НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет/(ННІ) _інформаційних технологій_

ПОГОДЖЕНО
Декан факультету (Директор ННІ)
_____інформаційних технологій_____
(назва факультету (ННІ))
_____ ___Ігор БОЛБОТ_____
(підпис) (ім'я ПРІЗВИЩЕ)
"___" _____ 2025 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ
Завідувач кафедри
_____комп'ютерних наук_____
(назва кафедри)
_____ __Белла ГОЛУБ____
(підпис) (ім'я ПРІЗВИЩЕ)
"___" _____ 2025 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему «Програмне забезпечення інформаційно-аналітичної системи для медичного персоналу поліклініки»

Спеціальність _121 – «Інженерія програмного забезпечення»_____
(код і найменування)

Освітня програма __Програмне забезпечення інформаційних систем_____
(назва)

Орієнтація освітньої програми___освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми
_____доц.к.ф.-м.н._____ _____ ____Віктор КИРИЧЕНКО_____
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи
_____к.ек.н., ст. викладач_____ _____ ____Дмитро НІКОЛАЄНКО_____
(науковий ступінь та вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)

Виконав _____ _____Владислав ВОЛОЧАЙ_____
(підпис) (ім'я ПРІЗВИЩЕ здобувача)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) __інформаційних технологій_____

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____компю'терних наук_____
_____доцент, к.т.н.____ _____ _Белла ГОЛУБ
(науковий ступінь, вчене звання) (підпис) (ім'я ПРІЗВИЩЕ)
"__1__" ____листопада_____2024року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

_____Волочаю Владиславу Євгенійовичу_____
(прізвище, ім'я, по батькові)

Спеціальність _121 – «Інженерія програмного забезпечення»_____
(код і найменування)

Освітня програма _Програмне забезпечення інформаційних систем_____
(назва)

Орієнтація освітньої програми__освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи___Програмне забезпечення інформаційно-аналітичної системи для медичного персоналу поліклініки_____
затверджена наказом від "_1__"__листопада__2024р. №1963 «С»

Термін подання завершеної роботи на кафедру __20.11.2025_____
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи _набори синтетичних даних, згенерованих для імітації роботи медичної інформаційно-аналітичної системи_____

Перелік питань, що підлягають дослідженню:

1. Аналіз ефективності роботи поліклініки, дослідження того як програмне забезпечення може оптимізувати робочі процеси.
2. Дослідження можливості застосування аналітичних методів.
3. Вплив автоматизації на якість медичного обслуговування, аналіз того як система сприяє підвищенню точності діагностики та ефективності лікування.
4. Аналіз взаємозв'язків між діагнозами, спеціальностями лікарів та даними пацієнтів для виявлення закономірностей.

Перелік графічного матеріалу (за потреби) _____

Дата видачі завдання "__1___"___листопада_____2024 р.

Керівник магістерської кваліфікаційної роботи _____ _Дмитро НІКОЛАЄНКО_
(підпис) (ім'я ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ _____Владислав ВОЛОЧАЙ_____
(підпис) (ім'я ПРІЗВИЩЕ)

ЗМІСТ

Перелік умовних позначень

OLAP - Online Analytical Processing (Оперативна аналітична обробка).

KPI - Key Performance Indicator (Ключовий показник ефективності).

EHR - Electronic Health Record (Електронний Запис Здоров'я).

EMR - Electronic Medical Records (Електронний Медичний Запис).

HIS - Hospital Information System (Госпітальна інформаційна система).

HMS - Healthcare Management System (Система управління охороною здоров'я).

БД - база даних.

ВСТУП

Сучасні медичні заклади функціонують у середовищі з високим рівнем інформаційного навантаження, де щодня формується велика кількість даних про пацієнтів, лікарів, захворювання, лікувальні послуги та результати прийомів. Ефективне управління цими даними є ключовим чинником забезпечення якості медичного обслуговування, проте ручна обробка інформації призводить до значних часових витрат, помилок і ускладнює прийняття оперативних управлінських рішень. У зв’язку з цим зростає потреба у створенні сучасних інформаційно-аналітичних систем, які здатні не лише зберігати дані, але й виконувати інтелектуальний аналіз з метою підтримки діяльності медичного персоналу.

Розвиток технологій обробки даних, зокрема систем OLAP, Data Mining і Machine Learning, відкриває нові можливості для побудови інтелектуальних медичних рішень. Такі підходи дозволяють здійснювати автоматизований пошук закономірностей у медичних записах, прогнозувати кількість звернень пацієнтів, оцінювати ефективність роботи лікарів, а також виявляти асоціативні зв’язки між типами послуг, діагнозами та характеристиками пацієнтів. Використання цих методів у рамках єдиної інформаційно-аналітичної системи підвищує продуктивність роботи лікарів і точність прийняття клінічних рішень.

Актуальність дослідження полягає у необхідності впровадження інтегрованих рішень для управління медичною інформацією на основі сучасних аналітичних технологій. Така система дає змогу підвищити ефективність діяльності поліклініки, забезпечити достовірність статистичних показників, оптимізувати навантаження на лікарів і покращити якість обслуговування пацієнтів.

Об’єкт дослідження — процеси інформаційно-аналітичної діяльності медичного персоналу поліклінічного закладу.

Предмет дослідження — методи, технології та програмні засоби побудови аналітичних інформаційних систем у сфері охорони здоров’я.

Мета дослідження — розроблення програмного забезпечення інформаційно-аналітичної системи, яка забезпечує збір, обробку, аналіз і візуалізацію медичних даних для підвищення ефективності роботи персоналу поліклініки.

Завдання. Для досягнення поставленої мети необхідно було вирішити такі завдання:

1. Провести аналіз сучасних медичних інформаційних систем і визначити їх переваги та недоліки.
2. Спроектувати архітектуру інформаційно-аналітичної системи для поліклініки.
3. Розробити базу даних і забезпечити її наповнення структурованими медичними записами.
4. Реалізувати модулі аналітики, що використовують алгоритми Time Series, 1-Rule, Naive Bayes, Apriori та K-Means.
5. Розробити систему оцінки ефективності роботи лікарів на основі KPI-показників.
6. Провести експериментальний аналіз отриманих результатів і оцінити ефективність впровадження системи.

Методи дослідження. У роботі використано методи аналізу та синтезу інформаційних систем, технології обробки даних OLAP, алгоритми інтелектуального аналізу даних (Data Mining), машинного навчання (Machine Learning), а також програмну реалізацію засобами C# у середовищі Microsoft Visual Studio та Microsoft SQL Server для роботи з базою даних.

Наукова новизна полягає в тому, що вперше розроблено інтегрований аналітичний модуль для медичної інформаційної системи, який поєднує методи прогнозування, класифікації, кластеризації та асоціативного аналізу в єдиному середовищі. Запропоновано удосконалення архітектури системи шляхом додавання

аналітичного блоку з підтримкою OLAP-запитів і динамічної візуалізації результатів.

Апробація результатів дослідження. Основні результати магістерської роботи були оприлюднені на XVI Міжнародній науково-практичній конференції молодих вчених «Інформаційні технології: економіка, техніка, освіта». Отримані результати мають практичне значення та можуть бути використані для подальших досліджень у сфері розробки інформаційно-аналітичних систем.

Структура роботи. Магістерська робота складається зі вступу, п’яти розділів, висновків, списку використаних джерел і додатків.

- У першому розділі наведено аналіз предметної області, описано проблеми автоматизації медичних установ і проведено огляд існуючих інформаційних систем.
- Другий розділ присвячено проектуванню архітектури системи та структури бази даних.
- У третьому розділі розглянуто реалізацію функціональних модулів системи.
- У четвертому розділі подано опис розробленого аналітичного модуля з реалізацією методів Data Mining.
- У п’ятому розділі проведено експериментальні дослідження, оцінку ефективності системи та інтерпретацію отриманих результатів.

Загальний обсяг роботи становить близько 118 сторінок, містить 40 рисунків та 2 додатка.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ.

1.1. Аналіз предметної області.

Медичні установи, зокрема поліклініки, є складними організаційними системами, у межах яких постійно відбувається взаємодія пацієнтів, лікарів, адміністративного персоналу та численних інформаційних потоків. У процесі надання медичних послуг формується велика кількість даних — про пацієнтів, лікарів, діагнози, захворювання, призначення, лікування, обстеження, рецепти, направлення та інші складові медичного процесу. З кожним роком обсяг цих даних зростає, що ускладнює їх зберігання, обробку й аналіз традиційними методами.

Проблема полягає у тому, що більшість медичних закладів досі використовують або застарілі, або фрагментовані інформаційні системи, які не забезпечують комплексної аналітичної підтримки. Такі системи, як правило, виконують лише реєстраційно-облікові функції — ведення електронних карток пацієнтів, облік прийомів чи звітність щодо наданих послуг. Проте відсутність аналітичних механізмів не дозволяє виявляти приховані закономірності у роботі поліклініки, прогнозувати динаміку звернень, оцінювати ефективність роботи лікарів або своєчасно виявляти проблемні тенденції.

У сучасних умовах цифрової трансформації медицини зростає потреба у створенні інтегрованих аналітичних платформ, що поєднують функції збору, зберігання, обробки та інтелектуального аналізу даних. Такі системи дозволяють не лише автоматизувати рутинні процеси, а й надавати управлінцям та медичним працівникам інструменти для підтримки прийняття рішень на основі даних.

Одним із ключових напрямів розвитку є застосування технологій OLAP, Data Mining та Machine Learning, які забезпечують багатовимірний аналіз, побудову прогнозних моделей, класифікацію пацієнтів за різними ознаками, а також

виявлення асоціативних зв’язків між діагнозами, послугами й характеристиками пацієнтів. Це відкриває можливості для підвищення точності діагностики, оптимізації завантаження лікарів і покращення якості медичного обслуговування.

Отже, у межах дослідження актуальним є створення інформаційно-аналітичної системи для поліклінічного закладу, яка інтегрує базу даних, аналітичні методи та засоби візуалізації результатів. Такий підхід спрямований на перехід від простого обліку даних до глибокого аналізу медичної інформації, що є важливим кроком у розвитку сучасних медичних інформаційних технологій.

1.2. Проблеми обробки та аналізу медичних даних.

Сучасні медичні заклади накопичують великі обсяги даних про пацієнтів, лікарів, діагнози, лікування, послуги та результати прийомів. Проте більшість інформаційних систем, які використовуються в поліклініках, обмежуються лише зберіганням цієї інформації без глибокого аналітичного опрацювання. Це створює низку проблем, що безпосередньо впливають на якість управлінських рішень і ефективність роботи медичного персоналу.

Основними проблемами при обробці та аналізу медичних даних є:

1. **Відсутність аналітичних інструментів для прийняття рішень.**

   У більшості існуючих медичних систем відсутні засоби для аналізу закономірностей у роботі поліклініки, виявлення факторів, що впливають на ефективність лікування або навантаження лікарів. Це обмежує можливість використання даних для підтримки управлінських і клінічних рішень.

2. **Низький рівень автоматизації аналітичних процесів.**

   Медичний персонал часто змушений виконувати аналіз вручну — формувати звіти, підраховувати показники, порівнювати дані за періодами. Це призводить до помилок і значних часових витрат.

3. **Складність візуалізації результатів аналізу.**

   Навіть якщо дані обробляються, вони часто подаються у вигляді таблиць або звітів, що ускладнює сприйняття результатів. Відсутність зручних графічних інтерфейсів знижує ефективність аналітичної роботи.

4. **Відсутність засобів прогнозного аналізу.**

   Для ефективного управління діяльністю поліклініки важливо не лише фіксувати поточний стан, а й передбачати тенденції — динаміку відвідуваності, зміну потреб пацієнтів, навантаження лікарів. Однак більшість систем не мають механізмів прогнозування.

5. **Відсутність оцінки ефективності роботи медичного персоналу.**

   У системах часто не реалізовані інструменти для розрахунку показників ефективності — KPI завантаженості лікарів, інтенсивності прийомів, динаміки притоку пацієнтів. Це унеможливлює об’єктивну оцінку діяльності персоналу.

6. **Недостатнє використання технологій інтелектуального аналізу даних.**

   Алгоритми класифікації, кластеризації, асоціативного аналізу та прогнозування практично не застосовуються в існуючих рішеннях, хоча саме вони дозволяють отримати нові знання з накопичених медичних даних.

Отже, основна проблема сучасних поліклінічних систем полягає не у відсутності даних, а у неможливості їх ефективного використання для аналітики та прогнозування.

Вирішення цих проблем можливе завдяки впровадженню технологій OLAP (Online Analytical Processing), Data Mining (інтелектуальний аналіз даних) і Machine Learning (машинне навчання), які забезпечують глибоку аналітичну обробку медичних даних і автоматичне виявлення закономірностей.

1.3. Аналіз існуючих систем.

Сучасний ринок програмного забезпечення для медичних закладів представлений великою кількістю систем, які вирішують задачі обліку, планування та управління медичною діяльністю. Проте більшість із них зосереджені переважно на операційному рівні — реєстрації пацієнтів, веденні медичних карток, фіксації призначень і звітності, тоді як аналітична складова залишається обмеженою.

1.3.1. Типи медичних інформаційних систем

За функціональним призначенням медичні інформаційні системи можна поділити на такі категорії:

- **Клінічні системи (EHR/EMR)** — системи електронних медичних записів (Electronic Health Record, Electronic Medical Record), що забезпечують збереження історій хвороб, результатів обстежень, лабораторних досліджень, призначень лікарів.
- **Адміністративно-управлінські системи (HIS, HMS)** — рішення, орієнтовані на управління персоналом, фінансами, розкладом прийомів і ресурсами клініки.
- **Аналітичні та статистичні модулі** — додатки, інтегровані до основних систем, які забезпечують формування звітів, але здебільшого не реалізують повноцінний інтелектуальний аналіз.
- **Комплексні медичні платформи** — інтегровані рішення, які поєднують клінічні, адміністративні й аналітичні функції, проте їх вартість і вимоги до інфраструктури часто перевищують можливості поліклінічних закладів.

1.3.2. Огляд типових систем

По закінченню аналізу існуючих систем було виділено чотири системи для проведення огляду.

Перша система яка буде розглянута це Doctor Eleks на Рис. 1.1:

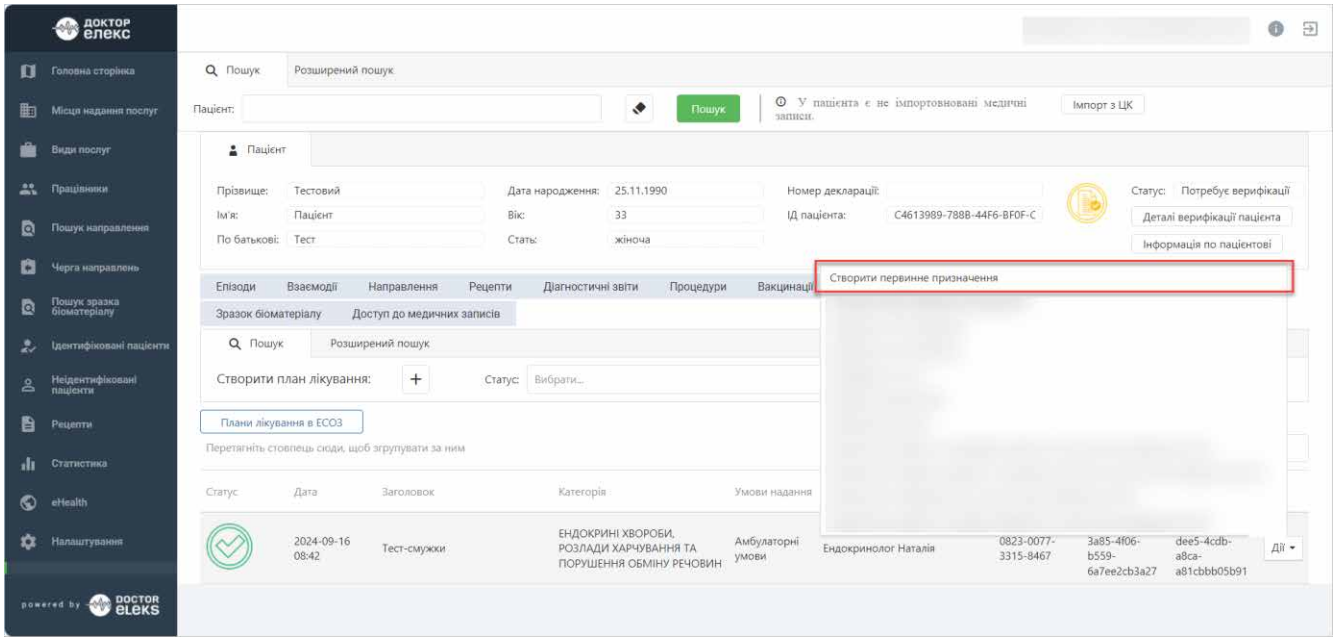

Рис. 1.1. – Інтерфейс Doctor Eleks.

Програмне забезпечення Doctor є українською медичною інформаційною системою, яка призначена для автоматизації діяльності лікувально-профілактичних закладів. Вона підтримує ведення електронних медичних карток, розкладів лікарів, управління чергами, облік лабораторних досліджень і формування звітності. Система відзначається модульною архітектурою та високим рівнем локалізації під потреби українських клінік. Однак її аналітичні можливості обмежуються стандартними статистичними звітами, без використання методів прогнозування або виявлення закономірностей у медичних даних.

Друга система під розгляд є Meditech, зображення на Рис. 1.2:

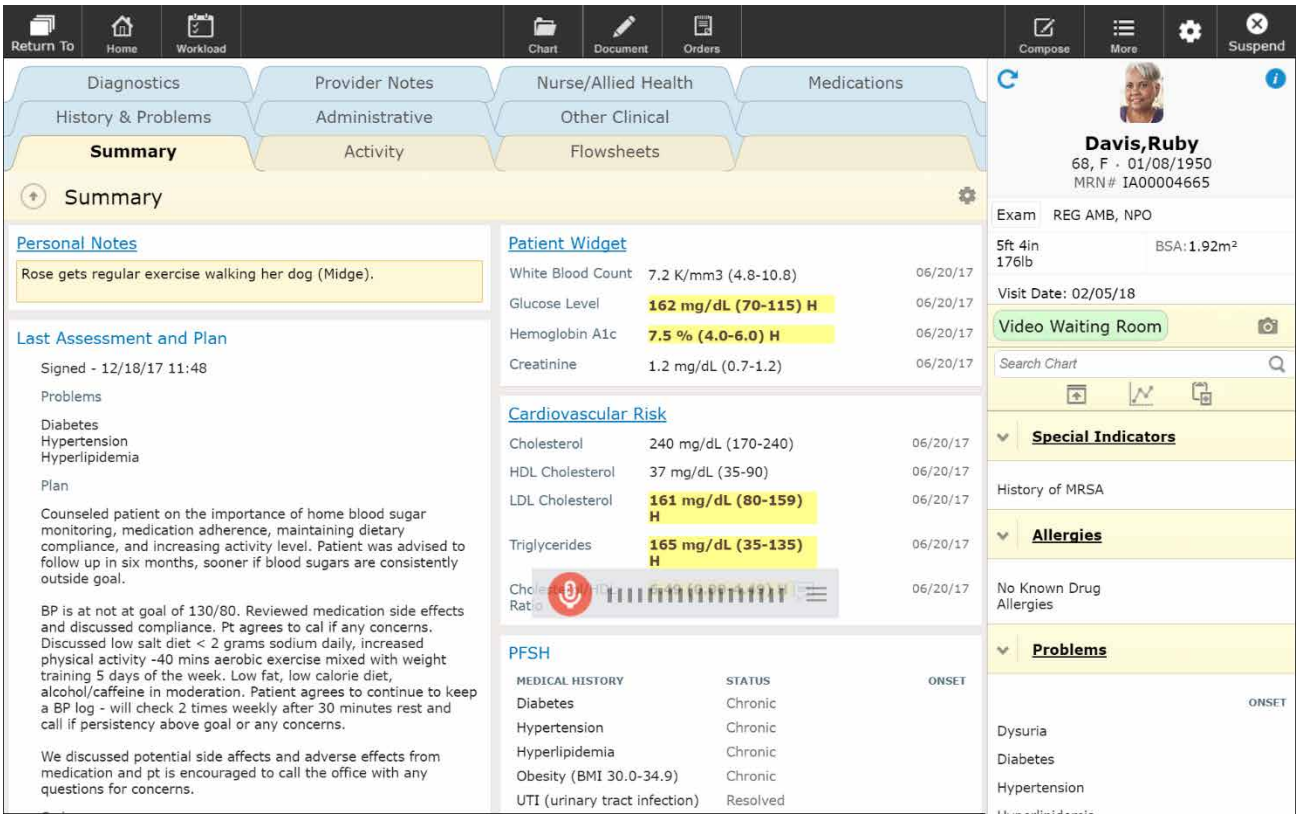

Рис. 1.2. – Інтерфейс Meditech.

Meditech — одна з поширених комерційних систем електронної медичної документації (EHR), що використовується в клініках та лікарнях різних країн. Вона забезпечує інтеграцію клінічних, адміністративних і фінансових даних, підтримує ведення історій хвороб, управління потоками пацієнтів, лабораторними дослідженнями та розрахунками. Система має вбудовані засоби звітності та базової аналітики, однак розширені можливості аналізу даних часто потребують додаткових модулів або інтеграції з зовнішніми BI-рішеннями.

Третьою системою на розгляд є OpenMRS, на Рис. 1.3:

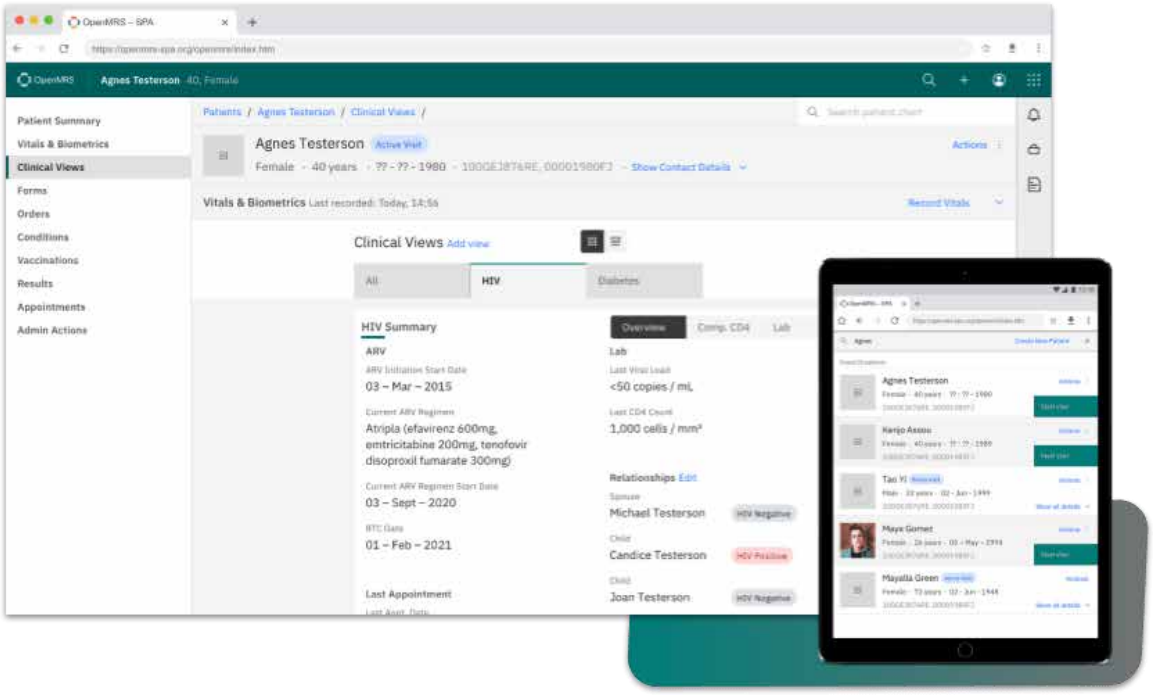

Рис. 1.3. – Інтерфейс OpenMRS.

Програмне забезпечення OpenMRS(Open Medical Record System) — відкрита (open-source) система управління медичними записами, яка широко використовується у навчальних, дослідницьких та міжнародних проєктах, особливо в умовах обмежених ресурсів. Вона дозволяє гнучко налаштовувати структуру даних, модулі форми введення, а також розширювати функціонал через власні плагіни. Водночас реалізація повноцінного аналітичного модуля в OpenMRS потребує значних зусиль з боку розробників, оскільки механізми Data Mining не входять до базового ядра системи.

