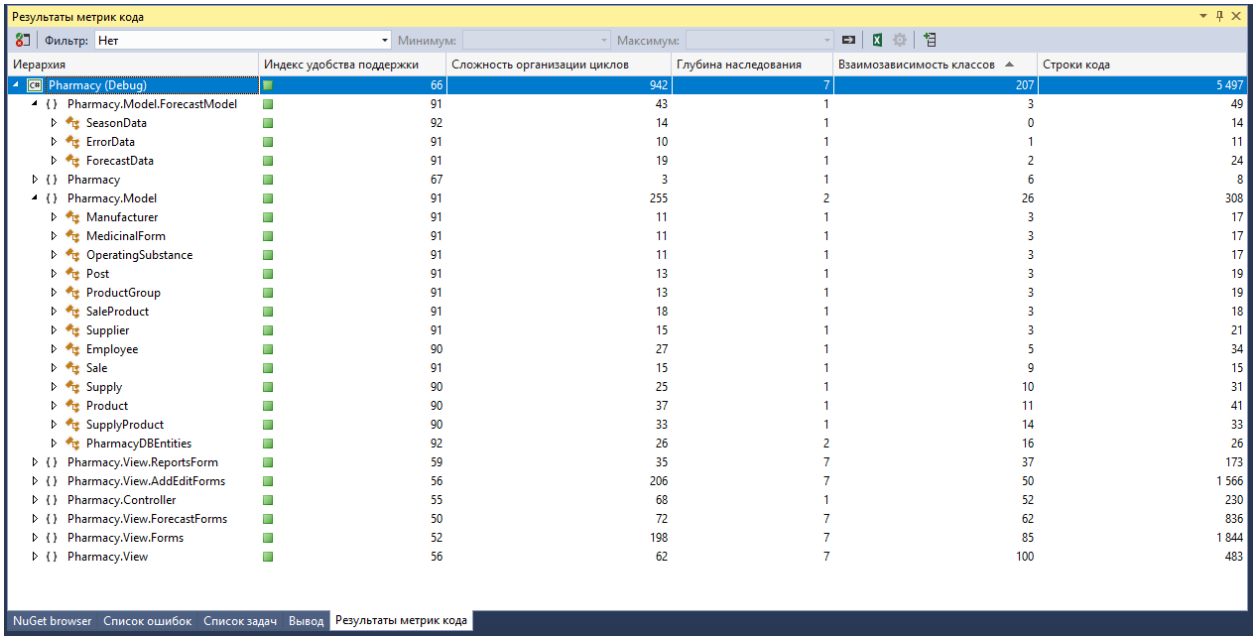
Складність організації циклів - вимірює структурну складність коду. Він створюється шляхом обчислення кількості розгалужень коду в потоці програми. Програма, яка має складний потік управління, вимагає більшої кількості тестів для досягнення хорошого покриття коду і менш зручна в обслуговуванні.

Глибина спадкування - вказує кількість різних класів, які успадковуються один від одного, аж до базового класу. Глибина успадкування схожа на зв'язування класів в тому, що зміна базового класу може вплинути на будь-який з його успадкованих класів. Чим вище це число, тим глибше успадкування і вище ймовірність того, що зміни базового класу приведуть до серйозних змін всієї програми. Для глибини успадкування низьке значення - це добре, а високе значення - погано.

Об'єднання класів - вимірює зв'язок з унікальними класами за допомогою параметрів, локальних змінних, які повернуться типів, викликів методів, узагальнених або шаблонних реалізацій, базових класів, реалізацій інтерфейсу, полів, певних для зовнішніх типів, і декорації атрибутів. Хороший програмний дизайн вимагає, щоб типи і методи мали низьку зв'язок. Висока зв'язок вказує на дизайн, який важко використовувати і підтримувати через безліч взаємозалежностей від інших типів.

Рядки коду - вказує приблизну кількість рядків в коді. Кількість засноване на коді IL і, отже, не є точним числом рядків у файлі вихідного коду. Велика кількість може вказувати на те, що тип або метод намагаються виконати занадто багато роботи і повинні бути розділені. Це також може вказувати на те, що тип або метод важко підтримувати.

Показники метрик коду для додатка співробітника представлені на рисунку 3.29.