Останньою розглянутою системою буде CareCloud на Рис. 1.4:

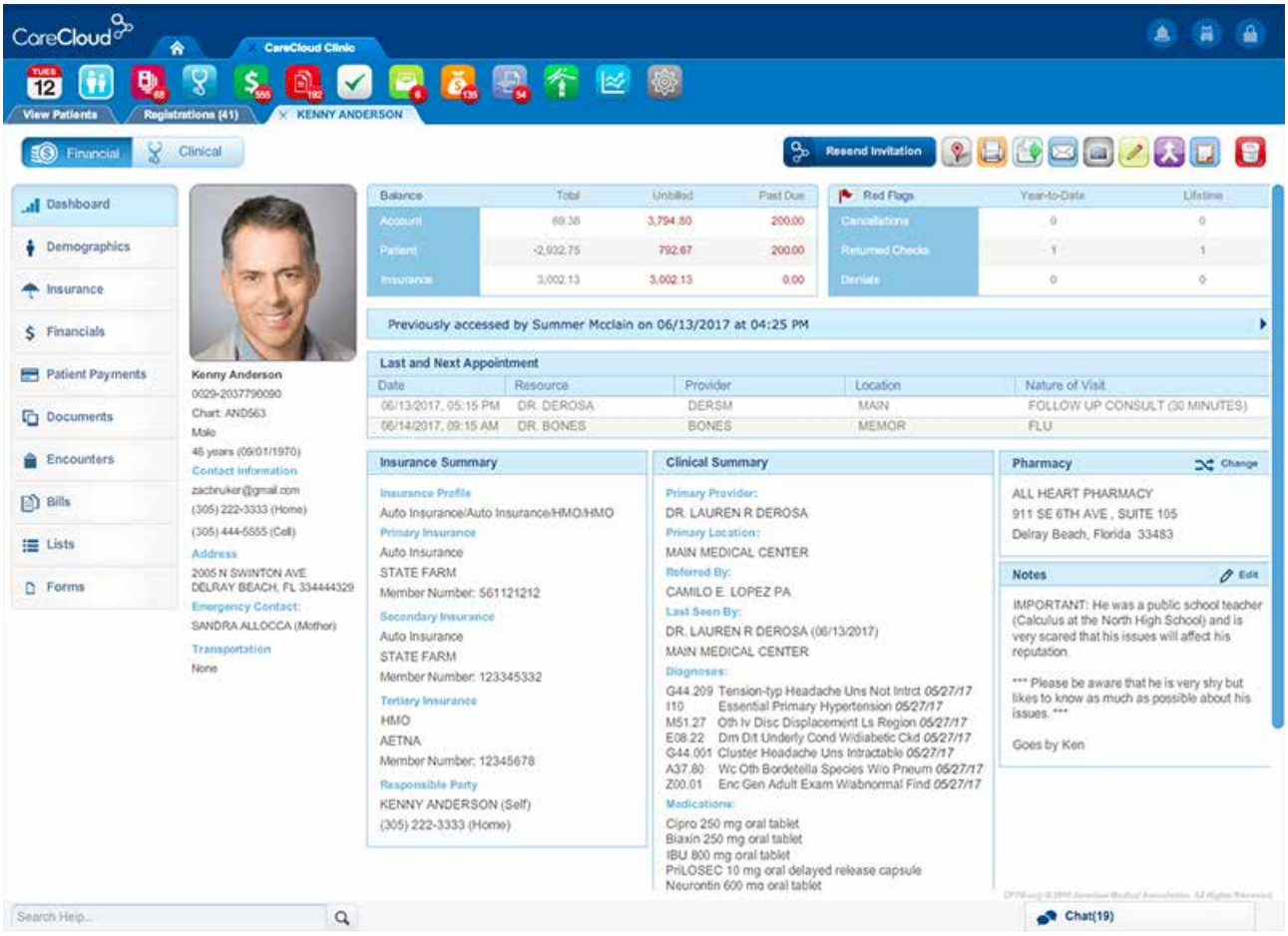

Рис. 1.4. – Інтерфейс CareCloud.

CareCloud — комерційна платформа, орієнтована на медичні центри й приватні практики, яка надає інструменти для керування записами пацієнтів, фінансами, розкладом та бізнес-аналітикою. Система пропонує розвинуті засоби моніторингу доходів, продуктивності лікарів і якості надання послуг на основі інтерактивних панелей (dashboard). Її недоліками є закритість архітектури, залежність від постачальника та висока вартість ліцензування, що обмежує можливості гнучкого наукового експериментування.

Більшість існуючих систем зупиняються на рівні статистичної звітності, не переходячи до етапу інтелектуального аналізу даних (Data Mining). Це обмежує можливості закладу у прийнятті обґрунтованих управлінських рішень.

1.3.3. Результати огляду існуючих систем

В результаті огляду, ми змогли зрозуміти що більшість сучасних медичних інформаційних систем зосереджуються переважно на реєстраційно-облікових функціях — веденні електронних медичних карток, управлінні розкладами, фінансами та звітністю. Такі системи, як Doctor Eleks, Meditech, OpenMRS та CareCloud, забезпечують автоматизацію основних бізнес-процесів поліклініки, але їхні аналітичні можливості обмежені статистичним рівнем обробки даних.

Більшість із них не реалізує механізмів інтелектуального аналізу даних , таких як класифікація, прогнозування чи пошук асоціативних закономірностей. Унаслідок цього інформаційні системи не підтримують глибокий аналіз ефективності роботи лікарів, прогнозування навантаження чи оцінку якості надання медичних послуг.

Отже, існує потреба у створенні рішення, яке поєднує класичні інструменти обліку з сучасними аналітичними підходами. Це дозволить не лише зберігати та систематизувати медичні дані, а й отримувати на їх основі нові знання для підтримки прийняття управлінських і клінічних рішень, підвищуючи загальну ефективність роботи медичного закладу.

1.4. Постановка завдання дослідження.

Проведений аналіз показав, що більшість сучасних медичних інформаційних систем орієнтовані на автоматизацію облікових процесів, але не забезпечують достатньої аналітичної підтримки для управлінських і клінічних рішень. Відсутність інтегрованих механізмів інтелектуального аналізу даних унеможливлює ефективне використання накопиченої інформації для прогнозування, оцінки якості послуг та виявлення закономірностей у роботі медичного персоналу.

Таким чином, постає завдання створення інформаційно-аналітичної системи, яка поєднує можливості бази даних, аналітичних алгоритмів і засобів візуалізації результатів для комплексної підтримки діяльності поліклініки.

Метою даного дослідження є розроблення програмного забезпечення, яке забезпечить:

- централізоване зберігання та структуроване представлення медичних даних;
- використання алгоритмів Time Series, 1-Rule, Naive Bayes, Apriori та K-Means для реалізації функцій прогнозування, класифікації, кластеризації та асоціативного аналізу;
- побудову інтерактивних графічних інтерфейсів для візуалізації результатів аналітики;
- оцінку ефективності роботи лікарів на основі KPI-показників (завантаженість, інтенсивність прийомів, повнота записів).

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Розробити концептуальну модель предметної області поліклініки, яка відображає взаємозв’язки між пацієнтами, лікарями, послугами та прийомами.
2. Побудувати архітектуру інформаційно-аналітичної системи з модульною структурою, що забезпечує розширюваність, гнучкість і можливість подальшої інтеграції з іншими медичними сервісами.
3. Реалізувати програмний модуль аналітики, що дозволяє проводити обчислення на основі алгоритмів Data Mining і Machine Learning.
4. Провести експериментальні дослідження отриманих результатів і оцінити ефективність застосованих методів.

Очікуваним результатом дослідження є створення прототипу системи, здатної не лише виконувати функції обліку, а й дозволить автоматизувати аналітичні процеси у поліклінічному середовищі, забезпечить підвищення точності

прогнозів, оптимізацію навантаження лікарів та підтримку прийняття управлінських рішень у сфері охорони здоров’я.

1.5 Підсумки системного аналізу медичних інформаційних систем

У результаті системного аналізу предметної області медичних інформаційних систем встановлено, що більшість традиційних рішень орієнтовані переважно на автоматизацію облікових процесів і не забезпечують належної аналітичної підтримки для прийняття управлінських рішень.

Проаналізовано сучасні системи, такі як Doctor Eleks, OpenMRS, CareCloud та Meditech, які демонструють різний рівень функціональності, однак не реалізують повноцінних механізмів інтелектуального аналізу даних. Це підтвердило необхідність створення нової системи, здатної об’єднати зберігання, обробку й аналітику медичної інформації в єдиному середовищі.

Результати аналізу стали основою для формування вимог до інформаційно-аналітичної системи нового покоління, спрямованої на підвищення ефективності роботи медичного персоналу, оптимізацію управлінських процесів і покращення якості медичного обслуговування.

На базі отриманих висновків було здійснено подальше моделювання предметної області та проектування архітектури системи.

РОЗДІЛ 2. МОДЕЛЮВАННЯ СИСТЕМИ.

2.1. Функціональне моделювання.

Функціональне моделювання передбачає опис бізнес-процесів, які відбуваються у медичній інформаційно-аналітичній системі, та визначення основних учасників і їх ролей. Цей підхід дозволяє зрозуміти, як користувачі взаємодіють із системою на рівні її функцій та логіки роботи.

Для функціонального моделювання системи було задіяно наступні діаграми та схеми:

- Діаграма прецеденту (Use case diagram) — описує основних користувачів системи (акторів) і функції, які вони виконують, демонструючи взаємозв’язок між користувачами та бізнес-процесами;
- Діаграма послідовності (Sequence diagram) — відображає порядок обміну повідомленнями між об’єктами системи під час виконання конкретного процесу;
- Діаграма активності (Activity diagram) — ілюструє послідовність дій, умови переходів і паралельність виконання процесів у межах робочого циклу
- Абстракції предметно області (Subject domain abstractions) — визначають основні сутності предметної області, їх властивості та обов’язки, що забезпечують логічну основу для побудови об’єктної моделі системи.

Перейдемо до їх розгляду по порядку.

2.1.1.Діаграма прецедентів

Діаграма прецедентів відображає основних користувачів інформаційно-аналітичної системи для медичного персоналу поліклініки та їх взаємодію з

ключовими функціями системи. Вона забезпечує наочне представлення взаємодій між користувачами і системою та допомагає визначити обсяг роботи.

Така діаграма зазвичай будується на основі трьох основних елементів:

1. **Перший елемент - Актори**. Це зовнішні сутності, які взаємодіють із системою. Актори можуть бути людьми, пристроями або навіть іншими системами.
2. **Другий елемент – Прецеденти**. Описують специфічні функції або дії, які система повинна виконувати у відповідь на запити акторів. Прецеденти повинні бути описані з точки зору користувача і фокусуватись на результатах, які вони отримують.
3. **Третій елемент – Зв’язки**. Вони бувають 4 типів:

- **Асоціація.** («association») Зв’язок між акторами та прецедентами, необхідний для відображення того , хто взаємодія з якими функціями системи.
- **Включення** («include»). Зв'язок який вказує на те, що один прецедент включає в себе поведінку іншого прецеденту.
- **Розширення** («extend»). Зв'язок який вказує на те, що прецедент може бути розширений іншим прецедентом при певних умовах.
- **Узагальнення** («generalization») Показує, що актор чи прецедент є спеціалізацією іншого актора або прецеденту.

На Рис. 2.1. зображено діаграму прецедентів, в ній ми можемо побачити що виділено 5 акторів – Лікар, Реєстратура, Пацієнт Аналітик та Керівник. Кожен із них має асоціації з певними прецедентами, що відображають дії в межах медичного закладу:

- Пацієнт — може записатися на прийом до лікаря, надавати власну медичну інформацію, здійснювати запити щодо прийомів.
- Реєстратор — створює медичні картки пацієнтів, здійснює запис на прийом, видає направлення або довідки, взаємодіє з модулем прийому у лікаря.
- Лікар — проводить прийом пацієнтів, ставить діагноз, призначає лікування, затверджує документи, формує медичні висновки та взаємодіє з пацієнтськими записами.
- Аналітик — відповідає за обробку накопичених даних, проводить аналітику, формує звітність, виконує моніторинг показників ефективності та оцінює результати діяльності поліклініки.
- Керівник — використовує результати аналітики для прийняття управлінських рішень, перегляду звітності, моніторингу роботи персоналу та розробки стратегічних цілей на основі отриманих даних.

Основні прецеденти системи охоплюють такі процеси:

1. Прийом у лікаря – основний сценарій взаємодії лікаря з пацієнтом. Включає постановку діагнозу, призначення лікування, видачу направлень чи довідок.

- Оптимістичний сценарій: пацієнт звертається на прийом, лікар ставить діагноз, призначає лікування.
- Альтернативний сценарій: для постановки діагнозу потрібні аналізи або консультація іншого лікаря, пацієнт отримує направлення.

2. Запис на прийом – реєстратор або сам пацієнт записує пацієнта на прийом до лікаря.

- Умови: наявність вільного часу у лікаря.

3. Створення медичної картки пацієнта – реєстратор додає нову інформацію про пацієнта в систему.

4. Проведення аналітики – аналітик аналізує дані щодо роботи лікарів, статистики захворювань, ефективності послуг.

5. Створення звітності – аналітик формує звіти для керівника.

6. Моніторинг та оцінка ефективності роботи – керівник аналізує звіти, приймає управлінські рішення.

7. Взаємодія з пацієнтами – надання інформації, прийом запитів пацієнтів, видача медичних документів.

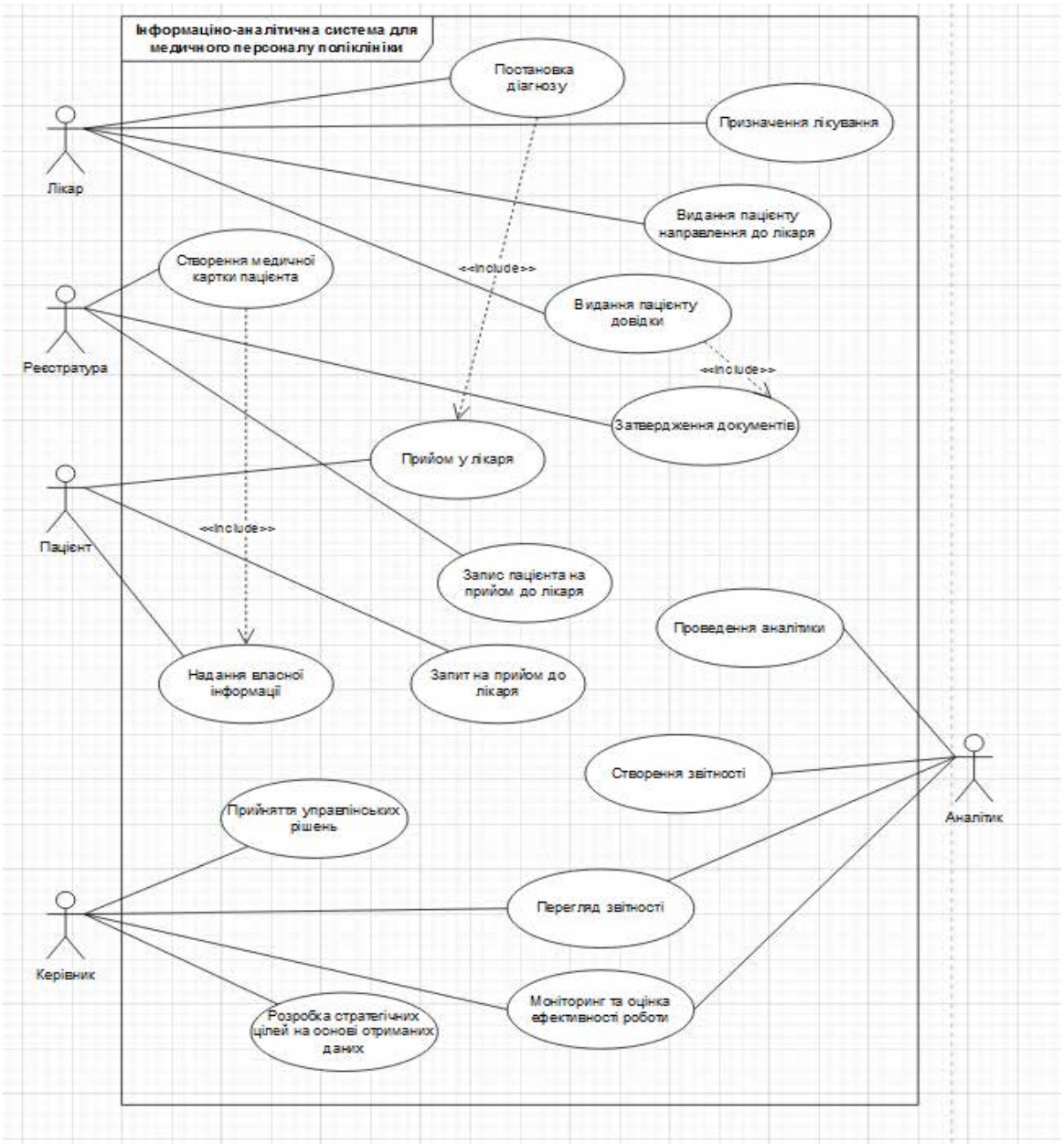

Рис. 2.1. Діаграма прецедентів медичного закладу.

2.1.2.Діаграма послідовності

Діаграма послідовності використовується для відображення взаємодії між об'єктами в системі у певній послідовності. Вона допомагає зрозуміти динамічні аспекти системи, тобто як об'єкти взаємодіють між собою для виконання певної функції або процесу. Цим самим вона сприяє ефективному плануванню, розробці та підтримці програмного забезпечення, забезпечуючи високу якість кінцевого продукту.

Діаграма послідовності будується з застосування наступних елементів:

1. **Актори.** Користувачі, які взаємодіють із системою.
2. **Лінії життя.** Вертикальні лінії, що показують існування об'єктів протягом часу.
3. **Повідомлення.** Горизонтальні стрілки, що показують обмін інформацією між об'єктами.
4. **Активні стани.** Прямокутники на лініях життя, що показують періоди, коли об'єкти виконують певні дії.
5. **Петлі, умовні блоки та інші конструкції.** Використовуються для моделювання циклів, умовних операцій та інших складних логік.

На рисунку 2.2 представлено діаграму послідовності для процесів роботи поліклінічного закладу, що охоплюють етапи запису пацієнта, прийому у лікаря та аналітично-звітної діяльності.

Основними учасниками взаємодії є:

- **Пацієнт**, який надає власні дані, записується на прийом та отримує результати обстеження або довідки;
- **Реєстратор**, що відповідає за створення медичної картки пацієнта, організацію запису та видачу документів;

- **Лікар**, який проводить прийом, діагностує, призначає лікування та формує медичні висновки;
- **Аналітик**, що здійснює збір, обробку та аналіз даних, формує аналітичні звіти;
- **Керівник**, який використовує результати аналітики для моніторингу ефективності роботи медичного персоналу;
- **База даних** — основне сховище інформації, де зберігаються відомості про пацієнтів, прийоми, призначення, медичні картки;
- **Аналітичний модуль** — програмна складова, що реалізує обчислення та візуалізацію результатів аналітичних алгоритмів (кластеризація, прогнозування, асоціативний аналіз тощо).

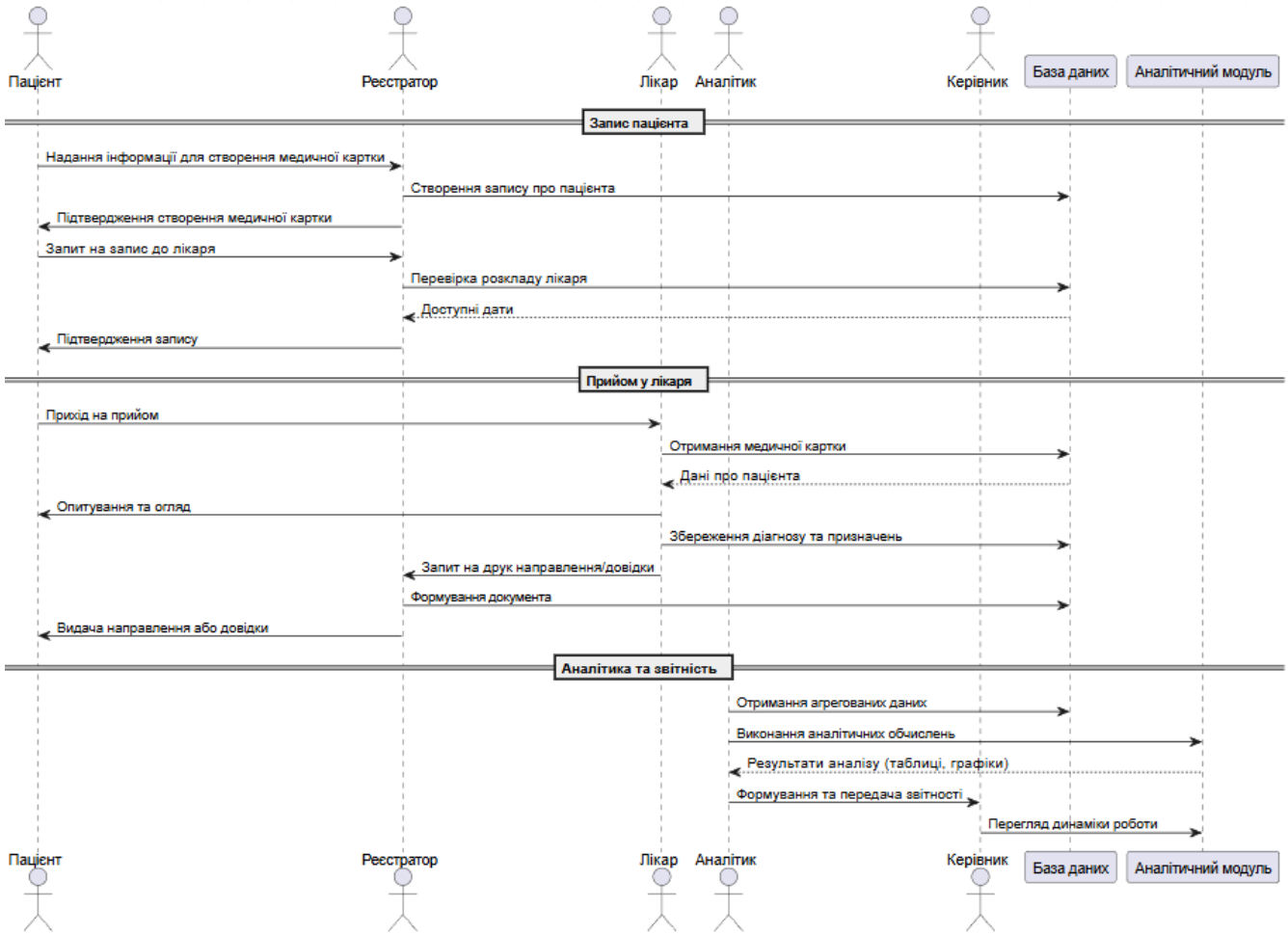

Рис. 2.2. Діаграма послідовності.

Завдяки діаграмі послідовності, було продемонстровано логіку взаємодії користувачів і підсистем у часі, а також відображено інтеграцію аналітичного модуля з основними бізнес-процесами поліклініки. Вона забезпечує повне уявлення про інформаційні потоки в системі та використовується як основа для побудови діаграм активності й компонентів.

2.1.3.Діаграма активності.

Діаграма активності використовується для відображення послідовності дій, що виконуються в межах певного бізнес-процесу системи. Вона дозволяє зрозуміти логіку виконання процесів, точки прийняття рішень, паралельність дій та умови переходу між станами. Такий тип діаграм є важливим інструментом під час моделювання поведінки системи, оскільки демонструє динаміку процесів від початку до завершення роботи користувача.

В діаграмі активності зазвичай використовуються наступні елементи:

1. **Дія (Activity).** Це вузли, що представляють виконання певних завдань або операцій.
2. **Потік управління (Control Flow).** Стрілки, що показують порядок виконання дій.
3. **Початкова точка (Initial Node).** Чорний круг, що означає початок процесу.
4. **Кінцева точка (Final Node).** Чорний круг з червоним обведенням навколо нього, що позначає кінець процесу.
5. **Розгалуження (Decision Node).** Ромб, що показує розгалуження потоку управління залежно від умов.
6. **Паралельні процеси (Fork and Join Nodes).** Товсті горизонтальні або вертикальні лінії, що показують розділення або злиття паралельних потоків.

7. **Об'єктні потоки (Object Flows).** Показують потік даних між діями.

На Рис. 2.3.1. і Рис. 2.3.2 наведено діаграму активності, що моделює процес прийому пацієнта у лікаря. Цей процес є ключовим у діяльності поліклініки, оскільки охоплює взаємодію між пацієнтом, реєстратурою та лікарем, починаючи від моменту прибуття до закладу й завершуючи формуванням результатів прийому.

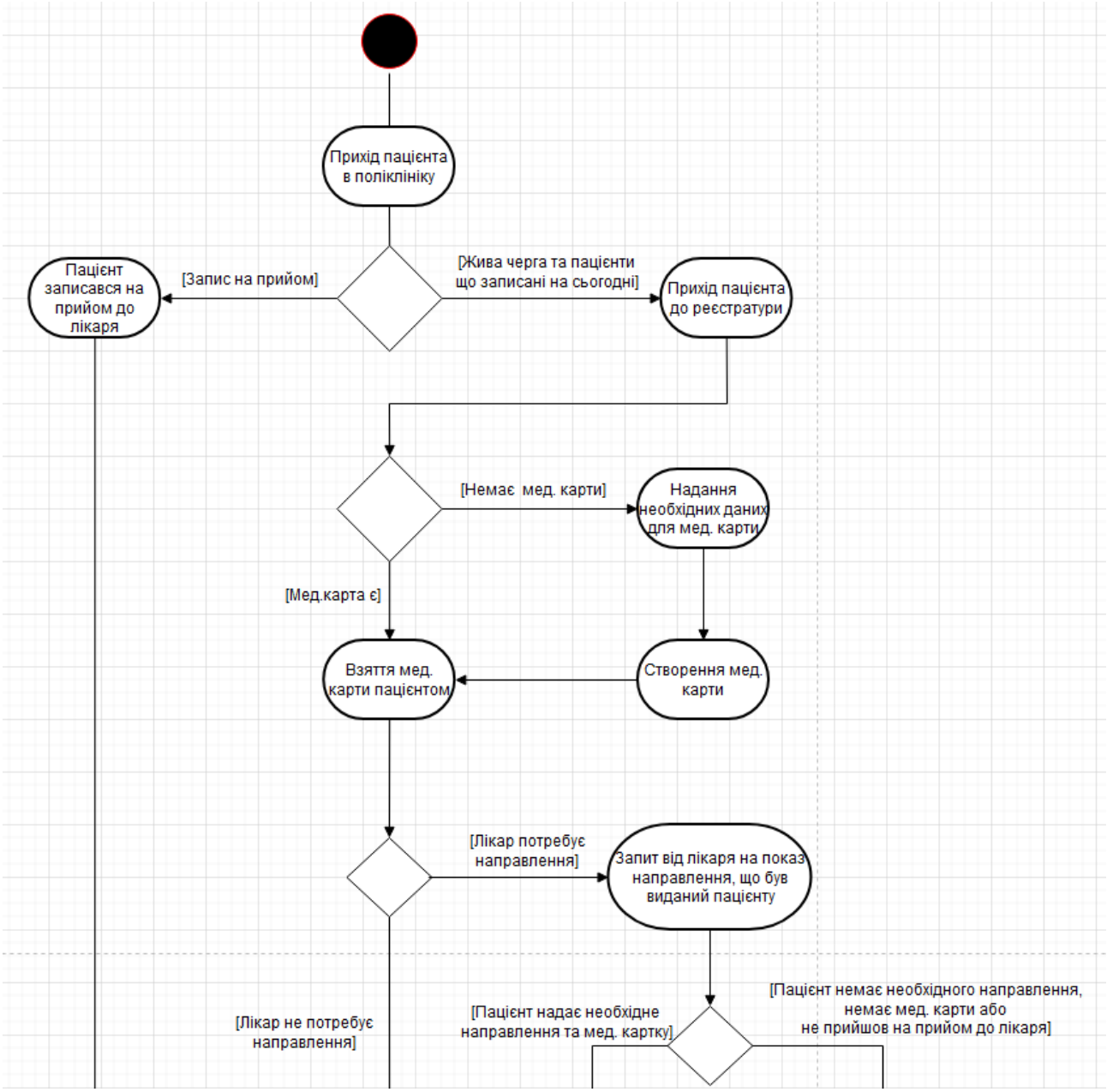

Рис. 2.3.1. Діаграма активності медичного закладу (Частина 1)

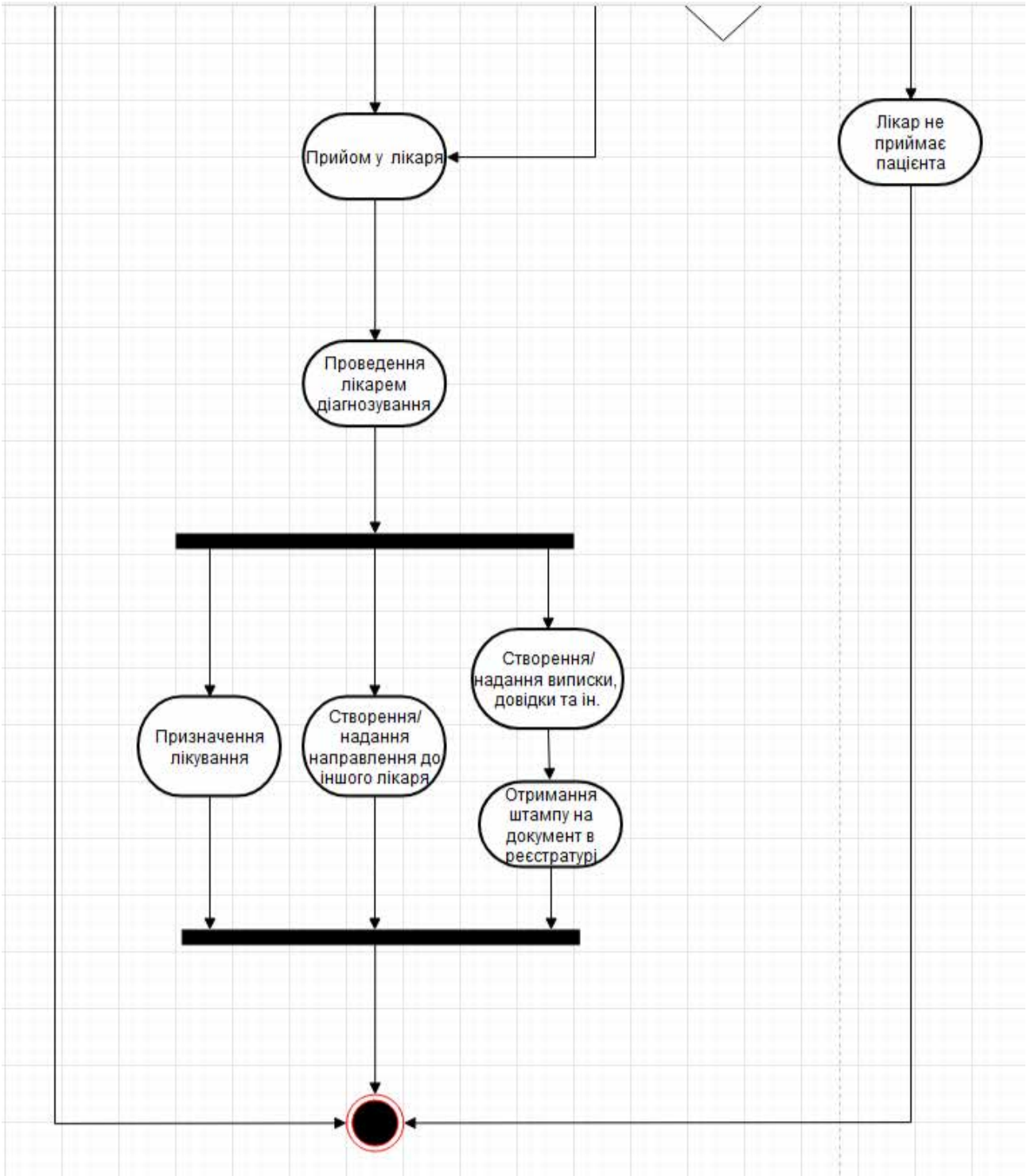

Рис. 2.3.2. Діаграма активності медичного закладу (Частина 2)

Послідовність процесу:

1. **Реєстрація пацієнта.**

   Пацієнт прибуває в поліклініку, проходить реєстрацію або підтверджує попередній запис. Якщо медична картка відсутня, реєстратор створює її на основі наданих даних.

2. **Підготовка до прийому.**

   Після отримання медичної картки пацієнт очікує прийому у лікаря. Лікар перевіряє, чи є направлення або попередній запис. У разі відсутності направлення — пацієнт не допускається до прийому.

3. **Прийом у лікаря.**

   Лікар проводить опитування, огляд і діагностування пацієнта. Залежно від результатів прийому може бути призначене лікування, видано направлення до іншого спеціаліста або сформовано медичну довідку.

4. **Завершення процесу.**

   Документи, сформовані лікарем, проходять підтвердження в реєстратурі (наприклад, отримання печатки). Після цього пацієнт завершує процес прийому.

Представлена діаграма демонструє логічну структуру процесу прийому пацієнта у медичній установі та взаємозв’язок між його етапами. Вона чітко відображає взаємодію між персоналом та пацієнтом, можливі варіанти розвитку подій залежно від умов (наявність медичної картки чи направлення) та синхронізацію дій між різними ролями.

Таким чином, діаграма активності дає змогу візуалізувати алгоритм роботи поліклініки, допомагає оптимізувати процес прийому та визначити точки, де може бути застосована автоматизація.

2.1.4.Абстракції предметної області.

Абстракції предметної області дозволяють виділити основні сутності системи, їх властивості та поведінку, що лежать в основі функціонування інформаційно-аналітичної системи поліклініки. Вони забезпечують зв’язок між реальними об’єктами предметної області (пацієнтами, лікарями, діагнозами, послугами тощо) та програмними компонентами системи.

На Рис. 2.4. зображено одні з основних абстракцій медичного закладу. На них зображено які властивості приймає і містить в собі кожна з абстракцій та які обов’язки вони можуть виконувати.

Підводячи загальні підсумки абстракції предметної області необхідні для того щоб показати що може виконувати той чи інший елемент в предметній області та які властивості він буде мати.

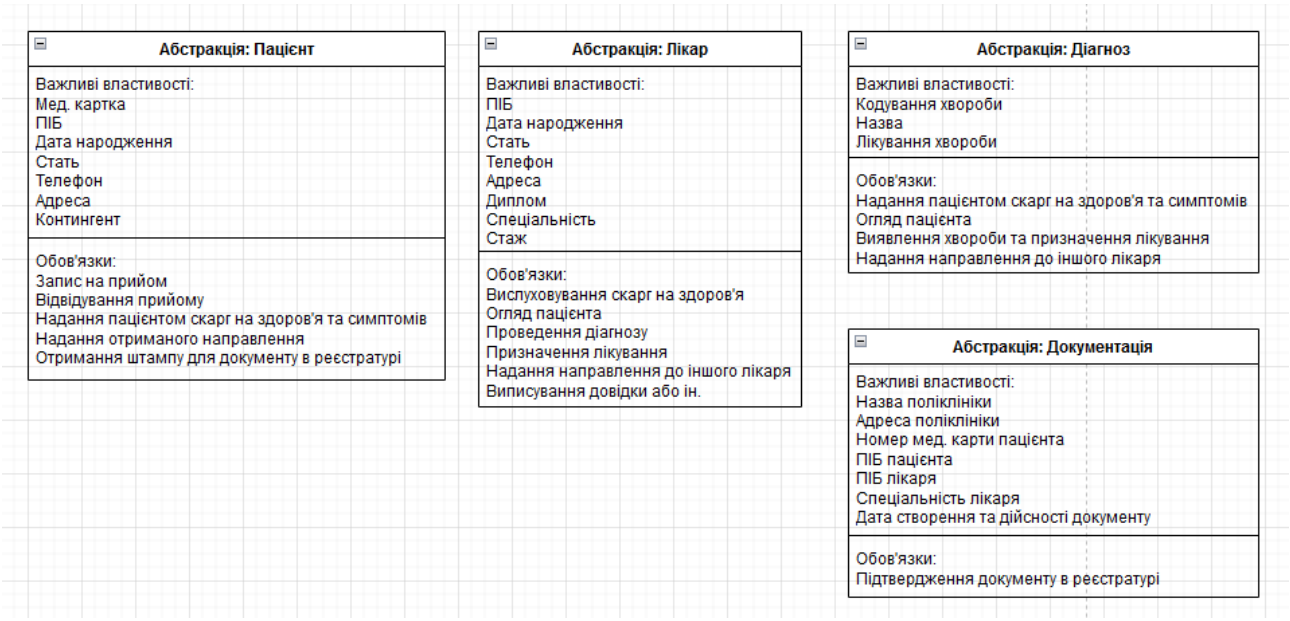

Рис. 2.5. Абстракції медичного закладу

2.2. Об’єктно-орієнтоване моделювання

Об’єктно-орієнтоване моделювання (ООМ) є одним із ключових етапів процесу проєктування програмних систем, оскільки воно дозволяє представити структуру системи у вигляді взаємопов’язаних об’єктів, що мають власні властивості, поведінку та зв’язки. Такий підхід є основою сучасних методологій розроблення програмного забезпечення, адже забезпечує логічну узгодженість між етапами аналізу, проєктування та реалізації.

Мета об’єктно-орієнтованого моделювання полягає у визначенні структурних компонентів системи, їх взаємодії та залежностей між даними й функціональністю. На відміну від функціонального моделювання, яке описує що робить система, об’єктно-орієнтоване моделювання зосереджується на тому, як саме система організована всередині.

У межах розроблення інформаційно-аналітичної системи для медичного персоналу поліклініки об’єктно-орієнтоване моделювання охоплює такі елементи:

- Діаграма класів (Class Diagram) — визначає основні класи системи, їх атрибути, методи та типи зв’язків (асоціації, агрегації, композиції, наслідування). Вона відображає логічну структуру системи та є основою для її програмної реалізації.
- Прості кооперації (Simple Cooperation Diagram) — демонструють взаємодію між об’єктами у межах конкретних сценаріїв, відображаючи обмін повідомленнями та залежності між класами.
- Діаграма компонентів (Component Diagram) — описує модульну архітектуру системи, показуючи, як різні програмні частини (модулі, бібліотеки, бази даних) взаємодіють між собою.
- Діаграма розгортання (Deployment Diagram) — ілюструє фізичну структуру системи, зокрема розміщення програмних компонентів на апаратних вузлах

(сервер, клієнтський ПК, база даних), що дозволяє оцінити архітектуру розгортання та взаємодію середовищ виконання.

2.2.1. Діаграма класів

Діаграма класів є одним з основних інструментів моделювання у UML (Unified Modeling Language). Вона використовується для відображення структури системи, показуючи класи, їх атрибути, методи та взаємозв'язки між ними. Діаграма класів надає статичне уявлення про систему, зосереджуючись на її ключових компонентах.

Основна мета діаграми класів полягає у тому, щоб надати структурну модель інформаційно-аналітичної системи, яка згодом може бути безпосередньо реалізована у вигляді програмного коду. Така діаграма дозволяє ефективно спроектувати архітектуру системи, забезпечити узгодженість між логічною моделлю бази даних та програмними модулями, а також полегшити подальшу підтримку і розвиток програмного забезпечення.

Для того щоб спроектувати діаграму класів, необхідно використовувати наступні елементи:

Класи. Прямокутники, що представляють класи в системі. Вони містять три частини: назву класу, атрибути і методи.

- **Назва класу**. Ім'я класу, яке зазвичай розміщується у верхній частині прямокутника.
- **Атрибути**. Властивості або характеристики класу, що розміщуються в середній частині.
- **Методи**. Операції або функції, які може виконувати клас, розташовуються в нижній частині.

Асоціації. Лінії, що з'єднують класи і показують їхні взаємозв'язки.

Агрегація. Спеціальний тип асоціації, що показує, що один клас є частиною іншого. Позначається порожнім ромбом на стороні цілого.

Композиція. Сильніший тип агрегації, що показує, що клас не може існувати окремо від іншого. Позначається заповненим ромбом на стороні цілого.

Успадкування (Генералізація). Лінія з порожнім трикутником на стороні базового класу, що показує відношення між суперкласом і підкласами.

Залежність. Пунктирна стрілка, що вказує на те, що один клас залежить від іншого.

На Рис. 2.6. зображено діаграму класів з використанням лише асоціацій між ними для предметної області медичний заклад:

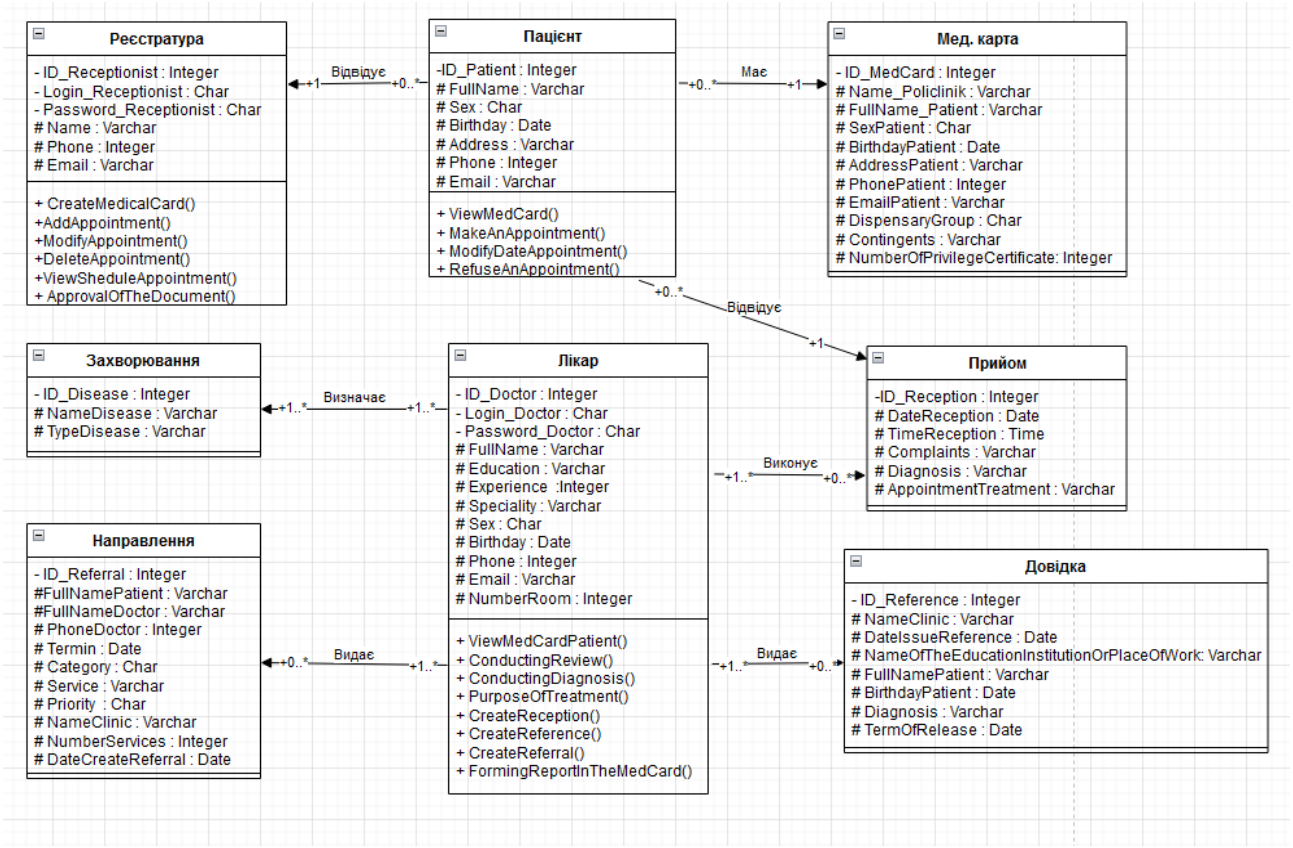

Рис. 2.6. Діаграма класів медичного закладу.

2.2.2.Прості кооперації

Діаграми простих кооперацій (collaboration diagrams) відображають взаємодію між об’єктами системи у межах конкретних сценаріїв її роботи. На відміну від діаграм класів, які демонструють статичну структуру системи, кооперації акцентують увагу на динамічних аспектах — хто, коли і яким чином взаємодіє для досягнення певного результату.

Прості кооперації дозволяють наочно представити, як об'єкти взаємодіють для досягнення певної мети. Немало важливим є те що кооперації допомагають розробникам краще зрозуміти структуру і функціональність системи, що полегшує процес проектування. А також вони дають виявити залежності між компонентами системи, що важливо для подальшого розвитку та тестування.

Такі діаграми дозволяють деталізувати логіку виконання ключових бізнес-процесів, описаних у діаграмі прецедентів, і виступають проміжним етапом між функціональним і об’єктно-орієнтованим моделюванням.

Для предметної області «медичний заклад» було спроектовано чотири основні кооперації, що відображають ключові сценарії взаємодії між об’єктами системи: «Запис на прийом», «Постановка діагнозу», «Створення медичної картки» та «Надання направлення і довідки».

На рисунку 2.7.1 представлено просту кооперацію «Запис на прийом». Дана діаграма демонструє процес запису пацієнта на прийом до лікаря.

Основними об’єктами є:

- **Реєстратор**, який створює або змінює запис у системі;
- **Пацієнт**, що подає запит на запис;
- **Лікар**, до якого здійснюється запис;
- **Прийом**, як окрема сутність, що містить інформацію про дату, час і мету звернення.

Пацієнт звертається до реєстратури з метою запису, після чого реєстратор перевіряє наявність вільного часу у лікаря та, у разі можливості, створює відповідний запис у системі. У результаті формується сутність «Прийом», яка містить інформацію про дату, час і лікаря, до якого заплановано відвідування.

У визначений час пацієнт прибуває на прийом, де лікар проводить огляд і приймає скарги пацієнта. Важливо зазначити, що клас «Прийом» має композиційні зв’язки з класами «Пацієнт» та «Лікар», оскільки його існування безпосередньо залежить від наявності обох цих об’єктів. У разі відсутності лікаря або пацієнта прийом не може бути реалізований, що відображає реальну логіку взаємозалежності компонентів системи.

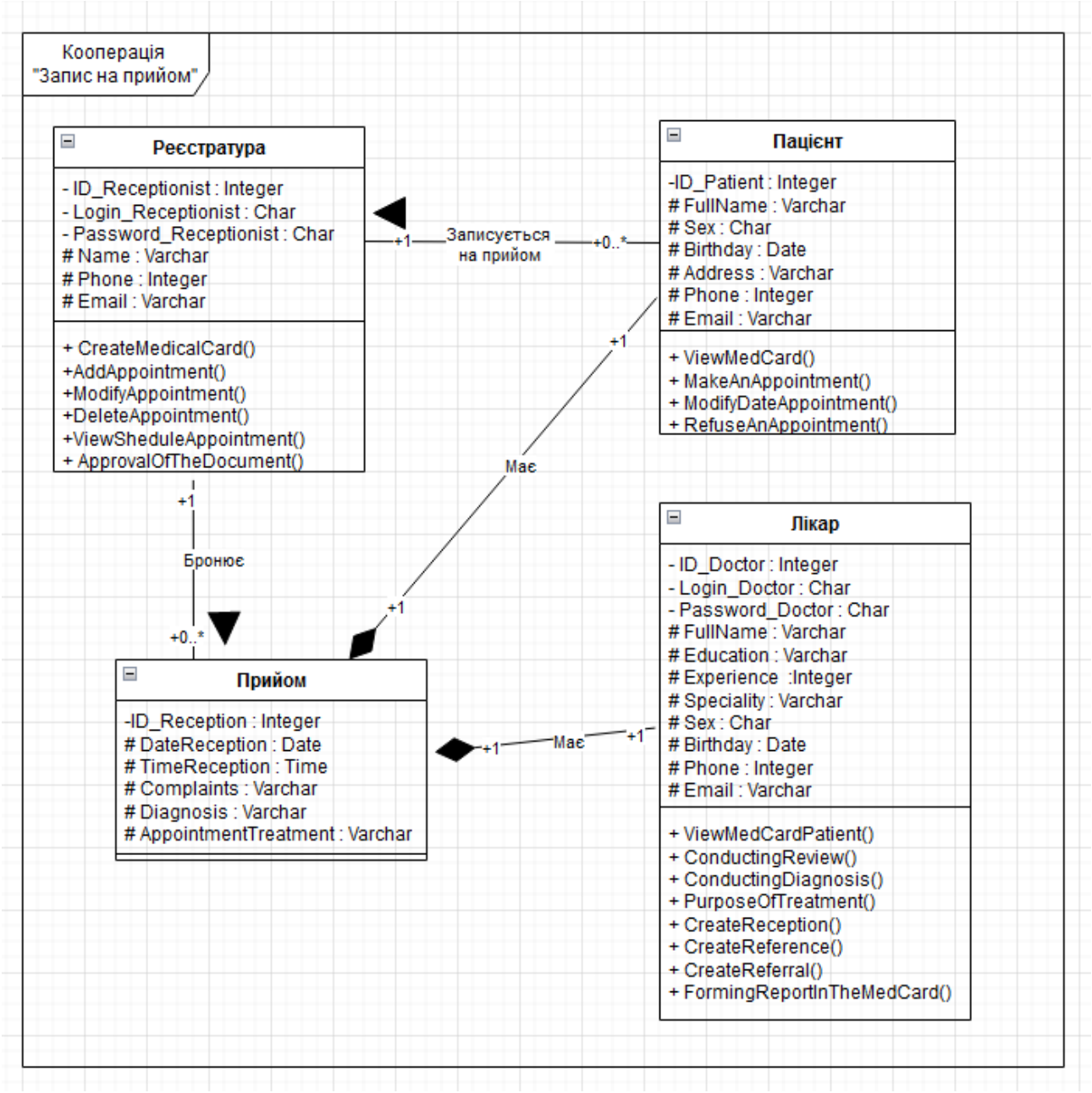

Рис. 2.7.1. Кооперація «Запис на прийом»

Наступне зображення (Рис. 2.7.2.) простої кооперації «Постановка діагнозу». Вона описує процес встановлення лікарем діагнозу для пацієнта під час прийому.