Иерархия	Индекс удобства поддержки	Сложность организации циклов	Глубина наследования	Взаимозависимость классов	Строки кода
Pharmacy (Debug)	66	942	7	207	5 497
Pharmacy.Model.ForecastModel	91	43	1	3	49
SeasonData	92	14	1	0	14
ErrorData	91	10	1	1	11
ForecastData	91	19	1	2	24
Pharmacy	67	3	1	6	8
Pharmacy.Model	91	255	2	26	308
Manufacturer	91	11	1	3	17
MedicinalForm	91	11	1	3	17
OperatingSubstance	91	11	1	3	17
Post	91	13	1	3	19
ProductGroup	91	13	1	3	19
SaleProduct	91	18	1	3	18
Supplier	91	15	1	3	21
Employee	90	27	1	5	34
Sale	91	15	1	9	15
Supply	90	25	1	10	31
Product	90	37	1	11	41
SupplyProduct	90	33	1	14	33
PharmacyDBEntities	92	26	2	16	26
Pharmacy.View.ReportsForm	59	35	7	37	173
Pharmacy.View.AddEditForms	56	206	7	50	1 566
Pharmacy.Controller	55	68	1	52	230
Pharmacy.View.ForecastForms	50	72	7	62	836
Pharmacy.View.Forms	52	198	7	85	1 844
Pharmacy.View	56	62	7	100	483

Рисунок 3.29 – Показники метрик коду для додатка

Для зручності сприйняття середовище Visual Studio відображає колірну індикацію загального рівня якості коду. Зелена індикація для розробленого додатка свідчить про високу якість коду.

ВИСНОВКИ

В результаті виконання дипломної роботи були розглянуті основні методи прогнозування економічних процесів і проведено їх порівняльний аналіз. Для реалізації системи прогнозування були обрані засоби розробки, а також обґрунтований вибір методу прогнозування.

На підставі обраного методу був розроблений алгоритм побудови прогнозу та оцінки його помилки. Розроблений алгоритм був реалізований програмно і впроваджений в інформаційну систему аптеки.Програма показала хороші результати точності прогнозування та повністю відповідає поставленим вимогам.

Система відповідає висунутим до розробки вимогам, володіє інтерактивними елементами управління і інтуїтивно-зрозумілим інтерфейсом. Графічна оболонка має просту навігаційну структуру, що полегшує доступ до необхідної інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дуброва Т.А. Статистические методы прогнозирования: Учеб, пособие для вузов. М.: ЮНИТИ-ДАНА, 2003. 206 с.

2. Афанасьев, В.Н. Анализ временных рядов и прогнозирование: учебник/В.Н. Афанасьев, М.М. Юзбашев. М.: Финансы и статистика; ИНФРА-М, 2010.

3. Бутакова М.М. Экономическое прогнозирование: методы и приемы практических расчетов: учебное пособие/М.М. Бутакова, 2-ое изд., испр. М.: КНОРУС, 2010, 168 с.

4. Айвазян, С. А. Прикладная статистика и основы эконометрики : учебник для вузов/С. А. Айвазян, В. С. Мхита- рян. М.: ЮНИТИ, 1998.

5. Кондратьев, Н. Д. Проблемы экономической динамики/ред. коллегия: Л. И. Абалкин [и др.]. М. : Экономика, 1989.

6. Лагутин, М. В. Наглядная математическая статистика : учеб, пособие / М. В. Лагутин. М. : БИНОМ ; Лаборатория знаний, 2009.

7. Лапыгин, Ю. Н. Экономическое прогнозирование : учеб, пособие / Ю. Н. Лапыгин. М.: Эксмо, 2009.

8. Кошечкин С.А. Алгоритм прогнозирования объема продаж в MS Excol / С.А. Кошечкин // Маркетинг за рубежом. 2001. No 5. С. 34-42.

9. Троелсен, Эндрю. Язык программирования C# 5.0 и платформа .NET 4.5, 6–е изд. : Пер. с англ. М. : ООО “И.Д. Вильямс”, 2013. 1312 с., ил.

10. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. СПб.: Питер, 2013. 896 с.: ил. (Серия «Мастер-класс»).

11. IntelliSense в Visual Studio. URL: https://docs.microsoft.com/en-us/visualstudio/ide/using-intellisense?view =vs-2019. (Дата звернення 18.09.2019).

12. Сажин Ю.В. Анализ временных рядов и прогнозирование: учебник Ю. В. Сажин, А.В. Катынь, Ю.В. Сарайкин. – Саранск: Изд-во Мородов. Унта, 2013. – 192 с.