Головними об’єктами є:

- **Пацієнт**, який надає скарги;
- **Лікар**, що проводить огляд і визначає захворювання;

- **Захворювання**, яке зберігає інформацію про тип і назву встановленої хвороби.

У межах цієї кооперації лікар визначає діагноз, пов’язуючи його з конкретним пацієнтом, що забезпечує логічну цілісність даних і дає змогу формувати статистику захворювань.

В цій кооперації Пацієнт має агрегований зв'язок з Захворюванням, адже для того щоб пацієнт прийшов на прийом, він повинен себе погано відчувати.

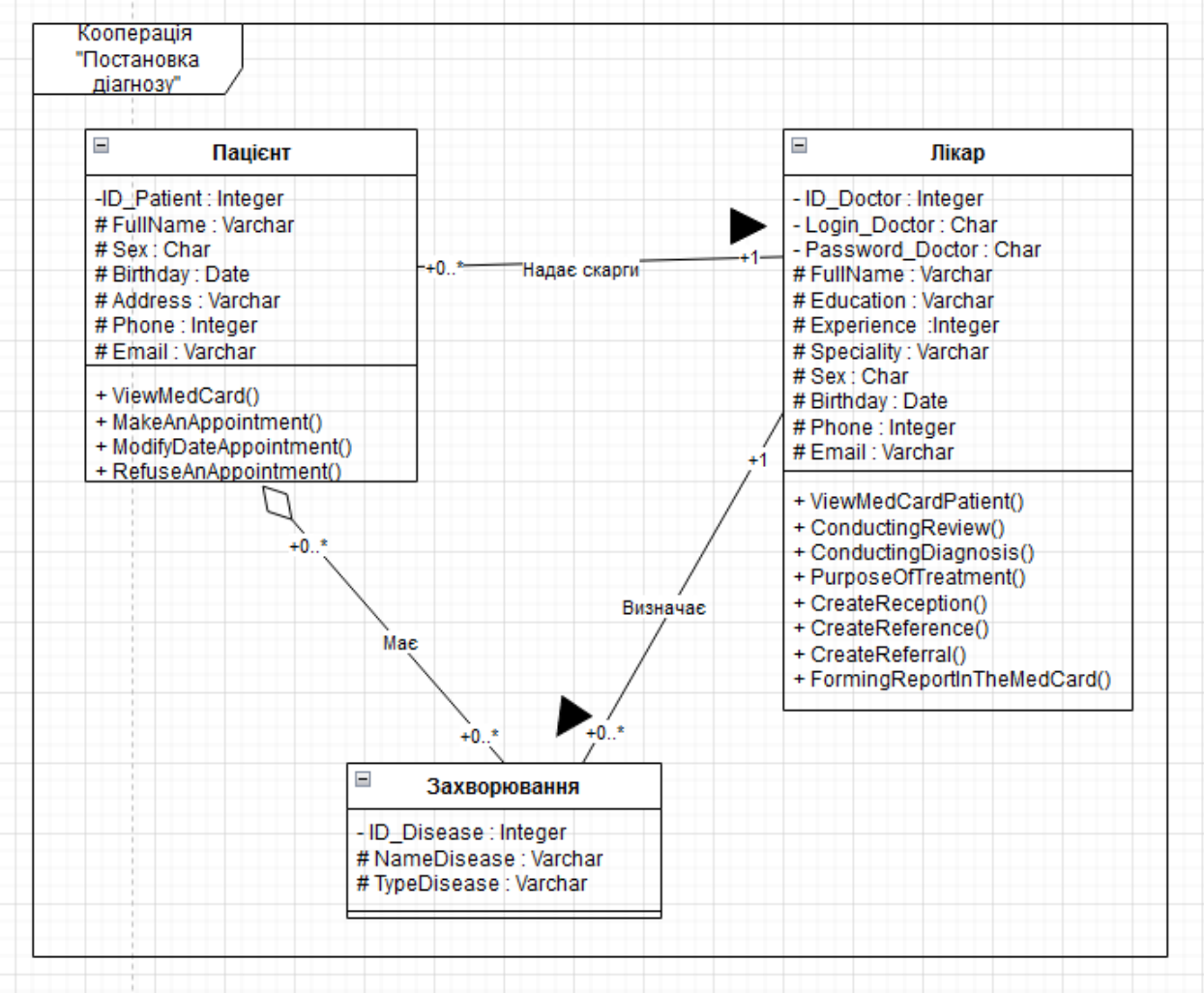

Рис. 2.7.2. Кооперація «Постановка діагнозу»

Третя кооперації «Створення мед. карти» що зображено на Рис. 2.7.3. демонструє процес створення медичної картки для пацієнта.

До взаємодії залучено три об’єкти:

- **Пацієнт**, який надає необхідні персональні дані;
- **Реєстратор**, що ініціює створення картки або її пошук у базі;
- **Медична картка**, у якій фіксується вся інформація про особу, звернення та лікування.

Кооперація відображає логіку створення запису про пацієнта, перевірку наявності існуючої картки, а також її зв’язок із реєстратурою.

Пацієнт приходить до медичного закладу та робить запит на створення Мед. карти в Реєстратурі. Реєстратура отримавши необхідні дані від Пацієнта створює йому Мед. картку та надає її йому.

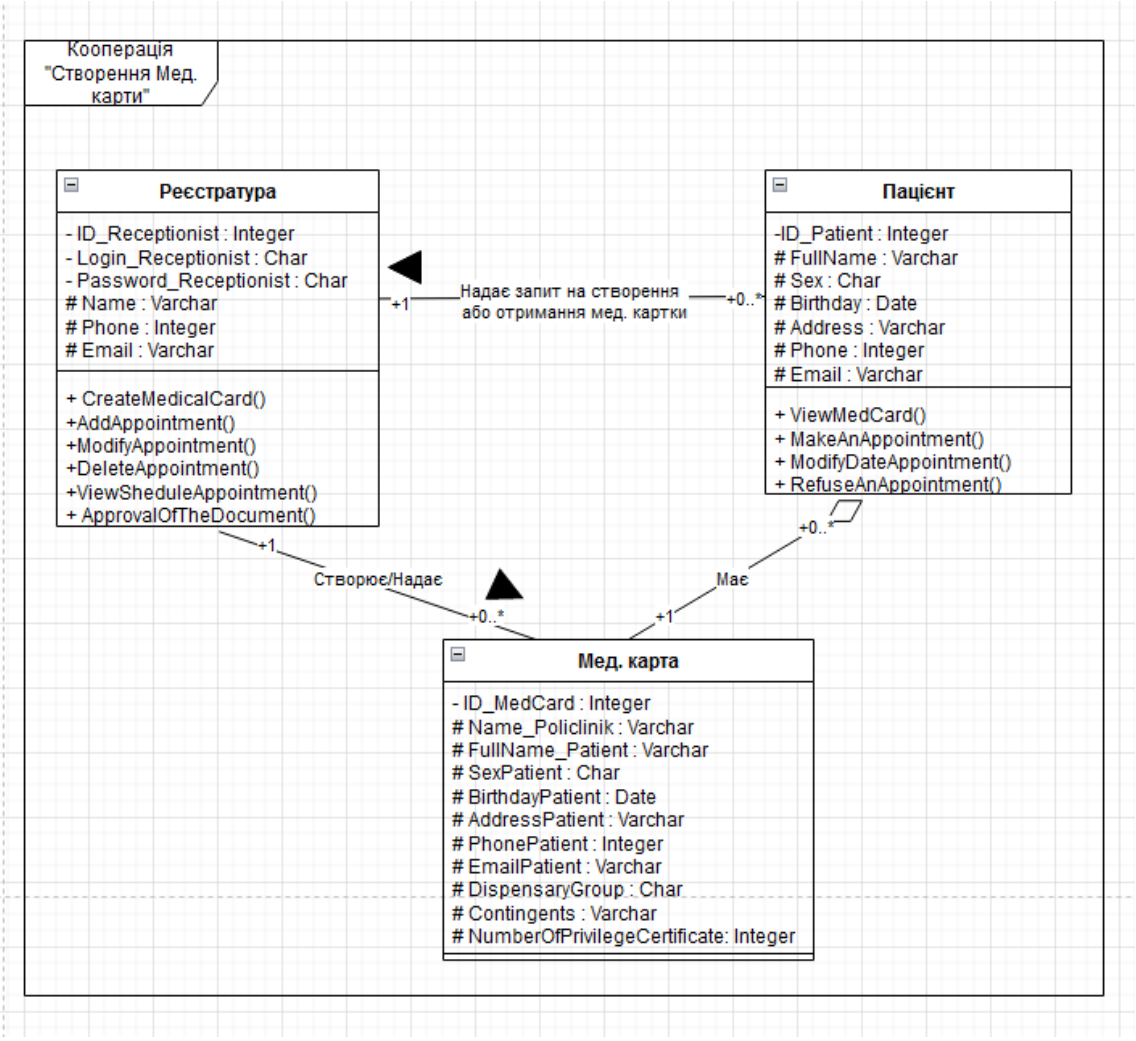

Рис. 2.7.3. Кооперація «Створення Мед. карти»

Останньою кооперацією є «Надання направлення і довідки», яка зображена на Рис. 2.7.4. Вона відображає процес оформлення медичних документів — направлень і довідок.

Основними учасниками є:

- **Лікар**, який визначає необхідність видачі документа;
- **Захворювання**, що є підставою для направлення або довідки;
- **Документ**, який є узагальненим класом для двох підтипів — Довідка і Направлення.

Кооперація демонструє, як під час прийому лікар створює документ, заповнює його даними про пацієнта, діагноз і тип обстеження, після чого система формує готову довідку або направлення для друку.

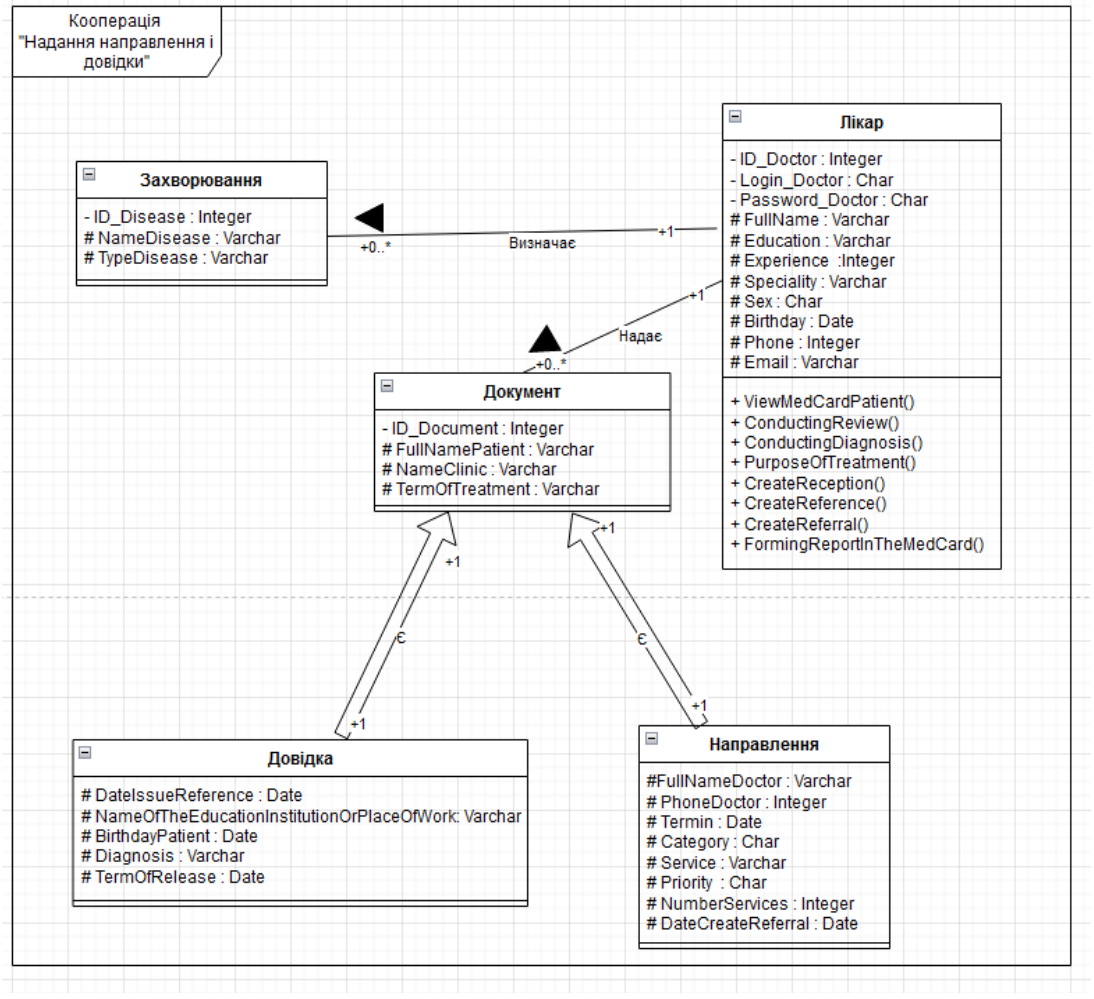

Рис. 2.7.4. Кооперація «Надання направлення і довідки»

2.2.3.Діаграма пакетів

Діаграма пакетів використовується для відображення логічної структури програмного забезпечення шляхом групування класів і компонентів у функціональні блоки. Така діаграма дозволяє описати взаємозв'язки між частинами системи на рівні архітектури та показати, як окремі модулі взаємодіють між собою. Вона є важливим інструментом для узагальненого представлення системи, полегшує її розуміння, супровід і подальший розвиток.

На рисунку 2.8 представлено діаграму пакетів, розроблену для інформаційно-аналітичної системи медичного персоналу поліклініки. Система реалізована у вигляді трирівневої архітектури, що складається з таких основних шарів:

- **GUI Layer (Рівень користувацького інтерфейсу)** – забезпечує взаємодію користувача із системою. Містить підпакети UserGUI і ClientNetwork, які відповідають відповідно за графічний інтерфейс користувача та мережеву взаємодію клієнта з сервером.
- **Business Layer (Рівень бізнес-логіки)** – реалізує основну функціональність системи. Містить пакети Application Facade, Business Workflow, Business Components та Business Entities. Цей рівень відповідає за обробку даних, реалізацію бізнес-процесів поліклініки, формування аналітичних результатів і звітів.
- **Data Layer (Рівень даних)** – забезпечує роботу з базою даних. Складається з підпакетів ServerNetwork, Access Data та Sources Data, які реалізують механізми підключення, доступу до сховища даних та взаємодії з ним.
- **Cross-Cutting Themes (Поперечні аспекти системи)** – окремий набір пакетів, які впливають на роботу всіх шарів. До них належать Security (модуль безпеки, що відповідає за автентифікацію, авторизацію й захист

даних) та Communication (модуль комунікацій, який забезпечує обмін інформацією між модулями системи).

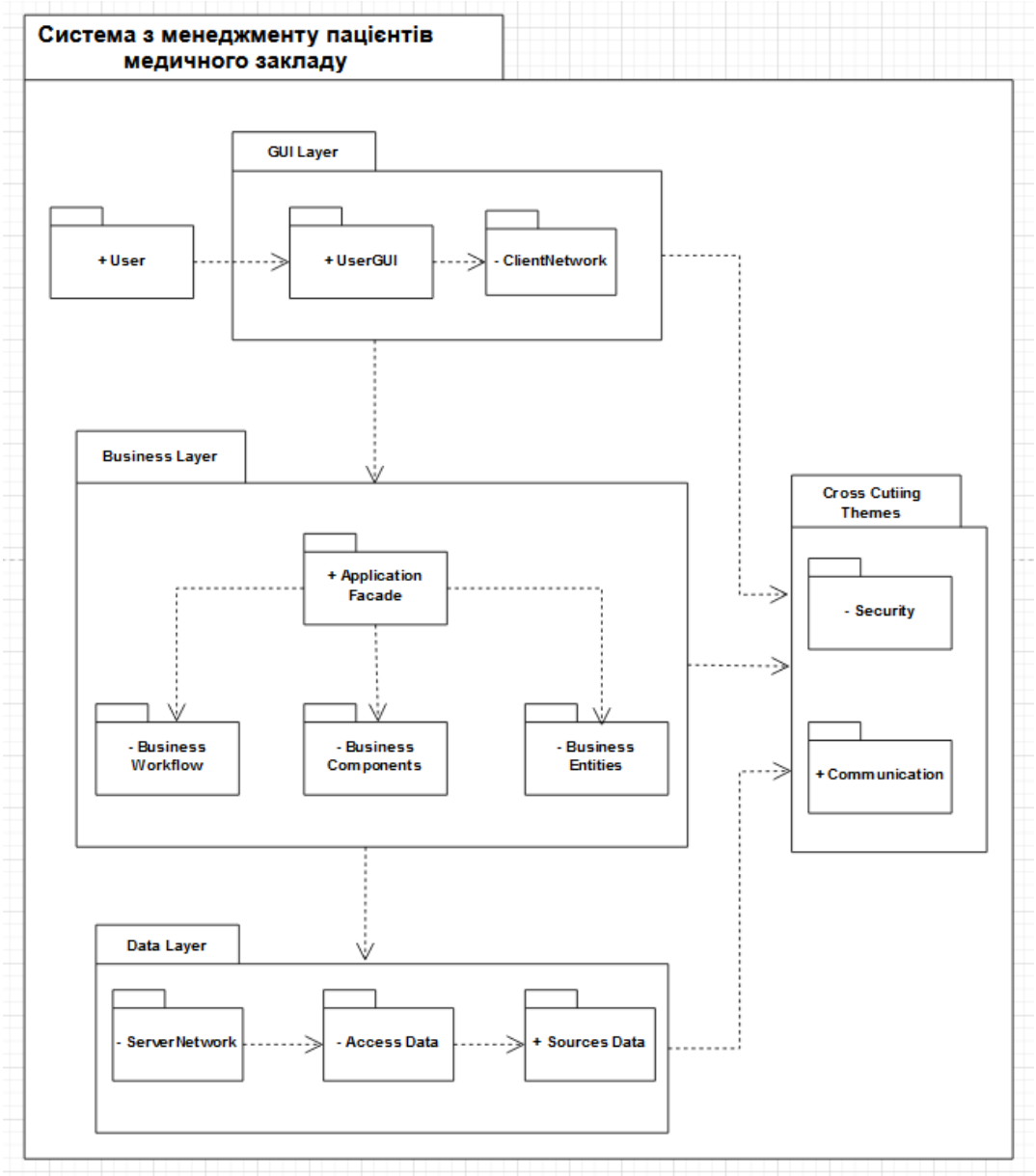

Рис. 2.8. Діаграма пакетів

Дана діаграма дозволяє чітко прослідкувати залежності між пакетами та відобразити взаємозв’язок між рівнями системи. Такий підхід сприяє

структурованому проектуванню, розширюваності та зручності супроводу програмного продукту.

2.2.4.Діаграма компонентів

Діаграма компонентів використовується для представлення фізичної структури програмного забезпечення, демонструючи основні модулі системи, їх функціональні призначення та взаємозв'язки. Така діаграма показує, з яких компонентів складається система, як вони організовані, взаємодіють між собою та яким чином забезпечується реалізація бізнес-логіки.

На рисунку 2.9 зображено діаграму компонентів, розроблену для інформаційно-аналітичної системи медичного персоналу поліклініки. Вона відображає архітектурну модель системи, що побудована за принципом багаторівневої (трирівневої) архітектури, яка включає такі основні компоненти:

- **GUI Interface (Інтерфейс користувача)** – компонент, який забезпечує взаємодію користувача із системою. Через нього лікарі, реєстратори, аналітики та керівники отримують доступ до необхідних функцій, здійснюють перегляд, введення та обробку інформації.
- **Application (Прикладний рівень)** – основний компонент, який містить дві логічні підсистеми:
  - Authentication – відповідає за процеси автентифікації користувачів, перевірку облікових даних, надання прав доступу до окремих модулів;
  - Business Logic and Data Processing – реалізує основні функції системи: обробку медичних даних, створення записів про прийоми, формування медичних карт, виконання аналітичних розрахунків і генерацію звітів.
- **DB Storage (Сховище даних)** – компонент, який забезпечує зберігання структурованих даних про пацієнтів, лікарів, послуги, результати

обстежень і статистичну інформацію. Зв’язок між цим компонентом та прикладним рівнем реалізується через канал Data Interaction, що гарантує обмін даними в реальному часі.

Компоненти системи взаємодіють між собою через стандартизовані інтерфейси:

- Application Interaction – забезпечує комунікацію між користувацьким інтерфейсом і прикладним рівнем;
- Data Interaction – визначає обмін інформацією між прикладним рівнем і базою даних.

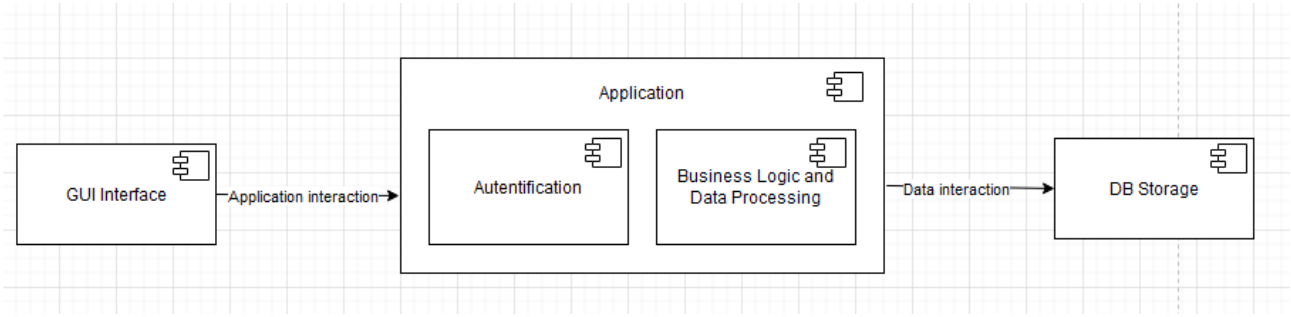

Рис. 2.9. Діаграма компонентів.

2.2.5.Діаграма розгортання

Діаграма розгортання (Deployment diagram) відображає фізичну структуру інформаційно-аналітичної системи та взаємодію її апаратних і програмних компонентів у середовищі виконання. Вона дозволяє наочно показати, на яких пристроях розгортаються окремі модулі системи, які з’єднання між ними встановлюються та як здійснюється обмін даними.