13. Чураков Е. П. Прогнозирование эконометрических рядов / Е. П. Чурков – М. : Финансы и статистика, 2008. – 208 с.

14. Диго С.М. Базы данных: проектирование и использование. Учебник. М.: Финансы и статистика, 2005.

15. Сибилёв, В.Д. Модели и проектирование баз данных: Учебное пособие. В 2-х частях. – Томск: Томский межвузовский центр дистанционного образования – 2002. – Ч.1. 133с.

16. Entity Framework 6 в Visual Studio [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/ef/ef6/. – Загл. с экрана. (дата обращения: 30.04.2019).

17. Маркин, Л. В. Программирование на SQL. В 2 ч. Часть 1: учебник и практикум для бакалавриата и магистратуры/А. В. Маркин. – М.: Издательство Юрайт, 2017. – 362 с. – Серия: Бакалавр и магистр. академический курс.

18. Гамма Э., Хелм R, Джонсон R, Влиссидес Дж. Приёмы объектно-ориентированного проектирования. Паттерны проектирования. - СПб.: Питер, 2015. - 368 с.: ил. - (Серия «Библиотека программиста»).

19. Бёрнс Б. Распределённые системы. Паттерны проектирования. – СПб. Питер, 2019. – 224 с.: ил. – (Серия «Бестселлеры O’Reilly»).

20. Скит Джон. C# для профессионалов: тонкости программирования, 3-е изд.: Пер. с англ. – М.: ООО “И.Д. Вильямс”, 2014. - 608 с. : ил. – Парал. тит. англ.

21. Обзор статического анализа кода для управляемого кода в Visual Studio. [Электронный ресурс]. - Режим доступа: https://docs.microsoft.com/en-us/visualstudio/code-quality/code-analysis-for-managed-code-overview?view=vs-2019 (дата обращения: 02.10.2019)

ДОДАТОК А

SQL скрипт створення таблиць БД зі зв'язками

```
/*==============================================================*/
/* Table: Employee                                              */
/*==============================================================*/
create table Employee (
   Employee_Id          int                  identity,
   Post_Id              int                  not null,
   FName                varchar(100)         not null,
   EmpName              varchar(100)         not null,
   LName                varchar(100)         not null,
   Phone                varchar(20)          not null,
   Login                varchar(150)         not null,
   Password             varchar(150)         not null,
   constraint PK_EMPLOYEE primary key (Employee_Id)
)
go

/*==============================================================*/
/* Index: Index_2                                               */
/*==============================================================*/
create unique nonclustered index Index_2 on Employee (Phone ASC)
go

/*==============================================================*/
/* Index: Index_3                                               */
/*==============================================================*/
create unique nonclustered index Index_3 on Employee (Login ASC)
go

/*==============================================================*/
/* Table: Manufacturer                                          */
/*==============================================================*/
create table Manufacturer (
   Manufacturer_Id      int                  identity,
   ManufacturerName     varchar(150)         not null,
   constraint PK_MANUFACTURER primary key (Manufacturer_Id)
)
go

/*==============================================================*/
/* Index: Index_2                                               */
/*==============================================================*/
create unique nonclustered index Index_2 on Manufacturer (ManufacturerName ASC)
go

/*==============================================================*/
/* Table: MedicinalForm                                         */
/*==============================================================*/
create table MedicinalForm (
   MedicinalForm_Id     int                  identity,
   MedicinalFormName    varchar(150)         not null,
   constraint PK_MEDICINALFORM primary key (MedicinalForm_Id)
)
go

/*==============================================================*/
/* Index: Index_2                                               */
/*==============================================================*/
create unique nonclustered index Index_2 on MedicinalForm (MedicinalFormName ASC)
```

```
go

/*==============================================================*/
/* Table: OperatingSubstance                                    */
/*==============================================================*/
create table OperatingSubstance (
   OperatingSubstance_Id int                  identity,
   OperatingSubstanceName varchar(200)         not null,
   constraint PK_OPERATINGSUBSTANCE primary key (OperatingSubstance_Id)
)
go

/*==============================================================*/
/* Index: Index_2                                               */
/*==============================================================*/
create unique nonclustered index Index_2 on OperatingSubstance (OperatingSubstanceName
ASC)
go

/*==============================================================*/
/* Table: Post                                                  */
/*==============================================================*/
create table Post (
   Post_Id              int                   identity,
   PostName             varchar(50)           not null,
   Details              varchar(2000)         null,
   constraint PK_POST primary key (Post_Id)
)
go

/*==============================================================*/
/* Index: Index_2                                               */
/*==============================================================*/
create unique nonclustered index Index_2 on Post (PostName ASC)
go

/*==============================================================*/
/* Table: Product                                               */
/*==============================================================*/
create table Product (
   Product_Id           int                   identity,
   ProductGroup_Id      int                   not null,
   Manufacturer_Id      int                   not null,
   MedicinalForm_Id     int                   not null,
   OperatingSubstance_Id int                   not null,
   ProductName          varchar(200)          not null,
   Code                 varchar(13)           not null,
   MarkupPercent        smallint              not null
      constraint CKC_MARKUPPERCENT_PRODUCT check (MarkupPercent between 0 and 100),
   constraint PK_PRODUCT primary key (Product_Id)
)
go

/*==============================================================*/
/* Index: Index_2                                               */
/*==============================================================*/
create unique nonclustered index Index_2 on Product (ProductName ASC)
go

/*==============================================================*/
/* Index: Index_3                                               */
/*==============================================================*/
create unique nonclustered index Index_3 on Product (Code ASC)
go
```

```
/*==============================================================*/
/* Table: ProductGroup                                          */
/*==============================================================*/
create table ProductGroup (
   ProductGroup_ID      int                  identity,
   ProductGroupName     varchar(100)         not null,
   Details              varchar(2000)        null,
   constraint PK_PRODUCTGROUP primary key (ProductGroup_ID)
)
go

/*==============================================================*/
/* Index: Index_2                                               */
/*==============================================================*/
create unique nonclustered index Index_2 on ProductGroup (ProductGroupName ASC)
go

/*==============================================================*/
/* Table: Sale                                                  */