На рисунку 2.10 подано діаграму розгортання інформаційно-аналітичної системи для медичного персоналу поліклініки. Архітектура реалізована за клієнт-серверною моделлю, у якій користувачі взаємодіють із центральним застосунком і базою даних через мережеві з’єднання TCP/IP.

Основні вузли системи:

- **Personal Device (Персональний пристрій)** – робоча станція користувача (лікаря, реєстратора, аналітика або керівника), на якій розгортається клієнтська частина застосунку.
    - Користувачі здійснюють вхід у систему, переглядають інформацію, створюють записи та формують запити до бази даних.
    - Пристрій може бути під’єднаний до принтера, що забезпечує друк медичних документів — довідок, направлень та звітів.
- **Application (Сервер застосунку)** – основний обчислювальний вузол, який виконує логіку роботи системи. До складу цього вузла входять такі компоненти:
    - Authentication and Authorization Module — забезпечує перевірку облікових даних користувачів, автентифікацію та керування правами доступу;
    - Patient Management — модуль, який відповідає за облік пацієнтів, створення медичних карт і записів на прийом;
    - User Interface — серверна частина користувацького інтерфейсу, що обробляє запити клієнтів і повертає результати;
    - Medical Data Processing — компонент, який реалізує основні функції з обробки медичних даних, постановки діагнозів і створення документів.

- **Database (База даних/ Clinic Database)** – сервер бази даних, у якому зберігається основна інформація про пацієнтів, лікарів, прийоми, діагнози, направлення, а також службові дані для роботи застосунку. Передача даних між базою та сервером застосунку здійснюється через протокол TCP/IP із використанням механізмів шифрування для забезпечення безпеки.
- **DataStorage (Сховище даних)** – окремий вузол, який містить агреговані, статистичні та аналітичні дані, сформовані на основі записів із основної бази

даних. На відміну від Clinic Database, це сховище призначене для обробки великих обсягів інформації під час виконання аналітичних запитів. Використання окремого DataStorage зменшує навантаження на основну базу даних, прискорює побудову звітів і забезпечує стабільну роботу аналітичних модулів.

- **Analyst Workstation (Робоча станція аналітика)** – окремий вузол, що містить підсистеми:
    - Analytical Data Module — модуль аналітичної обробки даних, який виконує статистичні та прогнозні обчислення;
    - Reporting and Statistics Module — компонент, який формує звіти та візуалізує результати аналізу для керівництва поліклініки. Робоча станція аналітика взаємодіє з окремим сховищем аналітичних даних (DataStorage), де зберігаються агреговані дані для побудови звітів.

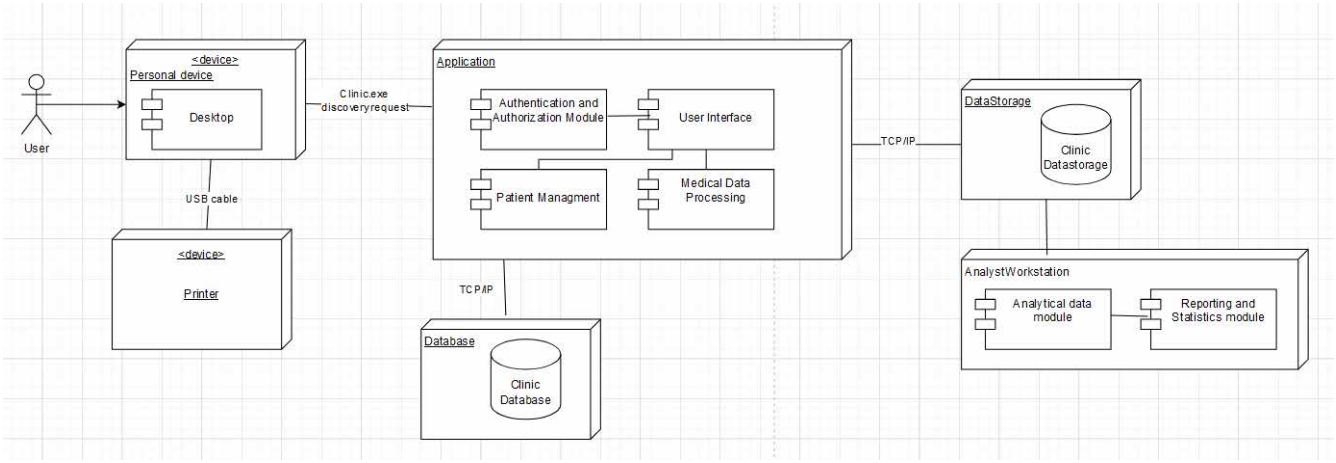

Рис. 2.10. Діаграма розгортання

Уся система функціонує у єдиному інформаційному просторі, де забезпечено безпечний обмін даними через стандартизовані протоколи, розмежування доступу користувачів та можливість масштабування.

2.3. Підсумки моделювання системи

У цьому розділі було проведено повне функціональне та об’єктно-орієнтоване моделювання інформаційно-аналітичної системи для медичного персоналу поліклініки. Розроблені моделі дозволили комплексно описати як бізнес-процеси предметної області, так і структуру майбутнього програмного забезпечення.

На етапі функціонального моделювання було визначено основні сценарії взаємодії користувачів із системою та побудовано діаграми прецедентів, послідовності й активності, які відображають динаміку роботи медичного закладу. Ці діаграми дозволили виявити ключові функціональні ролі (лікар, реєстратор, пацієнт, аналітик, керівник) і визначити основні процеси — створення медичних карт, запис на прийом, постановку діагнозу, формування звітності та виконання аналітики.

У рамках об’єктно-орієнтованого моделювання було побудовано діаграми класів і простих кооперацій, які формалізують структуру системи у вигляді об’єктів, атрибутів і зв’язків між ними. Кооперації деталізують взаємодію між сутностями у процесах «Запис на прийом», «Постановка діагнозу», «Створення медичної карти» та «Надання направлення і довідки», що дозволяє чітко простежити логіку інформаційних потоків. Діаграма пакетів забезпечила структуризацію програмного комплексу на рівні логічних шарів — інтерфейсного, бізнесового та рівня доступу до даних. Діаграма компонентів відобразила модульну побудову системи, яка передбачає чітке розмежування відповідальності між модулями аутентифікації, обробки медичних даних, бізнес-логіки та збереження інформації. Завершальним етапом моделювання стала діаграма розгортання, що демонструє фізичну архітектуру системи у середовищі виконання, включаючи взаємодію між персональними пристроями користувачів, сервером застосунку,

базою даних і аналітичним сховищем. Такий підхід забезпечує масштабованість, надійність та інтегрованість системи з іншими інформаційними ресурсами.

Отримані результати моделювання створюють міцну основу для подальшого етапу — проектування архітектури програмного забезпечення та реалізації системи у вигляді функціонального програмного продукту, який забезпечує ефективне управління медичними даними та підтримку прийняття рішень у поліклініці.

РОЗДІЛ 3. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1. Архітектура програмного забезпечення

Розробка архітектури програмного забезпечення є ключовим етапом створення інформаційно-аналітичної системи, оскільки саме на цьому рівні визначаються структурні компоненти, логіка їхньої взаємодії та принципи організації даних. Архітектура системи повинна забезпечувати високу надійність, масштабованість, зручність супроводу та можливість подальшого розширення функціональності.

Архітектура розробленого програмного комплексу базується на багаторівневій (трирівневій) структурі, що складається з:

- **рівня представлення (Presentation Layer)** — забезпечує взаємодію користувача із системою через графічний інтерфейс, реалізований засобами Windows Forms;
- **рівня бізнес-логіки (Business Logic Layer)** — відповідає за обробку даних, перевірку коректності введеної інформації, виконання розрахунків KPI, аналітичних алгоритмів та обробку запитів до бази даних;
- **рівня доступу до даних (Data Access Layer)** — здійснює з’єднання з базою даних Microsoft SQL Server, забезпечує виконання SQL-запитів, збереження, оновлення та вибірку даних.

Модульна структура системи дозволяє ізолювати функціональність кожного блоку — від авторизації користувачів до виконання аналітичних обчислень. Такий підхід спрощує тестування, підвищує стабільність роботи системи та забезпечує можливість інтеграції нових модулів без суттєвого втручання в існуючий код.

Ключовою особливістю архітектури є інтеграція аналітичного модуля, який реалізує алгоритми прогнозування, класифікації, кластеризації та асоціативного аналізу. Цей модуль взаємодіє з основною базою даних через запити до аналітичного сховища, використовуючи агреговані показники діяльності поліклініки. Отримані результати візуалізуються у вигляді графіків, таблиць та показників ефективності, що дозволяє медичному персоналу та адміністрації оперативно оцінювати стан медичних процесів.

Загальна структура архітектури відображена на діаграмі розгортання (Рис. 2.10), де показано взаємодію між клієнтською частиною (інтерфейсом користувача), сервером застосунку, базою даних та аналітичним модулем. Такий підхід дозволяє розподілити обчислювальні навантаження, забезпечити стабільність роботи та гнучкість у розширенні системи.

3.2. Модулі системи та їх функціональне призначення

Програмний комплекс інформаційно-аналітичної системи для медичного персоналу поліклініки має модульну структуру, що забезпечує розподіл функцій між окремими частинами системи. Такий підхід підвищує зручність супроводу, масштабованість і можливість подальшого розвитку програмного забезпечення. Кожен модуль виконує чітко визначене завдання та взаємодіє з іншими через базу даних і бізнес-логіку.

1. Модуль авторизації користувачів

Цей модуль відповідає за ідентифікацію та автентифікацію користувачів системи. Він забезпечує розмежування доступу до функціоналу відповідно до ролі користувача — реєстратор, лікар, аналітик або керівник. Після успішного входу користувачу надається доступ до інтерфейсу відповідного модуля системи.

Функціонал включає:

- вхід у систему з перевіркою облікових даних;

- захист від несанкціонованого доступу;
- контроль доступу до даних відповідно до ролі користувача.

2. Модуль управління пацієнтами

Модуль реалізує зберігання, редагування та перегляд інформації про пацієнтів, включаючи паспортні дані, історію хвороб, звернення та призначення. Реєстратор може створювати нові записи, редагувати існуючі та здійснювати пошук пацієнтів за різними критеріями.

Основні функції:

- створення та ведення електронної медичної картки;
- реєстрація звернень пацієнтів;
- перегляд історії прийомів та діагнозів;
- формування друкованих документів.

3. Модуль управління прийомами

Цей модуль використовується для планування, реєстрації та ведення прийомів у лікаря. Він забезпечує взаємодію між пацієнтом, реєстратурою та лікарем:

- запис пацієнта на прийом;
- ведення журналу прийомів;
- призначення лікування та діагнозів;
- формування висновків і довідок;
- контроль виконання прийомів за календарем.

4. Модуль обробки медичних даних

Призначений для збереження, обробки та структурованого представлення медичних записів. Він відповідає за формування таблиць і зв’язків між об’єктами

бази даних: пацієнтами, лікарями, діагнозами, послугами, направленнями. Завдяки цій структурі система забезпечує узгодженість і достовірність інформації, необхідної для подальшого аналітичного аналізу.

5. Аналітичний модуль

Аналітичний модуль є центральною інноваційною складовою системи. Він реалізує алгоритми інтелектуального аналізу даних (Data Mining) та машинного навчання (Machine Learning), що дозволяють виявляти приховані закономірності в медичних записах.

До складу аналітичного модуля входять:

- **алгоритм прогнозування (Time Series)** — прогнозує кількість відвідувань поліклініки;
- **класифікаційні алгоритми (1-Rule, Naive Bayes)** — аналізують взаємозв’язки між статтю пацієнтів, категорією послуг і результатами лікування;
- **алгоритм кластеризації (K-Means)** — групує пацієнтів за віком, статтю чи типом отриманих послуг;
- **алгоритм асоціативних правил (Apriori)** — виявляє залежності між видами послуг і діагнозами;
- **модуль KPI-аналізу** — обчислює коефіцієнти завантаженості лікарів, інтенсивності прийомів і притоку нових пацієнтів (MoM).

Результати обчислень візуалізуються у вигляді діаграм, графіків та таблиць, що дозволяє швидко оцінювати стан медичного обслуговування.

6. Модуль звітності та моніторингу

Модуль дозволяє керівництву поліклініки та аналітикам формувати звіти на основі даних, отриманих у процесі обробки та аналізу.

Звіти можуть відображати:

- кількість відвідувань за певний період;
- ефективність роботи лікарів (KPI);
- динаміку звернень та тенденції у лікуванні;
- прогнози щодо навантаження на персонал.

Такого роду модульна структура системи забезпечує цілісність, логічну взаємодію між компонентами, надійність обробки даних і створює основу для розширення функціональності шляхом інтеграції нових підсистем без порушення існуючої архітектури.

3.3. Алгоритм обробки даних

Аналітична підсистема розробленого програмного комплексу є центральною частиною інформаційно-аналітичної системи, яка забезпечує перехід від простого зберігання даних до їхнього інтелектуального аналізу. У межах системи реалізовано низку алгоритмів Data Mining та Machine Learning, що дозволяють автоматизовано виявляти закономірності, прогнозувати тенденції та оцінювати ефективність роботи поліклініки.

3.3.1. Кластеризація (алгоритм K-Means)

Алгоритм K-Means застосовується для групування пацієнтів або медичних послуг за подібними характеристиками. У контексті медичного закладу він дозволяє виділити типові групи пацієнтів, наприклад:

- за віковими категоріями;
- за статтю;

- за частотою звернень або типами отриманих послуг.

Такий підхід допомагає адміністрації краще розуміти структуру пацієнтопотоку, планувати кадрові ресурси та формувати спеціалізовані програми обслуговування.

Алгоритм працює у кілька етапів:

1. Вибір кількості кластерів *k*;
2. Ініціалізація центрів кластерів;
3. Обчислення відстаней між записами пацієнтів і центрами кластерів;
4. Перепризначення записів до найближчого центру;
5. Оновлення центрів і повторення процесу до стабілізації.

Результати кластеризації виводяться у вигляді графічних візуалізацій та звітів про склад кожного кластера.

3.3.2. Класифікація (алгоритм Naive Bayes)

Алгоритм Наївного Байєса використовується для передбачення ймовірності виникнення певних подій, наприклад встановлення діагнозу або вибору типу лікувальної послуги. На основі історичних даних система аналізує атрибути пацієнтів (вік, стать, діагноз, призначення, результати прийомів) і формує модель, здатну прогнозувати, яку послугу або лікаря буде обрано в подальшому.

Основна перевага алгоритму — його швидкість та інтерпретованість, що робить його придатним для медичних застосувань, де важливо пояснювати отримані результати. Наприклад, за результатами аналізу можна визначити, що пацієнти віком понад 50 років і чоловічої статі частіше звертаються до кардіолога або отримують послуги УЗД серця.

Отримані результати відображаються у вигляді таблиць ймовірностей та порівняльних графіків, що допомагає у підтримці клінічних рішень і плануванні медичних ресурсів.

3.3.3. Асоціативний аналіз (алгоритм Apriori)

Метою алгоритму Apriori є виявлення закономірностей, він реалізує пошук асоціативних правил між медичними послугами, діагнозами, результатами прийомів та характеристиками пацієнтів.

Алгоритм працює шляхом поетапного пошуку часто повторюваних комбінацій елементів у записах пацієнтів і формування правил на їх основі. Отримані правила можуть бути використані для:

- вдосконалення медичних протоколів;
- автоматичної рекомендації додаткових обстежень;
- оцінки ефективності певних методів лікування.

Наприклад, система може виявити, що 70% пацієнтів із діагнозом “Гіпертонія” отримували послуги “ЕКГ” та “УЗД серця”, що вказує на тісний зв’язок між цими процедурами.

3.3.4. Прогнозування часових рядів (Time Series)

Для передбачення майбутньої динаміки відвідуваності використовується метод аналізу часових рядів (Time Series Forecasting). Алгоритм працює на історичних даних про кількість прийомів за день, тиждень чи місяць і дозволяє:

- прогнозувати майбутні звернення пацієнтів;
- визначати сезонні піки навантаження;
- оптимізувати графіки роботи лікарів і реєстратури.

Це дозволяє керівництву приймати обґрунтовані управлінські рішення щодо розподілу персоналу, відкриття додаткових змін або залучення консультантів. Візуалізація прогнозів виконується у вигляді графіків, де фактичні дані порівнюються з прогнозованими значеннями.

3.3.5. KPI-аналіз (оцінювання ефективності персоналу)

Модуль KPI (Key Performance Indicators) забезпечує оцінку результативності діяльності лікарів і відділень. Для цього використовується система ключових показників ефективності, серед яких:

- кількість прийомів за певний період;
- середня тривалість консультацій;
- кількість нових пацієнтів;
- відсоток повторних звернень;
- місячна динаміка навантаження (Month-over-Month).

Система автоматично обчислює коефіцієнти ефективності та відображає їх у вигляді графічних індикаторів і діаграм. Це дає змогу керівництву оцінити продуктивність кожного лікаря та виявити потенційні зони для оптимізації роботи медичного персоналу.

Усі описані алгоритми інтегровані в єдиний аналітичний модуль, який працює поверх бази даних Microsoft SQL Server. Взаємодія між алгоритмами, сховищем даних та модулем візуалізації забезпечується через спеціальні аналітичні запити й обчислювальні процедури. Завдяки цьому система забезпечує повний цикл обробки даних — від збору та зберігання до аналізу, прогнозування й формування аналітичних звітів.

3.4. Підсумки проектування

У процесі проектування інформаційно-аналітичної системи для медичного персоналу поліклініки було сформовано повну архітектурно-аналітичну модель, яка визначає як структурну, так і функціональну організацію програмного комплексу.

На рівні архітектури було обґрунтовано вибір трирівневої моделі (Presentation, Business Logic, Data Access), що забезпечує модульність, масштабованість і стабільність роботи системи. Такий підхід дозволяє ізолювати користувацький інтерфейс, бізнес-логіку та взаємодію з базою даних, що значно спрощує супровід і подальше розширення функціоналу.

Також було приділено особливу увагу інтеграції аналітичного модуля, який забезпечує реалізацію алгоритмів обробки даних — кластеризації (K-Means), класифікації (Naive Bayes), асоціативного аналізу (Apriori), прогнозування часових рядів (Time Series) та KPI-оцінювання. Реалізація цих алгоритмів у межах єдиної системи створює основу для інтелектуальної підтримки прийняття рішень у медичному закладі.

Розроблена архітектура передбачає взаємодію між оперативною базою даних, аналітичним сховищем та модулями візуалізації, що дає змогу виконувати повний цикл обробки інформації — від збору первинних даних до отримання аналітичних звітів.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ПРОГРАМНОГО КОМПЛЕКСУ

4.1. Структура бази даних

База даних розробленої інформаційно-аналітичної системи є основою для зберігання, обробки та аналізу інформації про діяльність поліклінічного закладу. Вона побудована у середовищі Microsoft SQL Server із дотриманням принципів реляційного підходу та нормалізації даних до третьої нормальної форми (3НФ).

Архітектура бази яка представлена на Рис.4.1. орієнтована на забезпечення узгодженості, надійності та розширюваності. Кожна таблиця має чітко визначені первинні та зовнішні ключі, що формують логічні зв’язки між сутностями.

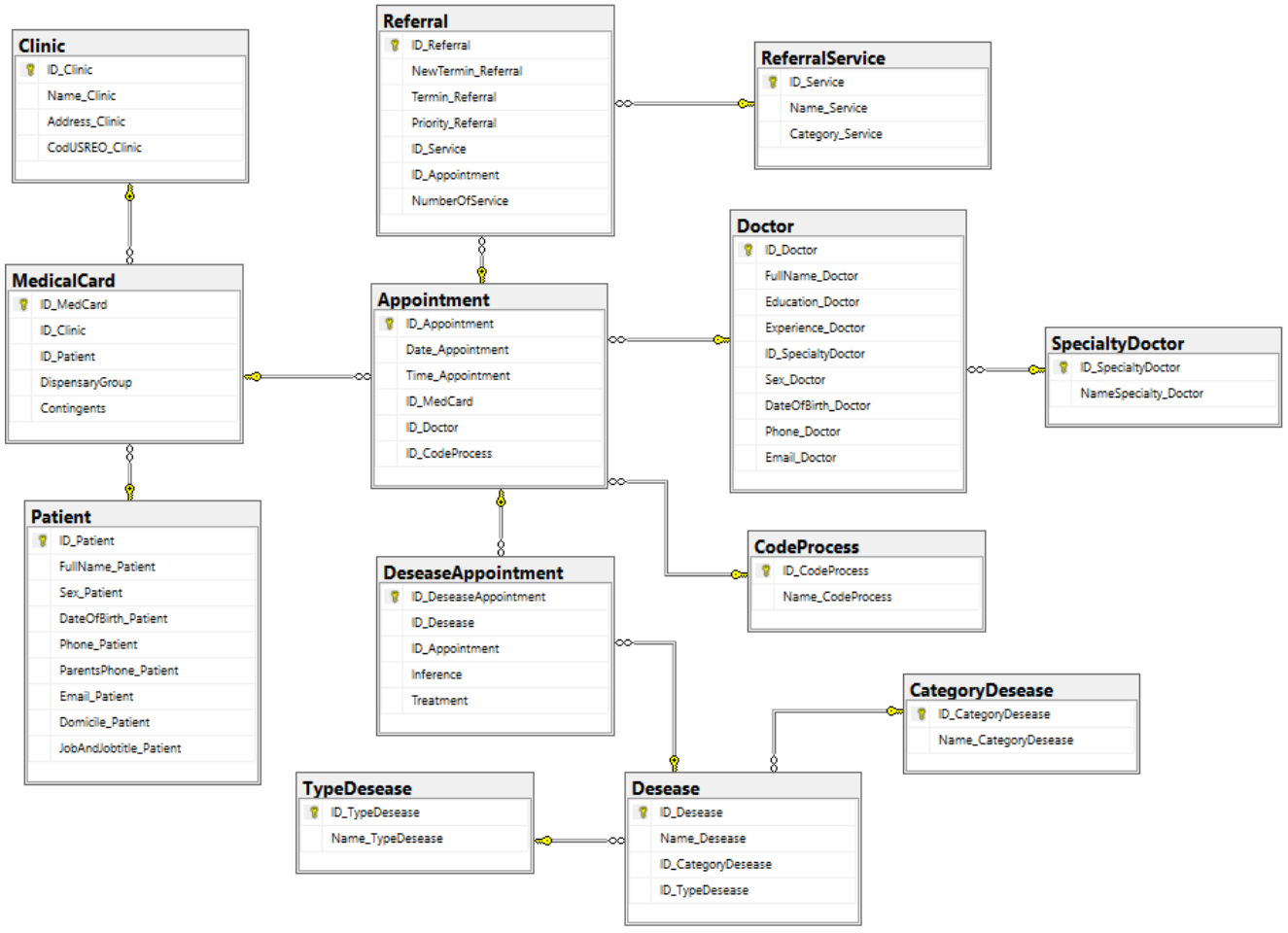

Рис.4.1. База даних.

Основні сутності бази даних

База даних складається з кількох груп таблиць, які відображають ключові об’єкти медичного процесу — пацієнтів, лікарів, захворювання, послуги, прийоми та медичні картки.

1. Пацієнти (Patient)

Зберігає персональні відомості про пацієнтів: ПІБ, стать, дату народження, адресу проживання, контактну інформацію, дані про опікунів чи батьків. Таблиця використовується як центральна точка зв’язку між медичною картою, прийомами та направленнями.

2. Медична карта (MedicalCard)

Вміщує узагальнену інформацію про історію пацієнта в межах поліклініки: групу диспансерного обліку, контингент, а також зв’язок із конкретним медичним закладом. Пов’язана з таблицями Clinic та Patient зв’язками “багато-до-одного”.

3. Медичний заклад (Clinic)

Містить загальну інформацію про поліклініку — її назву, адресу та код ЄДРПОУ. Використовується для зв’язку з медичними картками пацієнтів.

4. Лікарі (Doctor)

Відображає відомості про лікарів, які здійснюють прийоми. Містить дані про освіту, досвід роботи, спеціальність та контактну інформацію. Таблиця зв’язана з SpecialtyDoctor, Appointment, Referral і є одним із центральних елементів бази.

5. Спеціальність лікаря (SpecialtyDoctor)

Довідник, який визначає напрямки медичної діяльності (терапевт, хірург, педіатр тощо). Поле NameSpecialty_Doctor використовується як описова характеристика.

6. Прийом (Appointment)

Є однією з ключових таблиць, що відображає факт звернення пацієнта до лікаря. Вона пов’язує між собою пацієнта, лікаря, медичну картку та процеси лікування.

7. Класифікація процесів (CodeProcess)

Містить класифікатор процесів, що відбуваються під час прийому (обстеження, консультація, діагностика, лікування тощо). Використовується для типізації прийомів і формування статистики за видами медичних послуг.

8. Захворювання (Desease)

Містить перелік усіх можливих діагнозів, що можуть бути встановлені пацієнтам. Зв’язана з таблицями CategoryDesease (категорія захворювання) і TypeDesease (тип захворювання).

9. Категорія та тип захворювань

- CategoryDesease — групує захворювання за клінічними напрямками (інфекційні, хронічні, серцево-судинні тощо).
- TypeDesease — визначає характер перебігу хвороби (гостра, хронічна, спадкова).

10. Захворювання на прийомі (DeseaseAppointment)

Таблиця-зв’язок між Appointment та Desease, яка зберігає результати обстеження, висновки та призначення лікування. Вона є фактичною фіксацією результатів візиту до лікаря.

11. Направлення (Referral)

Фіксує факти направлення пацієнта до інших лікарів або на додаткові послуги. Зв’язана з таблицями Appointment та ReferralService.

12. Послуги за направленням (ReferralService)

Містить перелік можливих послуг, які можуть бути вказані в направленні (наприклад, лабораторне дослідження, рентген, фізіотерапія).

Загальна характеристика БД:

- Усі таблиці містять первинні ключі (Primary Keys) для ідентифікації записів.
- Між таблицями встановлено зовнішні ключі (Foreign Keys), які забезпечують логічну узгодженість даних.
- Система підтримує каскадне оновлення та видалення записів для запобігання порушенню зв’язків.
- Дані розподілені за функціональними групами, що забезпечує гнучкість і масштабованість під час розширення системи.

4.2. Сховище даних та аналітична модель

Для підтримки аналітичних функцій розробленої інформаційно-аналітичної системи створено сховище даних (Data Warehouse), побудоване за принципом зіркоподібної схеми (Star Schema). Таке рішення забезпечує оптимальне зберігання та швидку обробку великих обсягів даних, дозволяючи здійснювати

багатовимірний аналіз і побудову звітів за різними параметрами діяльності поліклінічного закладу.

Сховище даних створене на основі даних із оперативної бази шляхом виконання ETL-процесу (Extract – Transform – Load), який передбачає:

- **Extract** — вибірку необхідних даних із таблиць пацієнтів, лікарів, прийомів, послуг і діагнозів;
- **Transform** — очищення, агрегування, фільтрацію та узгодження форматів даних;
- **Load** — завантаження у відповідні фактні та вимірні таблиці сховища.

Завдяки цьому забезпечується централізоване зберігання інформації для подальшого аналітичного опрацювання, побудови звітів та прогнозування тенденцій у діяльності медичного закладу.

Модель сховища даних складається з однієї фактної таблиці — ReceptionFact та п’яти таблиць вимірів (dimension tables): PatientDim, DoctorDim, DiseaseDim, ServiceDim, TimeDim. Центральною ланкою є фактна таблиця ReceptionFact, яка зберігає агреговані дані про прийоми пацієнтів, а всі інші таблиці вимірів деталізують аналітичні аспекти — хто, коли, де та з яким результатом здійснював прийом.

Таке поєднання забезпечує гнучкість при побудові аналітичних звітів і дозволяє ефективно виконувати запити будь-якої складності — від базових статистичних звітів до глибокого аналітичного аналізу медичних процесів.

На рисунку 4.2 зображено логічну структуру сховища даних, яка побудована за зіркоподібною схемою.

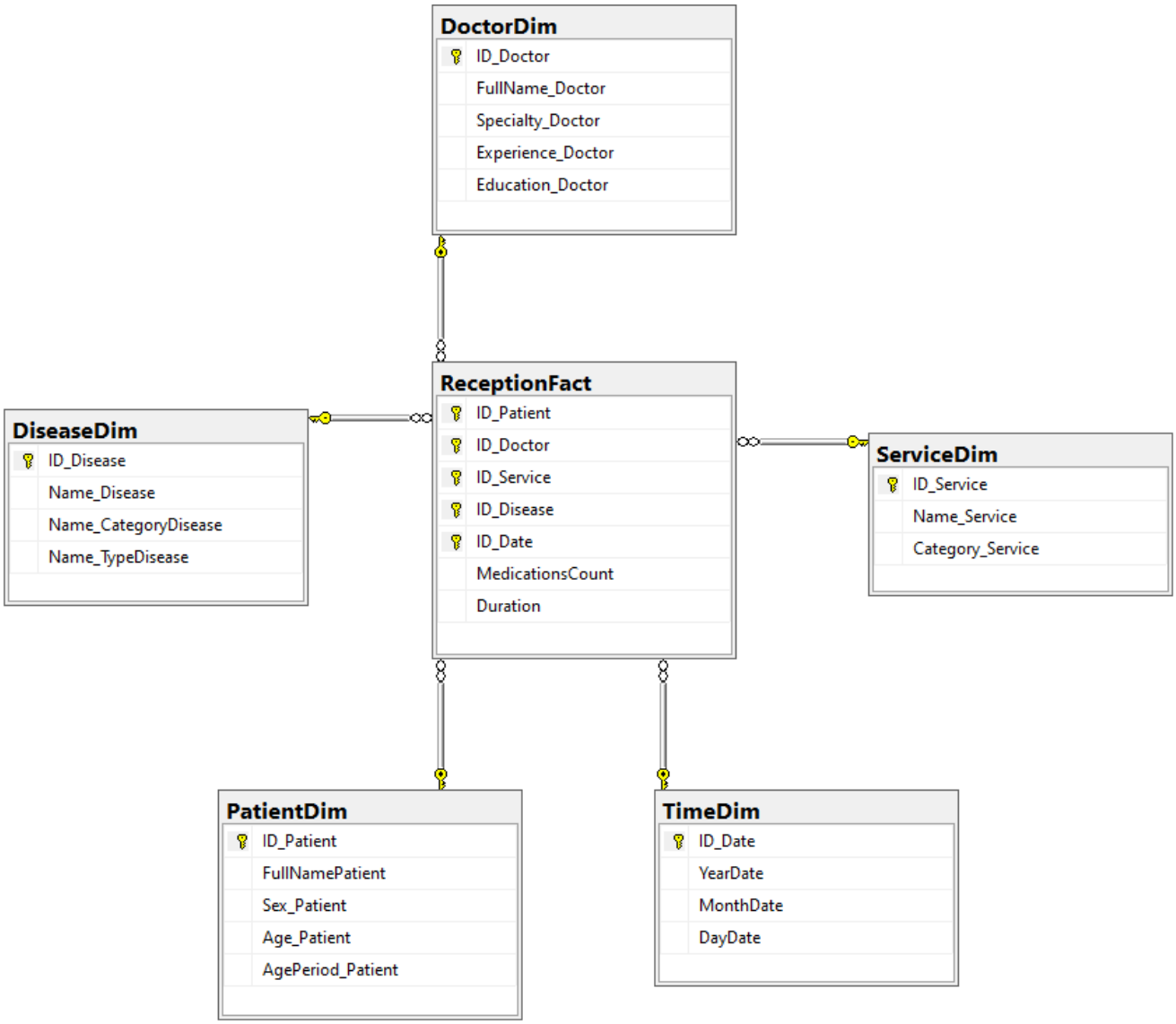

Рис. 4.2. Сховище даних

По зображенню можна побачити, що у центрі розташована фактна таблиця ReceptionFact, яка акумулює ключові показники діяльності поліклініки — інформацію про прийоми пацієнтів, лікарів, призначені послуги, встановлені діагнози, кількість медикаментів і тривалість прийомів.

Навколо таблиці фактів розміщено п’ять таблиць вимірів, що деталізують окремі аспекти аналітичних даних:

- **DoctorDim** — містить інформацію про лікарів, їхній досвід, освіту та спеціалізацію;
- **PatientDim** — зберігає демографічні характеристики пацієнтів (вік, стать, вікова група);
- **DiseaseDim** — описує діагнози, типи та категорії захворювань;
- **ServiceDim** — містить відомості про надані медичні послуги;
- **TimeDim** — визначає часові параметри аналізу (день, місяць, рік).

Таке структурування забезпечує можливість багатовимірного аналізу даних, наприклад, оцінювання кількості прийомів за певний період, визначення найпоширеніших захворювань, аналіз навантаження на лікарів чи вивчення ефективності надання послуг.

4.3. Реалізація основних форм системи.

Графічний інтерфейс користувача (GUI) інформаційно-аналітичної системи реалізовано з використанням технології Windows Forms у середовищі C# .NET Framework. Інтерфейс побудований за принципами сучасного UX-дизайну, що забезпечує зручність користування, візуальну привабливість і логічну структуризацію функціональних модулів. Основна увага приділена простоті навігації, інтуїтивному розташуванню елементів керування та дотриманню єдиного стилю оформлення в усіх формах системи.

4.3.1. Форма авторизації.

На рис. 4.3.1 представлено форму авторизації користувача. Вона є початковим етапом взаємодії із системою та забезпечує перевірку прав доступу до функціоналу.

Користувач вводить логін і пароль, після чого виконується перевірка автентичності через базу даних. У разі успішної перевірки система відкриває головне меню відповідно до ролі користувача (лікар, реєстратор, аналітик, керівник). Інтерфейс побудовано з використанням темного фону та акцентних фіолетових елементів, що підвищує читабельність і знижує навантаження на зір при тривалій роботі.

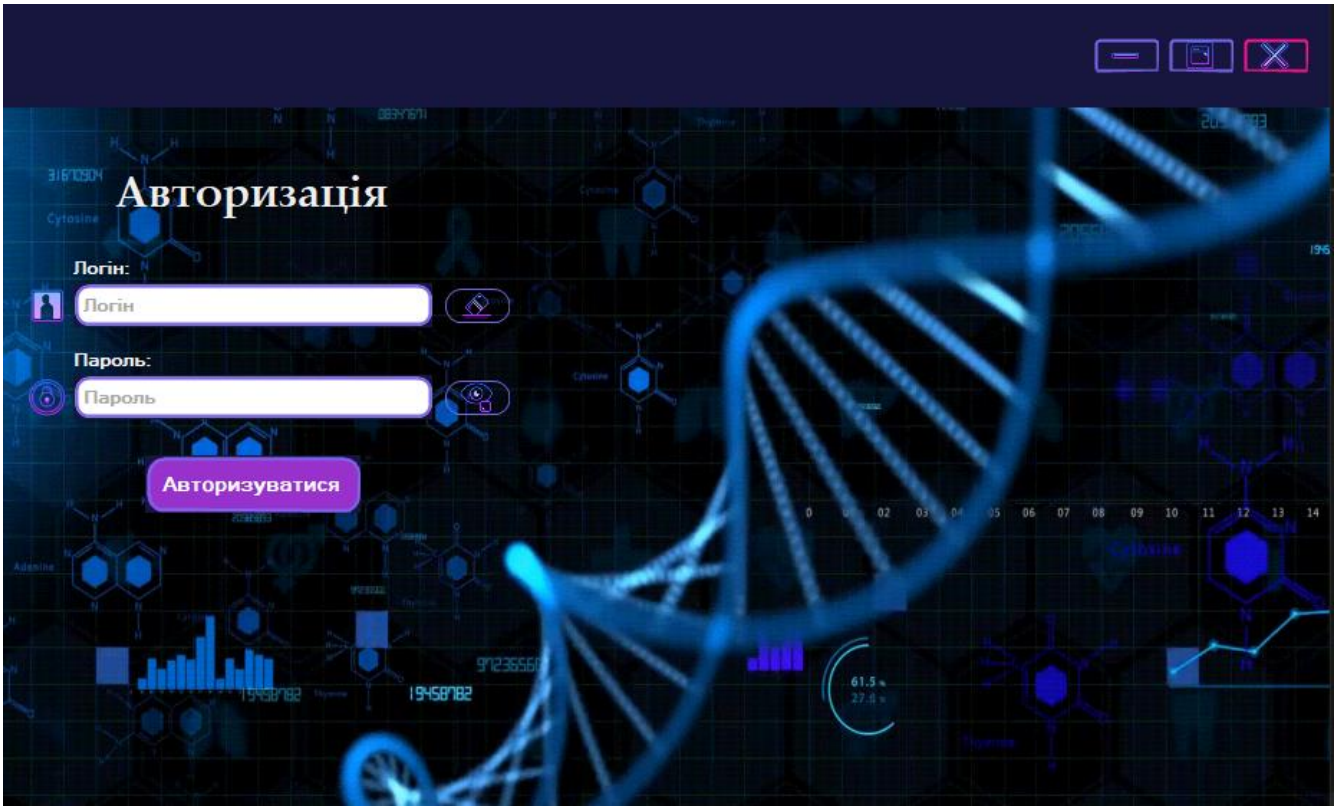

Рис. 4.3.1. Авторизація користувача.

4.3.2. Головне меню системи.

Після входу користувач потрапляє до головного меню (наприклад, рис. 4.3.2. Головне меню лікаря та 4.3.3. Головне меню реєстратора), яке адаптується відповідно до ролі користувача:

- **Головне меню лікаря** містить розділи: “Графік прийому”, “Прийоми”, “Заключення”, “Направлення”, “Друк направлення”.

- **Головне меню реєстратора** включає: “Графік прийому”, “Пацієнти”, “Медичні картки”, “Друк медичних карток”.
- **Головне меню аналітика** містить: “Загальна статистика”, “Аналіз пацієнтів, лікарів та захворювань”, “Прогнозування”, “KPI-оцінювання”, “Формування звітів”.
- **Головне меню адміна** має доступ до всього функціоналу.

Меню реалізовано у вигляді вертикальної панелі навігації з інтерактивними іконками. У центральній частині вікна розміщується інформаційна панель із поточним часом і датою, що оновлюються в реальному часі.

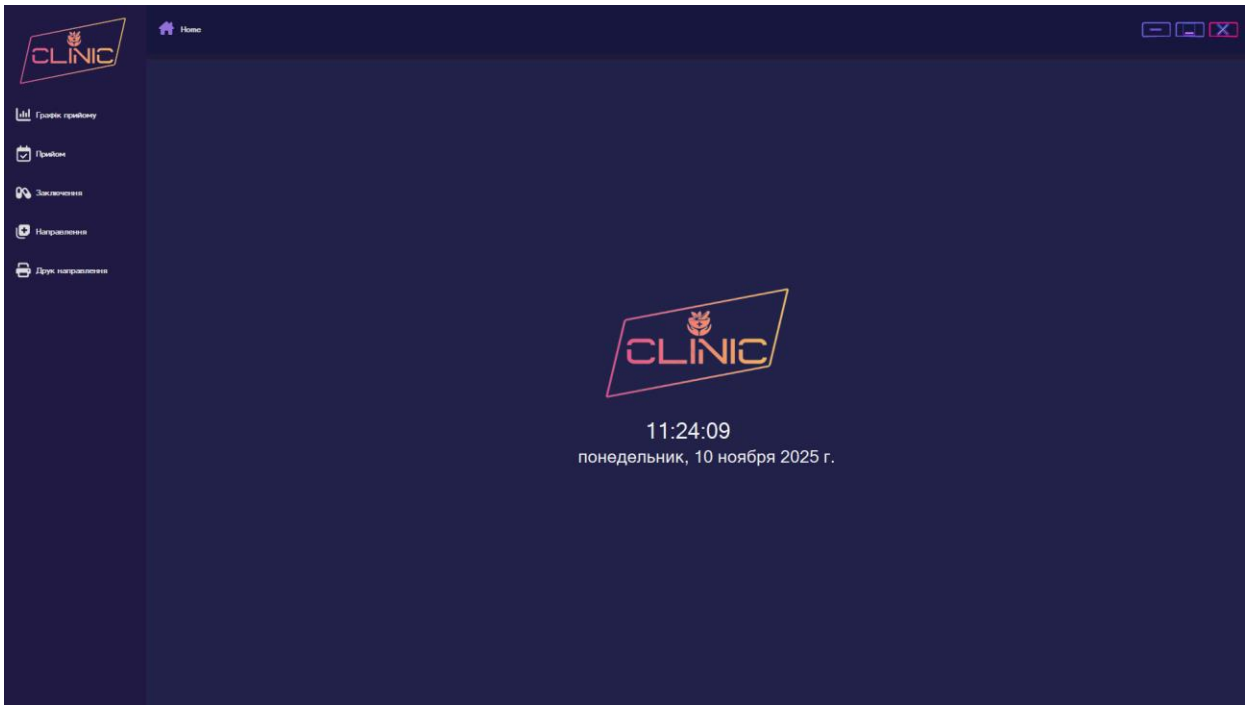

Рис. 4.3.2. Головне меню лікаря.

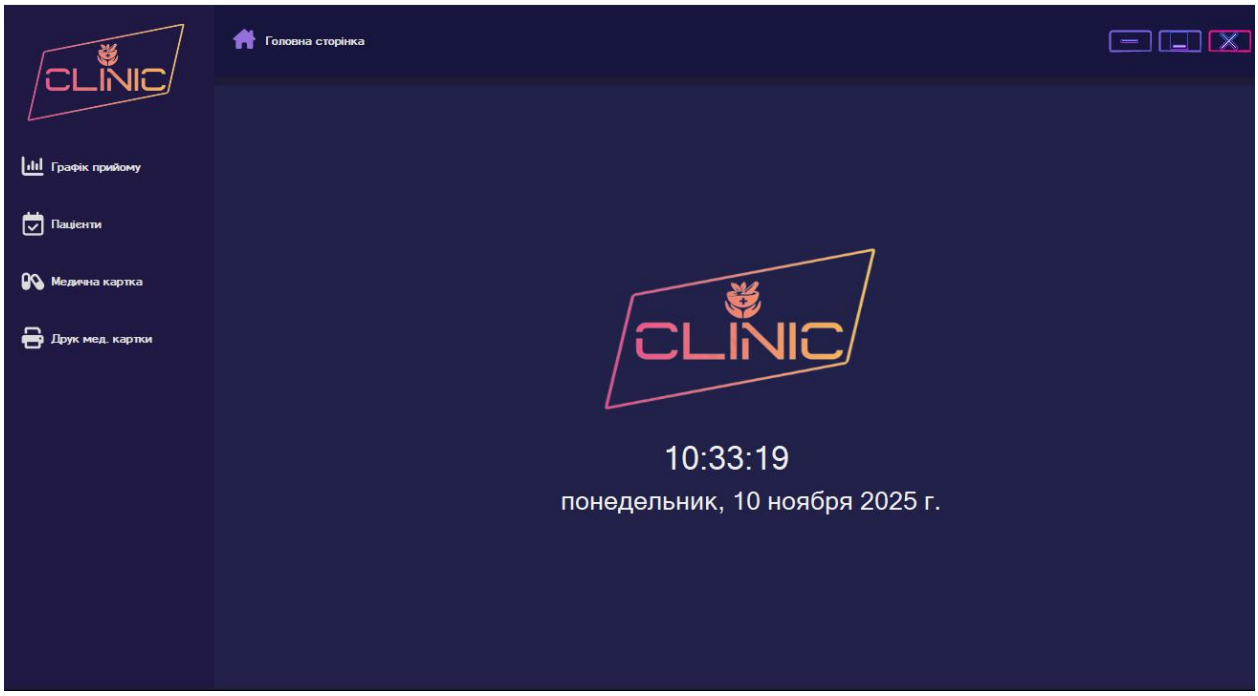

Рис. 4.3.3. Головне меню реєстратора.

4.3.3. Форми роботи з даними.

Основні функціональні модулі системи реалізовано через форми роботи з даними (частину з них зображено та описано на рис. 4.3.4–4.3.6), які забезпечують:

- перегляд, додавання, редагування та видалення інформації про пацієнтів, лікарів, прийоми, медичні картки та направлення;
- фільтрацію та пошук записів за заданими параметрами;
- збереження змін у базі даних через механізм двосторонньої інтеграції з MS SQL Server.

Форма “Інформація про пацієнта” (рис. 4.3.4) містить поля для введення персональних та контактних даних, статі, дати народження, місця проживання та професії пацієнта. Ліва панель містить форму введення, тоді як права — таблицю зі списком усіх пацієнтів із можливістю швидкої навігації між сторінками.

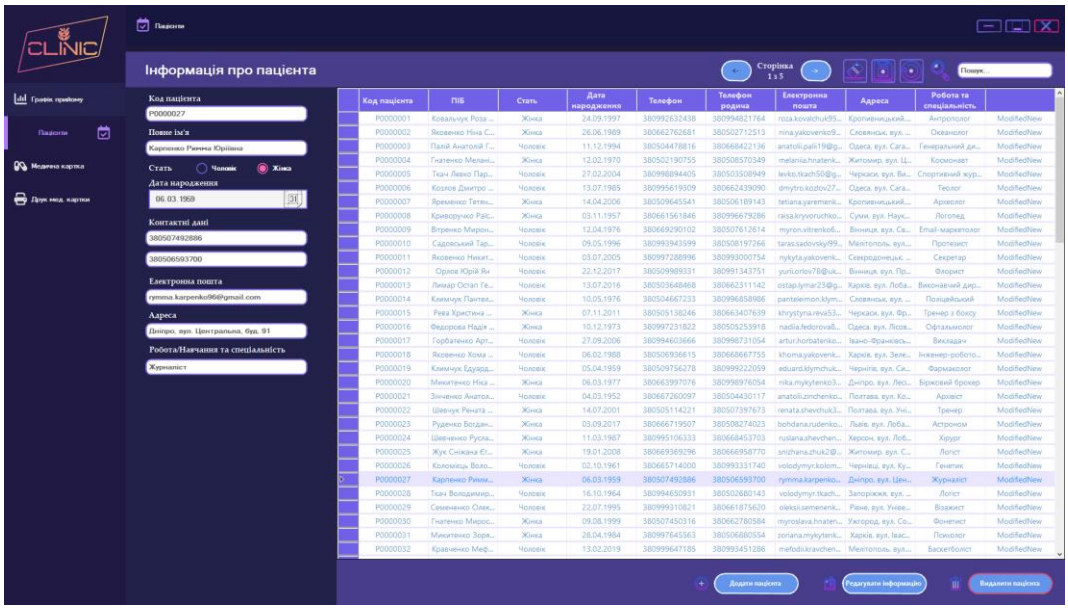

Рис. 4.3.4. Форма «Інформація про пацієнтів» / Панель реєстратора.

Форма “Інформація про прийоми” (рис. 4.3.5) дозволяє відображати інформацію про проведені та заплановані прийоми, включно з лікарем, пацієнтом, спеціальністю, датою й часом прийому. Передбачена можливість створення нового запису або його видалення.

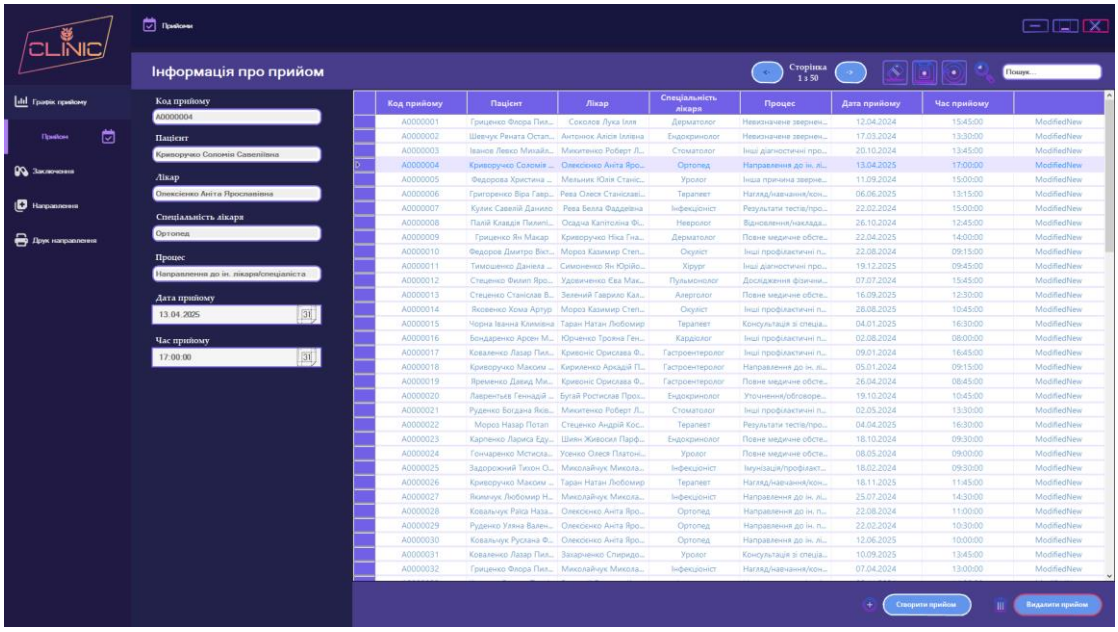

Рис. 4.3.5. Форма «Інформація про прийоми» / Панель лікаря.