/*==============================================================*/
create table Sale (
   Sale_Id              int                  identity,
   Employee_Id          int                  not null,
   SaleDate             datetime             not null,
   constraint PK_SALE primary key (Sale_Id)
)
go

/*==============================================================*/
/* Table: SaleProduct                                           */
/*==============================================================*/
create table SaleProduct (
   SaleProduct_Id       int                  identity,
   Sale_Id              int                  not null,
   SupplyProduct_Id     int                  not null,
   Quantity             int                  not null,
   constraint PK_SALEPRODUCT primary key (SaleProduct_Id),
   constraint AK_KEY_2_SALEPROD unique (Sale_Id, SupplyProduct_Id)
)
go

/*==============================================================*/
/* Index: Index_2                                               */
/*==============================================================*/
create nonclustered index Index_2 on SaleProduct (Sale_Id ASC)
go

/*==============================================================*/
/* Table: Supplier                                              */
/*==============================================================*/
create table Supplier (
   Supplier_Id          int                  identity,
   SupplierName         varchar(100)         not null,
   ContactPerson        varchar(150)         not null,
   Phone                varchar(20)          not null,
   constraint PK_SUPPLIER primary key (Supplier_Id)
)
go

/*==============================================================*/
/* Index: Index_2                                               */
/*==============================================================*/
create unique nonclustered index Index_2 on Supplier (SupplierName ASC)
go
```

```
/*==============================================================*/
/* Index: Index_3                                               */
/*==============================================================*/
create unique nonclustered index Index_3 on Supplier (Phone ASC)
go

/*==============================================================*/
/* Table: Supply                                                */
/*==============================================================*/
create table Supply (
   Supply_Id            int                  identity,
   Supplier_Id          int                  not null,
   Employee_Id          int                  not null,
   SupplyDate           datetime             not null,
   Complete             bit                  not null,
   constraint PK_SUPPLY primary key (Supply_Id)
)
go

/*==============================================================*/
/* Index: Index_2                                               */
/*==============================================================*/
create nonclustered index Index_2 on Supply (Supplier_Id ASC)
go

/*==============================================================*/
/* Index: Index_3                                               */
/*==============================================================*/
create nonclustered index Index_3 on Supply (Employee_Id ASC)
go

/*==============================================================*/
/* Table: SupplyProduct                                         */
/*==============================================================*/
create table SupplyProduct (
   SupplyProduct_Id     int                  identity,
   Supply_Id            int                  not null,
   Product_Id           int                  not null,
   Quantity             int                  not null
      constraint CKC_QUANTITY_SUPPLYPR check (Quantity >= 0),
   ExpDate              datetime             not null,
   Price                money                not null,
   constraint PK_SUPPLYPRODUCT primary key (SupplyProduct_Id)
)
go

/*==============================================================*/
/* Index: Index_Supply                                          */
/*==============================================================*/
create nonclustered index Index_Supply on SupplyProduct (Supply_Id ASC)
go

/*==============================================================*/
/* Index: Index_ProductUnique                                   */
/*==============================================================*/
create unique nonclustered index Index_ProductUnique on SupplyProduct (Product_Id ASC,
  Supply_Id ASC)
go

alter table Employee
   add constraint FK_EMPLOYEE_REFERENCE_POST foreign key (Post_Id)
      references Post (Post_Id)
go

alter table Product
```

```
   add constraint FK_PRODUCT_REFERENCE_PRODUCTG foreign key (ProductGroup_Id)
      references ProductGroup (ProductGroup_ID)
go

alter table Product
   add constraint FK_PRODUCT_REFERENCE_MANUFACT foreign key (Manufacturer_Id)
      references Manufacturer (Manufacturer_Id)
go

alter table Product
   add constraint FK_PRODUCT_REFERENCE_MEDICINA foreign key (MedicinalForm_Id)
      references MedicinalForm (MedicinalForm_Id)
go

alter table Product
   add constraint FK_PRODUCT_REFERENCE_OPERATIN foreign key (OperatingSubstance_Id)
      references OperatingSubstance (OperatingSubstance_Id)
go

alter table Sale
   add constraint FK_SALE_REFERENCE_EMPLOYEE foreign key (Employee_Id)
      references Employee (Employee_Id)
go

alter table SaleProduct
   add constraint FK_SALEPROD_REFERENCE_SUPPLYPR foreign key (SupplyProduct_Id)
      references SupplyProduct (SupplyProduct_Id)
go

alter table SaleProduct
   add constraint FK_SALEPROD_REFERENCE_SALE foreign key (Sale_Id)
      references Sale (Sale_Id)
         on update cascade on delete cascade
go

alter table Supply
   add constraint FK_SUPPLY_REFERENCE_EMPLOYEE foreign key (Employee_Id)
      references Employee (Employee_Id)
go

alter table Supply
   add constraint FK_SUPPLY_REFERENCE_SUPPLIER foreign key (Supplier_Id)
      references Supplier (Supplier_Id)
go

alter table SupplyProduct
   add constraint FK_SUPPLYPR_REFERENCE_PRODUCT foreign key (Product_Id)
      references Product (Product_Id)
go

alter table SupplyProduct
   add constraint FK_SUPPLYPR_REFERENCE_SUPPLY foreign key (Supply_Id)
      references Supply (Supply_Id)
         on update cascade on delete cascade
go
```