Форма “Інформація про спеціалістів” (рис. 4.3.6) дає змогу керувати довідником лікарів — редагувати їхні особисті дані, спеціалізацію, досвід, освіту, контактну інформацію. Система підтримує можливість швидкого пошуку лікаря за ПІБ або спеціальністю.

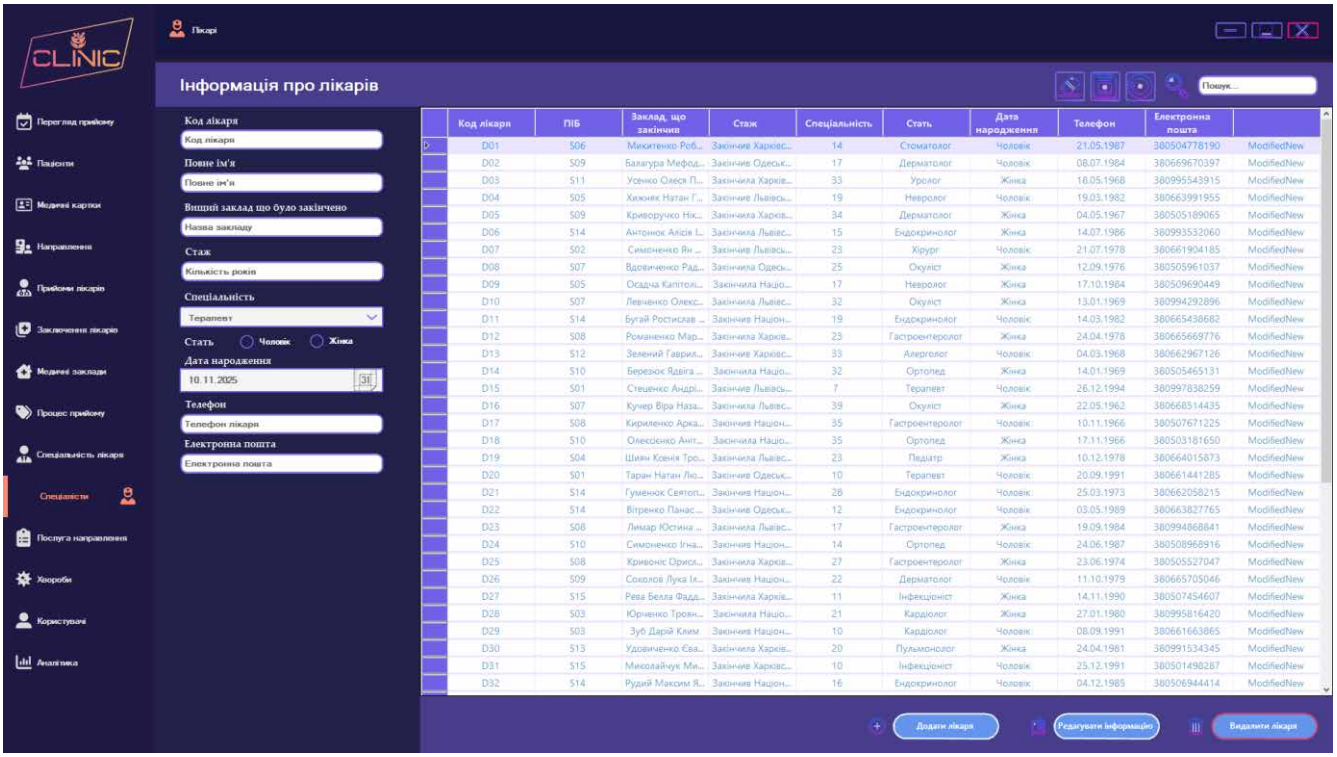

Рис. 4.3.6. Форма «Інформація про спеціалістів» / Панель адміна.

4.3.4. Модальні вікна та повідомлення.

Для спрощення введення нових даних передбачено модальні вікна — наприклад, форма “Додавання пацієнта” (рис. 4.3.7), у якій користувач заповнює обов’язкові поля, після чого система створює запис у базі даних. У разі успішного виконання операції з’являється інформаційне сповіщення (рис. 4.3.8), що підтверджує збереження даних.

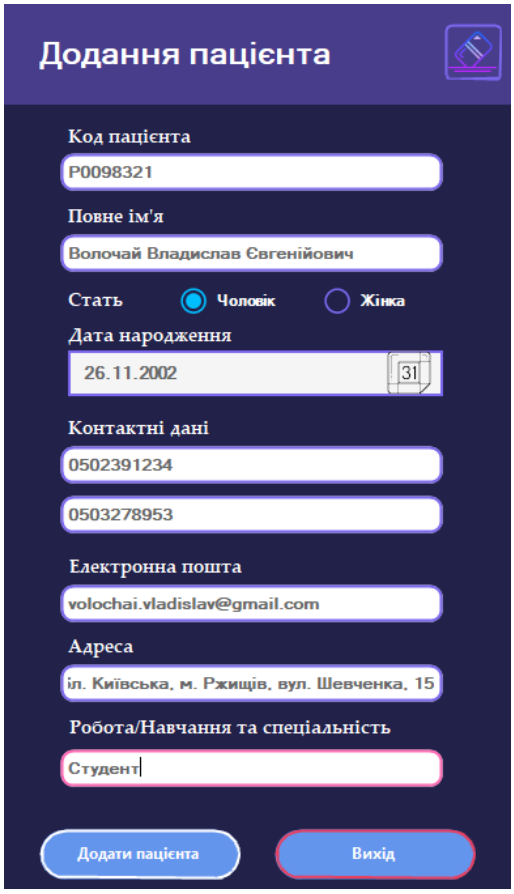

Рис. 4.3.7. Форма «Додання пацієнта».

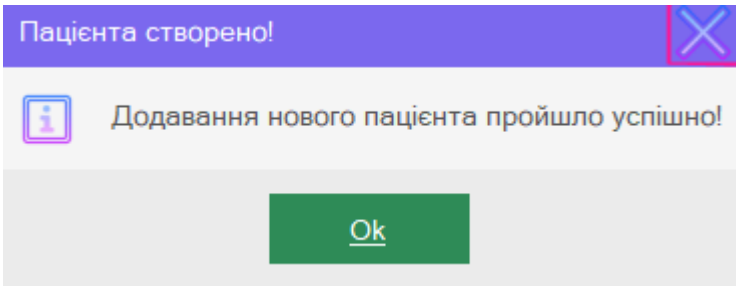

Рис. 4.3.8. Форма «Інформаційне сповіщення».

4.3.5. Форма друку.

Форма “Інформація про направлення” (рис. 4.3.9) надає можливість створювати направлення на обстеження або консультацію та друкувати їх у стандартизованому форматі. Відображається попередній перегляд PDF-документа

із зазначенням усіх реквізитів — ПІБ пацієнта, лікаря, терміну дії, послуги, категорії та кількості процедур.

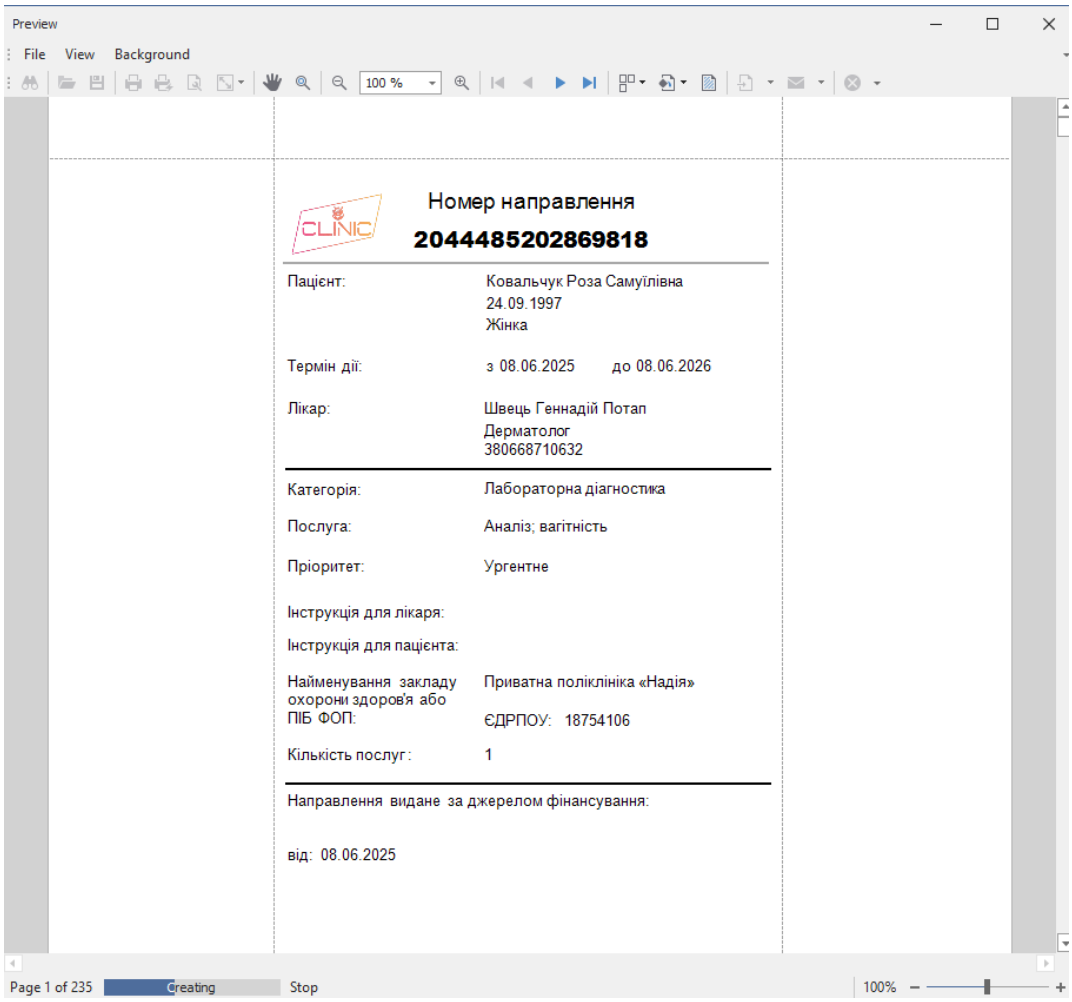

Рис. 4.3.9. Форма «Друк направлення».

4.4. Реалізація аналітичного модуля

Аналітичний модуль є ключовим компонентом інформаційно-аналітичної системи, який забезпечує збір, обробку та візуалізацію статистичних показників діяльності поліклініки. Основна мета модуля — надання керівництву та аналітикам інструментів для оцінки ефективності роботи персоналу, аналізу звернень пацієнтів, прогнозування динаміки відвідуваності та виявлення закономірностей у наданні медичних послуг.

Інтерфейс модуля реалізовано з використанням інтерактивних таблиць і графічних звітів, які автоматично формуються на основі даних із бази аналітичної системи. Кожен розділ містить власний набір показників (KPI) і візуалізацій.

4.4.1. Ефективність роботи персоналу

Цей розділ дає змогу аналізувати завантаженість лікарів**,** інтенсивність прийомів**,** повноту заповнення лікарських записів та розподіл звернень за спеціальностями**.**

Система автоматично обчислює KPI-коефіцієнт завантаженості та формує розподіл персоналу за категоріями:

- **оптимальне навантаження** (середня кількість прийомів відповідає нормі),
- **перевантаження** (перевищення норми > 20 %),
- **недовантаження** (менше – 20 %).

На рис. 4.4.1. зображено «Аналіз завантаженості лікарів». Він демонструє кількість прийомів кожного спеціаліста у порівнянні із середнім показником за спеціальністю, а також розподіл лікарів за рівнем навантаження (оптимальне, перевантажене, недостатнє).

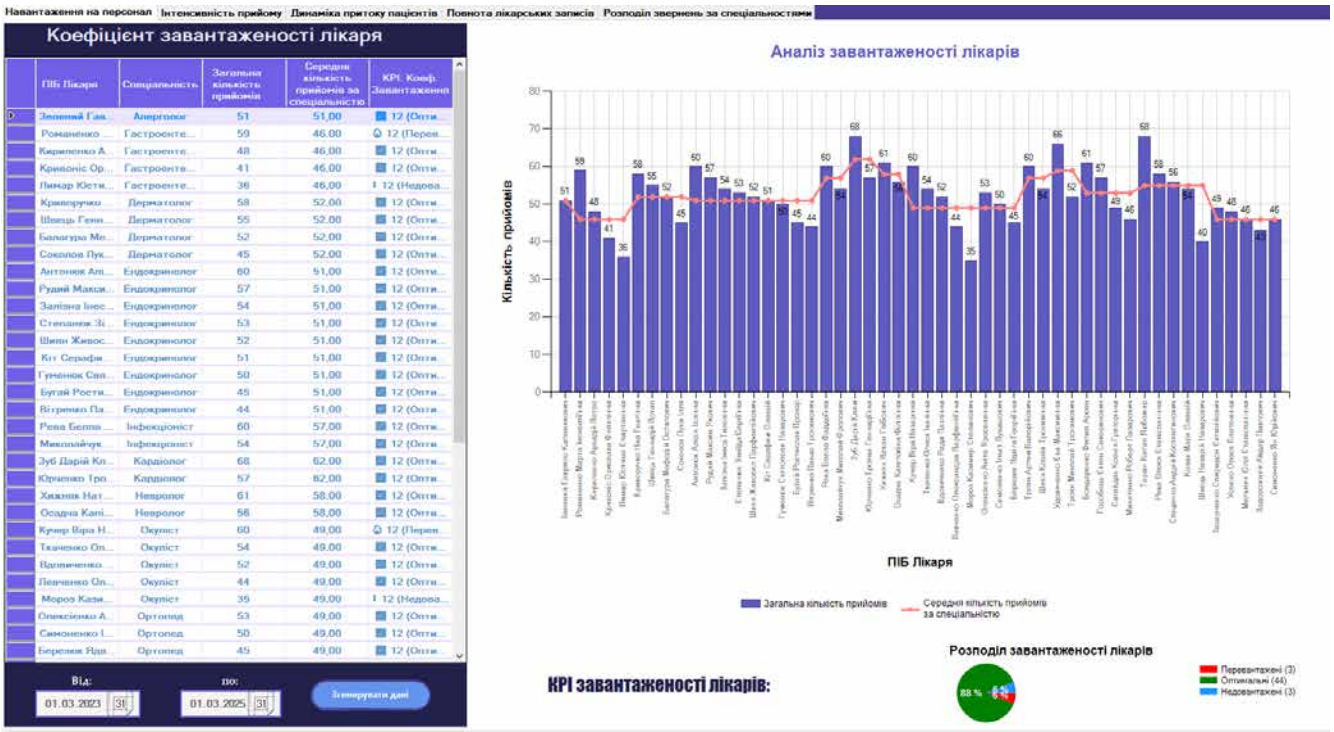

Рис. 4.4.1. Коефіцієнт завантаженості лікаря.

На рис. 4.4.2. показано «Інтенсивність прийомів». Вона відображає повну динаміку навантаження протягом робочого дня та допомагає визначити пікові години.

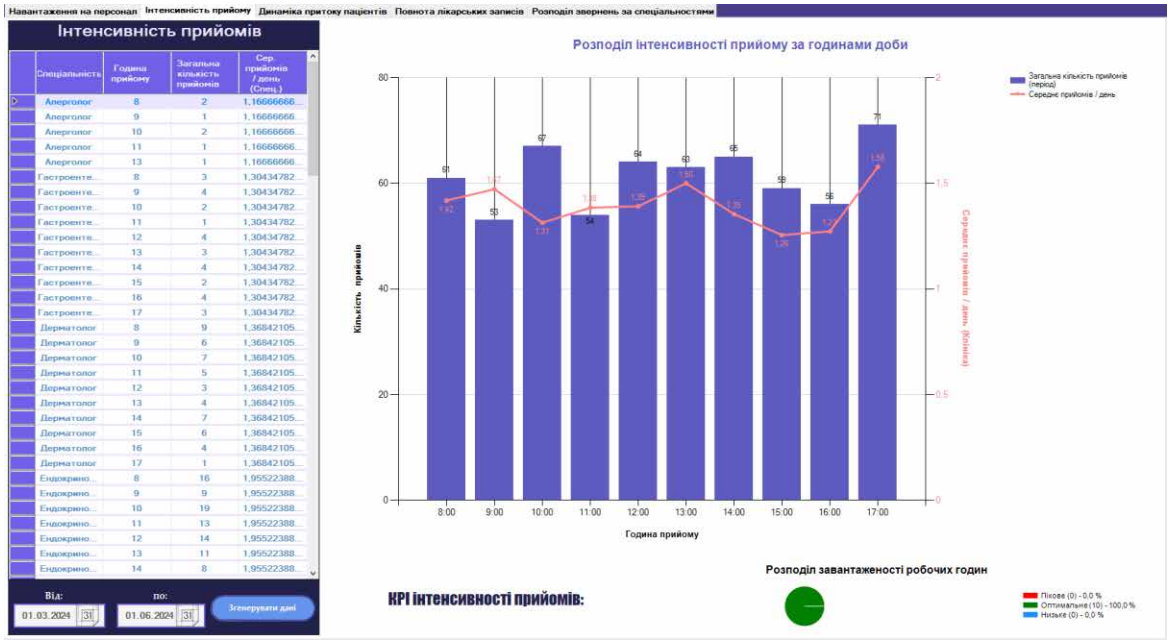

Рис. 4.4.2. Коефіцієнт інтенсивності прийомів.

Повнота лікарських записів зображена на рис. 4.4.3. На ній показано відсоток пропущених записів у медичних картках; показник використовується для оцінки якості ведення документації.

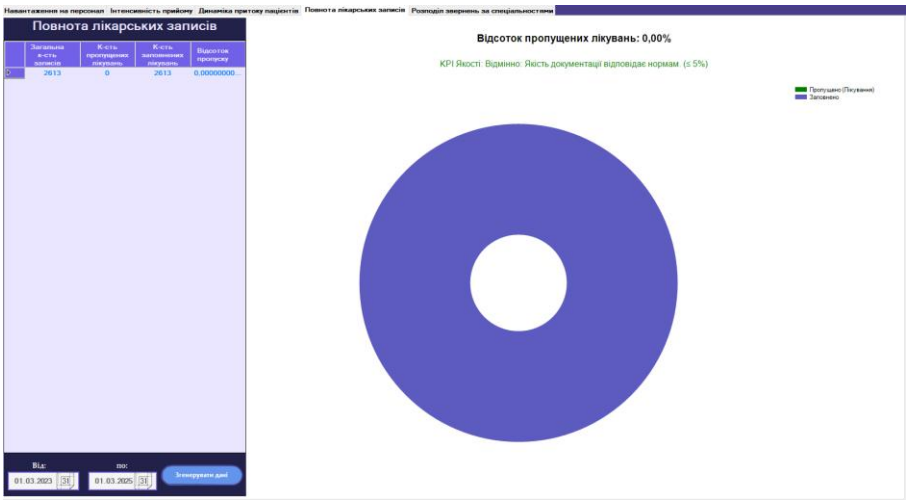

Рис. 4.4.3. Коефіцієнт повноти лікарських записів.

На рис. 4.4.4. зображено «Розподіл звернень за спеціальностями». Він показує кількість прийомів за напрямами медицини, що допомагає ідентифікувати найбільш затребувані послуги.

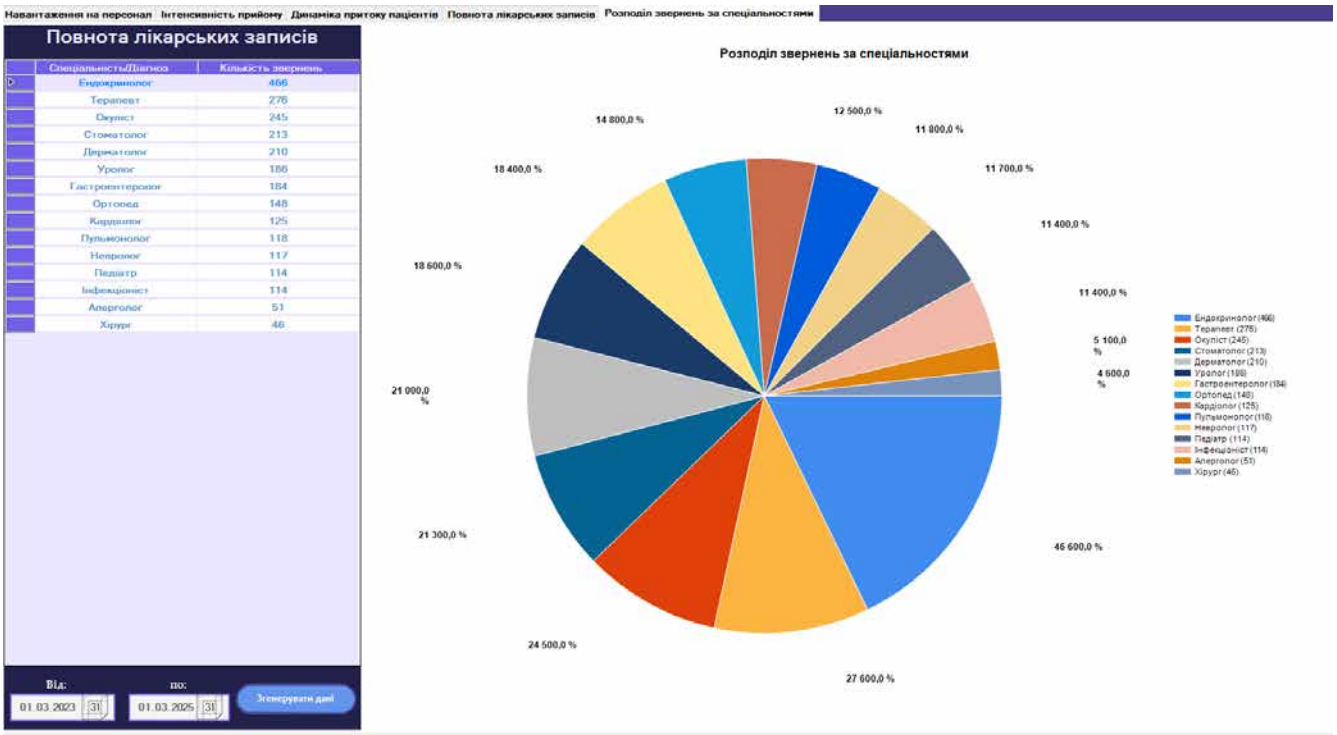

Рис. 4.4.4. Розподіл звернень за спеціальностями.

4.4.2. Аналіз попиту на послуги пацієнтами

У цьому підрозділі реалізовано порівняння ймовірностей отримання послуг різними групами пацієнтів за статтю та віком. Модуль дозволяє виявити, які категорії послуг більш характерні для чоловіків чи жінок, а також аналізувати загальні тенденції.

На рис 4.4.5. представлена таблиця ймовірностей отримання лікувальних послуг пацієнтами за алгоритмом 1-Rule. А на рис. 4.4.6. зображено їх порівняння за допомоги їх візуалізації.

Поділ даних на класи (“Так” – лікувальні, “Ні” – діагностичні чи інші послуги).