ДОДДАТОК Б

Код класу Repository.cs

```
public class Repository
    {
        public static Employee CurrentEmployee { get; set; }

        /// <summary>
        /// Метод возвращает данные для прогнозирования выбранного продукта
        /// Год, месяц,продано за месяц, название продукта
        /// </summary>
        /// <param name="supplyProduct"></param>
        /// <returns></returns>
        public List<ForecastData> GetForecastProductData(SupplyProduct
supplyProduct)
        {
            using (var context = new PharmacyDBEntities())
            {
                var query = "SELECT YEAR(Sale.SaleDate) AS Year," +
                            " MONTH(Sale.SaleDate) AS Month," +
                            " SUM(SaleProduct.Quantity) AS Count," +
                            " Product.ProductName AS Assortment " +
                            " FROM dbo.SaleProduct INNER JOIN dbo.Sale ON
SaleProduct.Sale_Id = Sale.Sale_Id " +
                            " INNER JOIN dbo.SupplyProduct ON
SaleProduct.SupplyProduct_Id = SupplyProduct.SupplyProduct_Id " +
                            " INNER JOIN dbo.Product ON
SupplyProduct.Product_Id = Product.Product_Id " +
                            " WHERE SupplyProduct.Product_Id =
"+supplyProduct.Product_Id+
                            " GROUP BY YEAR(Sale.SaleDate)
,MONTH(Sale.SaleDate) ,Product.ProductName ORDER BY Year";

                return
context.Database.SqlQuery<ForecastData>(query).ToList();
            }
        }
```

```
        /// <summary>
        /// Метод проверки первого запуска программы (база данных пуста, нет
админа)
        /// </summary>
        /// <returns></returns>
        public static bool FirstStart()
        {
            using (var context = new PharmacyDBEntities())
            {
                return !context.Employees.Any();
            }
        }

        /// <summary>
        /// Добавление сущности с обработкой исключения
        /// </summary>
        /// <typeparam name="T"></typeparam>
        /// <param name="entity"></param>
        public int AddEntityWithTryCatch<T>(T entity) where T : class
        {
            using (var context = new PharmacyDBEntities())
            {
                if (typeof(T).Name == "ProductGroup")
                {
                    var prodGr = entity as ProductGroup;
                    var result = context.ProductGroups.FirstOrDefault(group
=> group.ProductGroupName == prodGr.ProductGroupName);
                    if (result == null)
                    {
```

```
                        return (AddEntity(entity) as
ProductGroup).ProductGroup_ID;
                    }
                    return result.ProductGroup_ID;
                }
                if (typeof(T).Name == "Manufacturer")
                {
                    var item = entity as Manufacturer;
                    var result = context.Manufacturers.FirstOrDefault(ent =>
ent.ManufacturerName == item.ManufacturerName);
                    if (result == null)
                    {
                        return (AddEntity(entity) as
Manufacturer).Manufacturer_Id;
                    }
                    return result.Manufacturer_Id;
                }
                if (typeof(T).Name == "MedicinalForm")
                {
                    var item = entity as MedicinalForm;
                    var result = context.MedicinalForms.FirstOrDefault(ent =>
ent.MedicinalFormName == item.MedicinalFormName);
                    if (result == null)
                    {
                        return (AddEntity(entity) as
MedicinalForm).MedicinalForm_Id;
                    }
                    return result.MedicinalForm_Id;
                }
                if (typeof(T).Name == "OperatingSubstance")
                {
                    var item = entity as OperatingSubstance;
                    var result =
context.OperatingSubstances.FirstOrDefault(ent => ent.OperatingSubstanceName
== item.OperatingSubstanceName);
                    if (result == null)
                    {
                        return (AddEntity(entity) as
OperatingSubstance).OperatingSubstance_Id;
                    }
                    return result.OperatingSubstance_Id;
                }
                if (typeof(T).Name == "OperatingSubstance")
                {
                    var item = entity as OperatingSubstance;
                    var result =
context.OperatingSubstances.FirstOrDefault(ent => ent.OperatingSubstanceName
== item.OperatingSubstanceName);
                    if (result == null)
                    {
                        return (AddEntity(entity) as
OperatingSubstance).OperatingSubstance_Id;
                    }
                    return result.OperatingSubstance_Id;
                }
                return 0;
            }
        }
```

```
/// <summary>
        /// Метод создания изображения штрих-кода с помощью Barcode Library
        /// </summary>
        /// <param name="text"></param>
        /// <returns></returns>
        public static Image CreateBarcode(string text)
        {
            Barcode barcode = new Barcode()
            {
                IncludeLabel = true,
                Alignment = AlignmentPositions.CENTER,
                Width = 300,
                Height = 100,
                RotateFlipType = RotateFlipType.RotateNoneFlipNone,
```

```
                BackColor = Color.White,
                ForeColor = Color.Black,
            };
            try
            {
                return barcode.Encode(TYPE.EAN13, text);
            }
            catch (Exception)
            {
                MessageBox.Show(" Не найдена страна производителя!\nПроверьте
правильность введенного кода", "",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return null;
            }

        }

        /// <summary>
        /// Метод поиска поставки
        /// </summary>
        public List<Supply> SearchSupply(Supplier supplier, DateTime
supplyDate)
        {
            using (var context = new PharmacyDBEntities())
            {
                var query =
context.Supplies.Include("Supplier").Include("SupplyProducts").AsQueryable();
                context.Products.Load();
                if (supplier.Supplier_Id > 0)
                {
                    return query.Where(supply => supply.Supplier_Id ==
supplier.Supplier_Id).ToList().Where(supply => supply.SupplyDate.Date ==
supplyDate).ToList();
                }
                if (supplier.Supplier_Id == 0)
                {
                    return query.ToList().Where(supply =>
supply.SupplyDate.Date == supplyDate).ToList();
                }
                return query.ToList();
            }
        }
```

```
/// <summary>
        /// Метод поиска продуктов
        /// </summary>
        public List<Product> SearchProducts(string name, string code,
ProductGroup productGroup, OperatingSubstance substance)
        {
            using (var context = new PharmacyDBEntities())
            {
                context.Manufacturers.Load();
                context.MedicinalForms.Load();
                context.OperatingSubstances.Load();
                context.Supplies.Load();
context.SupplyProducts.Include("SaleProducts").Load();

                var query =
context.Products.Include("ProductGroup").Include("SupplyProducts").AsQueryabl
e();

                if (!string.IsNullOrWhiteSpace(name))
                {
                    query = query.Where(product =>
product.ProductName.ToLower().Contains(name));
                }

                if (!string.IsNullOrWhiteSpace(code))
                {
                    query = query.Where(product =>
product.Code.ToLower().Contains(code));
                }
```

```
                if (productGroup.ProductGroup_ID > 0)
                {
                    query = query.Where(product => product.ProductGroup_Id ==
productGroup.ProductGroup_ID);
                }

                if (substance.OperatingSubstance_Id > 0)
                {
                    query = query.Where(product =>
product.OperatingSubstance_Id == substance.OperatingSubstance_Id);
                }


                return query.ToList();
            }
        }

        /// <summary>
        /// Метод обновления сущности
        /// </summary>
        /// <typeparam name="T">Тип сущности</typeparam>
        /// <param name="entity">Объект сущности</param>
        /// <returns></returns>
        public bool UpdateEntity<T>(T entity) where T : class
        {
            using (var context = new PharmacyDBEntities())
            {
                context.Set<T>().Add(entity);
                var entry = context.Entry(entity);
                entry.State = EntityState.Modified;
                return context.SaveChanges() > 0;
            }
        }
```

```
/// <summary>
        /// Метод удаления сущности
        /// </summary>
        /// <typeparam name="T">Тип сущности</typeparam>
        /// <param name="entity">Объект удаляемой сущности</param>
        /// <param name="predicat">Условие поиска сущности в БД</param>
        /// <returns></returns>
        public bool RemoveEntity<T>(T entity, Func<T, bool> predicat) where T
: class
        {
            using (var context = new PharmacyDBEntities())
            {
                var ent = context.Set<T>().FirstOrDefault(predicat);
                if (ent == null) return false;
                context.Set<T>().Remove(ent);
                return context.SaveChanges() > 0;
            }
        }

        /// <summary>
        /// Метод удаления сущности
        /// </summary>
        /// <typeparam name="T">Тип сущности</typeparam>
        /// <param name="predicat">Условие поиска сущности в БД</param>
        /// <param name="predicate"></param>
        /// <returns></returns>
        public T GetEntity<T>(Expression<Func<T, bool>> predicate) where T :
class
        {
            using (var context = new PharmacyDBEntities())
            {
                if (typeof(T).Name == "SupplyProduct")
                {
                    context.Manufacturers.Load();
                    context.OperatingSubstances.Load();
                    context.ProductGroups.Load();
                    context.MedicinalForms.Load();
                    context.Products.Include("SupplyProducts").Load();
```

```
                    return
context.Set<T>().Include("SaleProducts").Include("Product").Include("Supply")
.FirstOrDefault(predicate);
                }
                return context.Set<T>().FirstOrDefault(predicate);
            }
        }

        /// <summary>
        /// Метод выбора коллекции сущностей по указнному условию предиката
        /// </summary>
        public List<T> GetEntityes<T>(Func<T, bool> predicate) where T :
class
        {
            using (var context = new PharmacyDBEntities())
            {

                if (typeof(T).Name == "SupplyProduct")
                {
                    context.Manufacturers.Load();
                    context.OperatingSubstances.Load();
                    context.ProductGroups.Load();
                    context.MedicinalForms.Load();
                    context.Products.Include("SupplyProducts").Load();
                    return
context.Set<T>().Include("SaleProducts").Include("Product").Include("Supply")
.Where(predicate).ToList();
                }
                if (typeof(T).Name == "SaleProduct")
                {

context.Products.Include("ProductGroup").Include("Manufacturer").Include("Sup
plyProducts").Load();

context.SupplyProducts.Include("Product").Include("SaleProducts").Load();
                    return
context.Set<T>().Include("SupplyProduct").Include("Sale").Where(predicate).To
List();
                }
                if (typeof(T).Name == "Sale")
                {
                    context.Supplies.Load();
                    context.Employees.Load();
                    context.SupplyProducts.Include("SaleProducts").Load();

context.Products.Include("ProductGroup").Include("SupplyProducts").Load();
                    context.ProductGroups.Load();
                    return
context.Set<T>().Include("SaleProducts").Where(predicate).ToList();
                }
                if (typeof(T).Name == "Product")
                {
                    context.Manufacturers.Load();
                    context.MedicinalForms.Load();
                    context.OperatingSubstances.Load();
                    context.Supplies.Load();
                    context.SupplyProducts.Include("SaleProducts").Load();
                    return
context.Set<T>().Include("ProductGroup").Include("SupplyProducts").Where(pred
icate).ToList();
                }
                if (typeof(T).Name == "TestParam")
                {
                    return
context.Set<T>().Include("Param").Where(predicate).ToList();
                }
                return context.Set<T>().Where(predicate).ToList();
            }
        }
```

```
/// <summary>
        /// Метод выбора коллекции сущностей
        /// </summary>
```

```
        public List<T> GetEntityes<T>() where T : class
        {
            using (var context = new PharmacyDBEntities())
            {
                if (typeof(T).Name == "SaleProduct")
                {

context.Products.Include("ProductGroup").Include("Manufacturer").Include("Sup
plyProducts").Load();

context.SupplyProducts.Include("Product").Include("SaleProducts").Load();

                    return
context.Set<T>().Include("SupplyProduct").Include("Sale").ToList();
                }
                if (typeof(T).Name == "SupplyProduct")
                {
                    context.Manufacturers.Load();
                    context.OperatingSubstances.Load();
                    context.ProductGroups.Load();
                    context.MedicinalForms.Load();
                    context.Products.Include("SupplyProducts").Load();
                    return
context.Set<T>().Include("SaleProducts").Include("Product").Include("Supply")
.ToList();
                }
                if (typeof(T).Name == "Supply")
                {
                    context.Products.Load();
                    return
context.Set<T>().Include("Supplier").Include("SupplyProducts").ToList();
                }
                if (typeof(T).Name == "Sale")
                {
                    context.Employees.Load();

context.SupplyProducts.Include("SaleProducts").Include("Product").Include("Su
pply").Load();
                    //context.Products.Load();
                    return context.Set<T>().Include("SaleProducts").ToList();
                }
                if (typeof(T).Name == "Employee")
                {
                    return context.Set<T>().Include("Post").ToList();
                }
                if (typeof(T).Name == "Product")
                {
                    context.Manufacturers.Load();
                    context.MedicinalForms.Load();
                    context.OperatingSubstances.Load();
                    context.Supplies.Load();
                    context.SupplyProducts.Include("SaleProducts").Load();
                    return
context.Set<T>().Include("ProductGroup").Include("SupplyProducts").ToList();
                }
                return context.Set<T>().ToList();
            }
        }

        /// <summary>
        /// Метод добавления сущности в БД
        /// </summary>
        /// <typeparam name="T">Тип сущности</typeparam>
        /// <param name="entity">Объект сущности</param>
        public T AddEntity<T>(T entity) where T : class
        {
            using (var context = new PharmacyDBEntities())
            {
                var adedEntity = context.Set<T>().Add(entity);
                context.SaveChanges();
                return adedEntity;
            }
```

```
        }

        /// <summary>
        /// Метод обработки исключений
        /// </summary>
        public void HandleException(Exception exception)
        {
            var sb = new StringBuilder();
            var validationException = exception as
DbEntityValidationException;
            if (validationException != null)
            {
                foreach (var dbEntityValidationResult in
validationException.EntityValidationErrors)
                {
                    foreach (var dbValidationError in
dbEntityValidationResult.ValidationErrors)
                    {
                        sb.AppendLine(dbValidationError.ErrorMessage);
                    }
                }
            }
            var innerEx = exception.InnerException ?? exception;
            while (true)
            {
                if (innerEx.InnerException == null)
                {
                    break;
                }
                innerEx = innerEx.InnerException;
            }
            MessageBox.Show(innerEx.Message + "\n" + sb + "\n" + "Исключение
получено в методе:\n" + exception.TargetSite + "\n Последний вызов:\n" +
exception.StackTrace.Split('\\').First(), "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
```

```
/// <summary>
        /// Метод для вход сотрудника в систему
        /// </summary>
        /// <param name="login">Логин</param>
        /// <param name="password">Пароль</param>
        /// <returns>true - сотрудник найден, false - такого логина нет в
БД</returns>
        public bool UserLogin(string login, string password)
        {
            using (var context = new PharmacyDBEntities())
            {
                CurrentEmployee =
context.Employees.Include("Post").FirstOrDefault(employee => employee.Login
== login && employee.Password == password);
                return CurrentEmployee != null;
            }
        }
    }
```