Стать	Категорія_послуг	Менше_середньог	Більше_середньог	Всього	Клас	Ймовірність
Чоловік	Візуалізація	2388	37	2425	Ні	98,47
Чоловік	Функціональна ...	1652	80	1732	Ні	95,38
Чоловік	Хірургічна проце...	10564	8601	19165	Ні	55,12
Жінка	Лабораторна ді...	16611	5827	22438	Ні	74,03
Жінка	Візуалізація	2420	50	2470	Ні	97,98
Жінка	Хірургічна проце...	177	4071	4248	Так	95,83
Жінка	Функціональна ...	2958	599	3557	Ні	83,16
Жінка	Діагностична пр...	2512	2316	4828	Ні	52,03
Чоловік	Діагностична пр...	1968	2095	4063	Так	51,56
Чоловік	Лабораторна ді...	4454	214	4668	Ні	95,42
Чоловік	Адміністративна	1315	22	1337	Ні	98,35
Чоловік	Процедура	1198	1310	2508	Так	52,23
Чоловік	Фізіотерапія/Ре...	7579	2550	10129	Ні	74,82
Жінка	Консультація	5002	1167	6169	Ні	81,08
Жінка	Адміністративна	1304	14	1318	Ні	98,94
Жінка	Процедура	927	1234	2161	Так	57,1
Жінка	Маніпуляція/Пе...	984	16	1000	Ні	98,4
Жінка	Фізіотерапія/Ре...	1713	29	1742	Ні	98,34
Чоловік	Маніпуляція/Пе...	989	14	1003	Ні	98,6
Чоловік	Консультація	2581	48	2629	Ні	98,17
Чоловік	Лікувально-діаг...	77	134	211	Так	63,51

Рис. 4.4.5. Таблиця ймовірностей отримання лікувальних послуг пацієнтами.

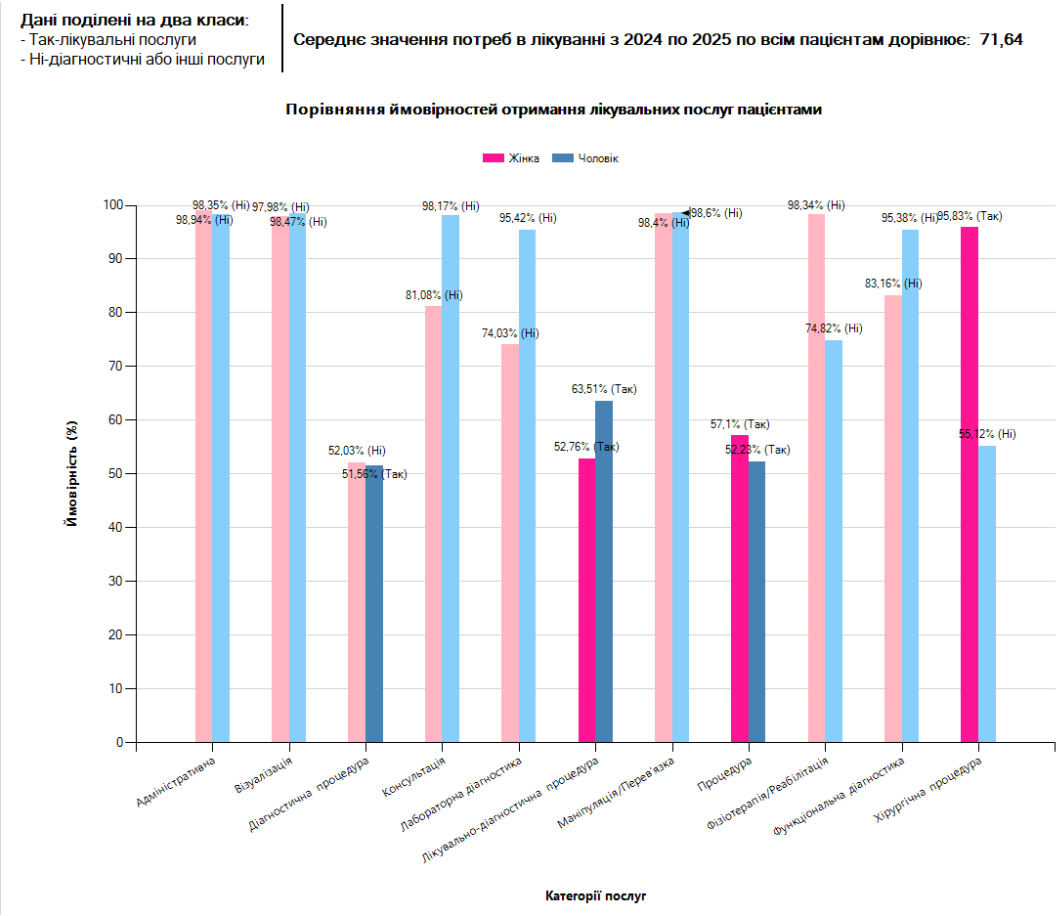

Рис. 4.4.6. Діаграма ймовірностей отримання лікувальних послуг пацієнтами.

На рис. 4.4.7. – 4.4.8. зображено таблицю та діаграму яка відображає яка стать зазвичай потребує найбільше саме цієї послуги. Визначення відбувалось за допомогою алгоритму Naive Bayes:

Категорія	Ймовірність_Жінк	Ймовірність_Чолс	Найімовірніша_ста	Перевага
Візуалізація	0,5046	0,4954	Жінка	0,92 %
Функціональна ...	0,6725	0,3275	Жінка	34,49 %
Хірургічна проце...	0,1815	0,8185	Чоловік	63,71 %
Лабораторна ді...	0,8278	0,1722	Жінка	65,55 %
Діагностична пр...	0,543	0,457	Жінка	8,6 %
Адміністративна	0,4964	0,5036	Чоловік	0,72 %
Процедура	0,4629	0,5371	Чоловік	7,43 %
Фізіотерапія/Ре...	0,1468	0,8532	Чоловік	70,64 %
Консультація	0,7011	0,2989	Жінка	40,23 %
Маніпуляція/Пе...	0,4993	0,5007	Чоловік	0,15 %
Лікувально-діаг...	0,4854	0,5146	Чоловік	2,91 %

Рис. 4.4.7. Порівняльна таблиця ймовірності послуг за статтю пацієнтів.

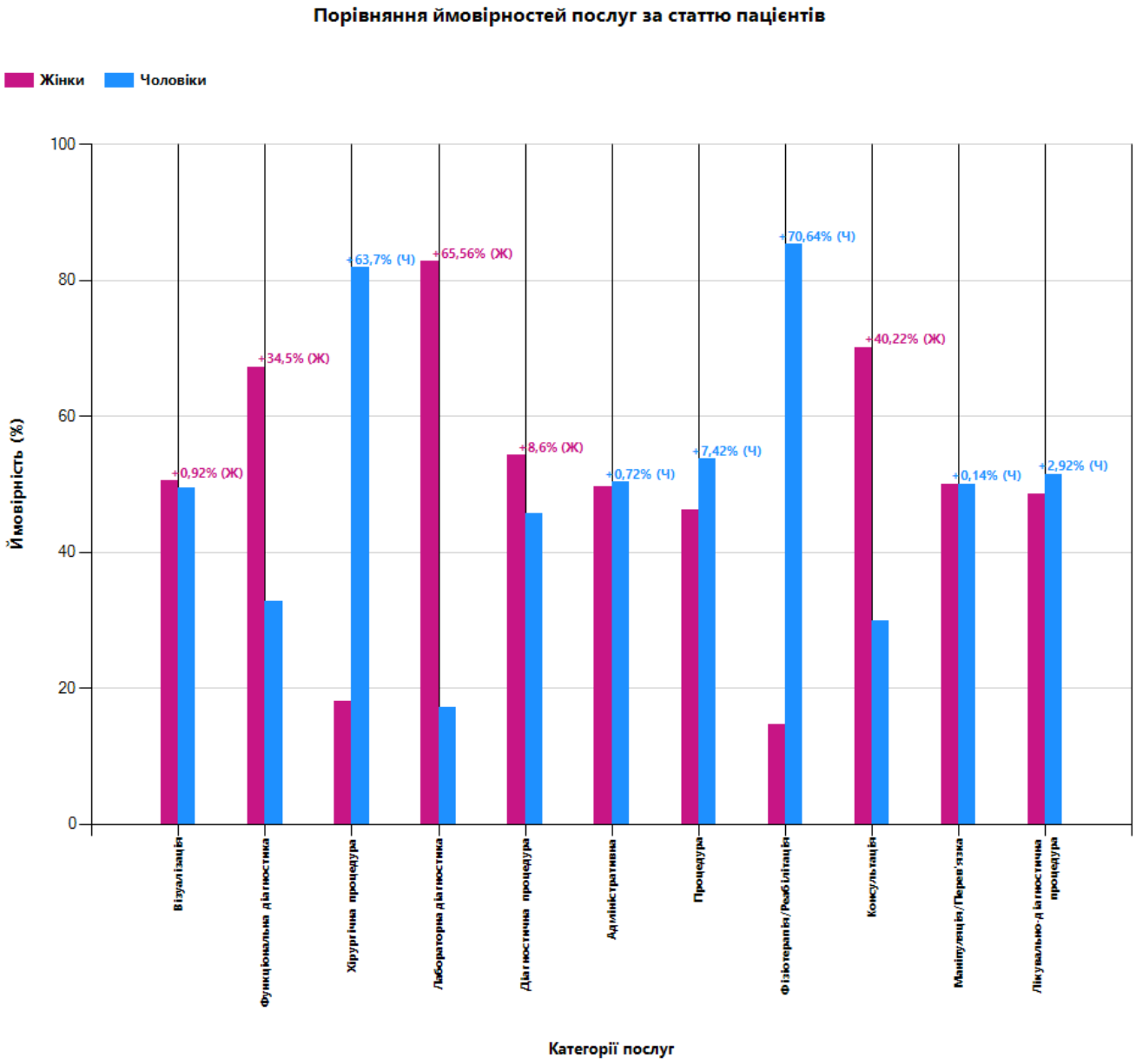

Рис. 4.4.8. Діаграма для візуалізації ймовірності послуг за статтю пацієнтів.

4.4.3. Якість обслуговування

Цей розділ містить модуль прогнозування та оцінки тенденцій у динаміці відвідуваності.

Зображення 4.4.9. відображає прогнозування динаміки відвідуваності. Це прогнозування демонструє реальні та прогнозовані показники кількості візитів пацієнтів із виділенням мінімальних і максимальних меж.

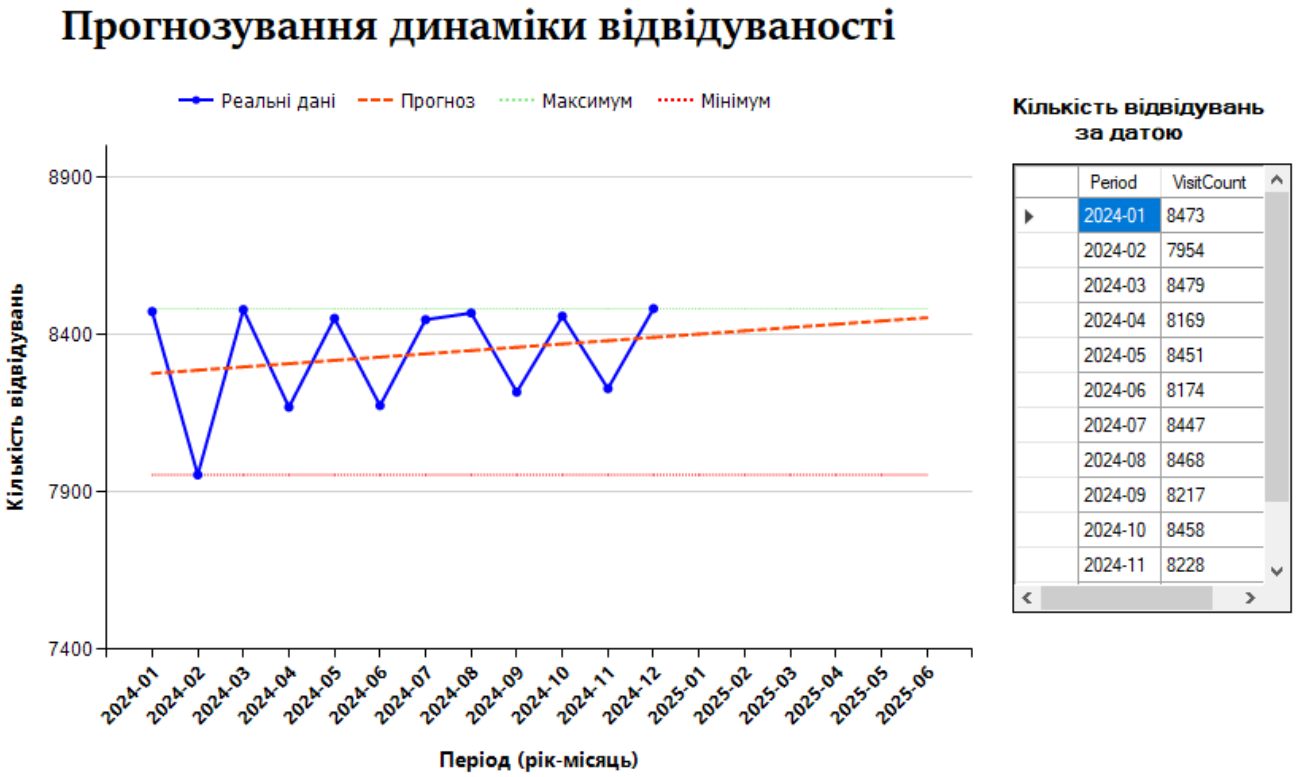

Рис. 4.4.9. Прогнозування динаміки відвідуваності.

4.4.4. Аналіз кореляцій та залежностей

Модуль забезпечує пошук асоціативних правил між категоріями медичних послуг, що дозволяє визначати типові комбінації процедур і досліджень.

На рис. 4.4.10. показано асоціативні залежності між категоріями медичних послуг — таблиця підтримки (Support), достовірності (Confidence) та коефіцієнта підсилення (Lift) для виявлених закономірностей.

Асоціативні залежності між категоріями медичних послуг

Antecedent	Consequent	Support	Confidence	Lift
Візуалізація - Інфекції	Функціональна діагностика - Симптоми/скарги	0,021	0,16	1,14
Візуалізація - Інфекції	Хірургічна процедура - Інфекції	0,045	0,347	0,875
Візуалізація - Інфекції	Хірургічна процедура - Симптоми/скарги	0,049	0,375	0,932
Візуалізація - Інфекції	Хірургічна процедура - Інші діагнози	0,032	0,249	0,89
Візуалізація - Інфекції	Хірургічна процедура - Травми	0,011	0,087	0,914
Візуалізація - Інфекції	Хірургічна процедура - Вроджені вади	0,005	0,039	0,797
Візуалізація - Інфекції	Хірургічна процедура - Новоутворення	0,036	0,28	0,908
Функціональна діагностика - Симптоми/скарги	Хірургічна процедура - Інфекції	0,044	0,312	0,787
Функціональна діагностика - Симптоми/скарги	Хірургічна процедура - Симптоми/скарги	0,045	0,32	0,794
Функціональна діагностика - Симптоми/скарги	Хірургічна процедура - Інші діагнози	0,03	0,216	0,773
Функціональна діагностика - Симптоми/скарги	Хірургічна процедура - Травми	0,009	0,065	0,68
Функціональна діагностика - Симптоми/скарги	Хірургічна процедура - Вроджені вади	0,005	0,036	0,736
Функціональна діагностика - Симптоми/скарги	Хірургічна процедура - Новоутворення	0,034	0,241	0,781
Хірургічна процедура - Інфекції	Хірургічна процедура - Симптоми/скарги	0,233	0,588	1,461
Хірургічна процедура - Інфекції	Хірургічна процедура - Інші діагнози	0,175	0,442	1,58
Хірургічна процедура - Інфекції	Хірургічна процедура - Травми	0,059	0,15	1,57
Хірургічна процедура - Вроджені вади	Хірургічна процедура - Інфекції	0,032	0,646	1,63
Хірургічна процедура - Інфекції	Хірургічна процедура - Новоутворення	0,188	0,474	1,535
Хірургічна процедура - Інші діагнози	Хірургічна процедура - Симптоми/скарги	0,174	0,623	1,547
Хірургічна процедура - Симптоми/скарги	Хірургічна процедура - Травми	0,064	0,158	1,654
Хірургічна процедура - Вроджені вади	Хірургічна процедура - Симптоми/скарги	0,032	0,656	1,629
Хірургічна процедура - Новоутворення	Хірургічна процедура - Симптоми/скарги	0,189	0,612	1,521
Хірургічна процедура - Інші діагнози	Хірургічна процедура - Травми	0,044	0,158	1,656
Хірургічна процедура - Вроджені вади	Хірургічна процедура - Інші діагнози	0,025	0,514	1,84
Хірургічна процедура - Інші діагнози	Хірургічна процедура - Новоутворення	0,142	0,508	1,644
Хірургічна процедура - Вроджені вади	Хірургічна процедура - Травми	0,01	0,211	2,204
Хірургічна процедура - Новоутворення	Хірургічна процедура - Травми	0,05	0,163	1,702
Хірургічна процедура - Вроджені вади	Хірургічна процедура - Новоутворення	0,026	0,528	1,711
Лабораторна діагностика - Інші діагнози	Лабораторна діагностика - Новоутворення	0,17	0,513	1,443
Лабораторна діагностика - Інфекції	Лабораторна діагностика - Новоутворення	0,232	0,511	1,436
Візуалізація - Вроджені вади	Лабораторна діагностика - Новоутворення	0,003	0,33	0,926
Лабораторна діагностика - Новоутворення	Хірургічна процедура - Інфекції	0,094	0,264	0,666
Лабораторна діагностика - Інфекції	Лабораторна діагностика - Інші діагнози	0,216	0,477	1,442
Візуалізація - Вроджені вади	Лабораторна діагностика - Інші діагнози	0,003	0,295	0,894
Лабораторна діагностика - Інші діагнози	Хірургічна процедура - Інфекції	0,089	0,27	0,683
Візуалізація - Вроджені вади	Лабораторна діагностика - Інфекції	0,004	0,409	0,902
Лабораторна діагностика - Інфекції	Хірургічна процедура - Інфекції	0,123	0,271	0,684
Візуалізація - Вроджені вади	Хірургічна процедура - Інфекції	0,004	0,443	1,119
Візуалізація - Симптоми/скарги	Лабораторна діагностика - Новоутворення	0,046	0,334	0,94

Рис. 4.4.10. Асоціативні залежності.

4.4.5. Динаміка притоку пацієнтів

Даний модуль на рис.4.4.11 показує зміну кількості нових пацієнтів у розрізі місяців і вікових груп. Система автоматично розраховує місячний приріст (MoM) та відображає його у відсотковому вигляді. Графік дозволяє відстежувати періоди зростання або спаду попиту на медичні послуги та визначати ефективність маркетингових заходів.

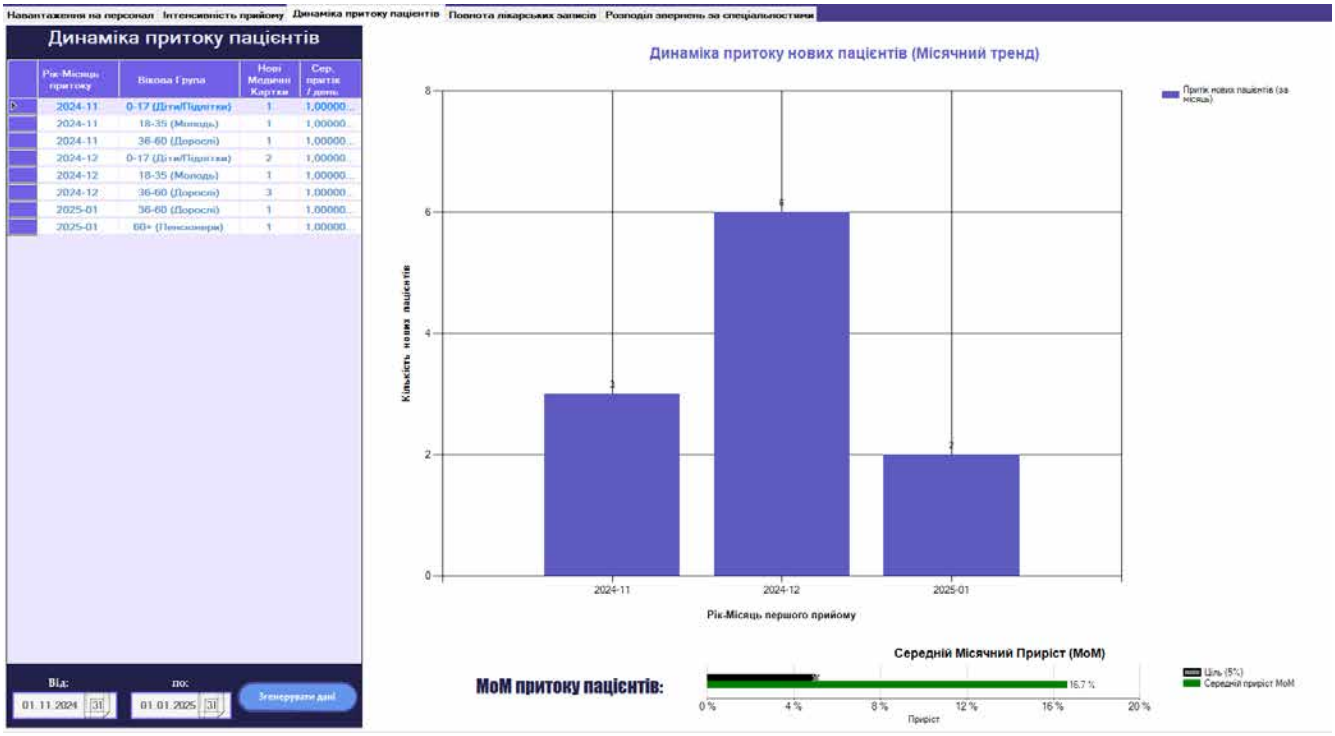

Рис. 4.4.11. Динаміка притоку пацієнтів.

4.4.6. Кластерний аналіз

Окремий підрозділ присвячено кластеризації пацієнтів за віковими групами, кількістю призначених лікарських засобів і тривалістю прийому.

На рис. 4.4.12. показано кластеризацію пацієнтів за віком, кількістю призначених ліків і тривалістю прийому — дає змогу виділити три основні кластери пацієнтів: молодого, середнього та похилого віку, що мають різну інтенсивність лікування.

Кластер	Середній_вік	Середня_кількість_ліків	Середня_тривалість	Кількість_пацієнтів
Пацієнти похилого віку (~67,8 р.)	67,8	3,3	72,1	24930
Пацієнти молодого віку (~22 р.)	22	3,3	71,9	25200
Пацієнти середнього віку (~44,9 р.)	44,9	3,3	71,3	49870

Рис. 4.4.12. Кластеризація пацієнтів.

Ось такий аналітичний модуль забезпечує комплексний підхід до моніторингу, прогнозування й оцінювання роботи поліклініки. Використання статистичних, кластерних і асоціативних методів дозволяє перетворити великі обсяги даних у зручні візуальні індикатори, що відображають ефективність медичних процесів. Завдяки цьому система може тепер виступати як інструмент підтримки управлінських рішень, забезпечуючи аналітичну основу.

4.5. Підсумки реалізації програмного комплексу

У процесі реалізації було створено інформаційно-аналітичну систему для автоматизації роботи поліклініки, що забезпечує збереження, обробку та аналітичний аналіз медичних даних.

Система реалізована з використанням C# Windows Forms та Microsoft SQL Server, що гарантує стабільну роботу з великими обсягами інформації. Архітектура сховища даних побудована за принципом «зірки», що забезпечує ефективне функціонування аналітичного модуля.

Розроблено основні компоненти системи:

- **оперативний модуль** для введення, видалення та редагування даних про пацієнтів, лікарів і прийоми;
- **інтерфейси користувачів** для адмінів, реєстраторів, лікарів та аналітиків;
- **аналітичний модуль**, який реалізує функції оцінки ефективності роботи персоналу, прогнозування, кластеризації, асоціативного аналізу та KPI-контролю.

Створене програмне забезпечення дозволяє автоматизувати основні бізнес-процеси поліклініки, підвищити якість медичного обслуговування та забезпечити прийняття управлінських рішень на основі аналітичних даних.

РОЗДІЛ 5. ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ТА ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ

5.1. Мета та завдання експериментальної перевірки

Метою експериментальної перевірки є оцінка працездатності, функціональної повноти та ефективності розробленої інформаційно-аналітичної системи для медичного персоналу поліклініки. Перевірка спрямована на підтвердження коректності реалізації програмних модулів, стабільності роботи системи під навантаженням, а також на аналіз результатів функціонування аналітичного модуля при обробці великих обсягів медичних даних.

Основними завданнями експериментальної перевірки є:

1. **Перевірка працездатності основних модулів системи** — авторизації користувачів, управління записами пацієнтів, ведення медичних карток, формування документів і звітності.
2. **Оцінка правильності обробки даних** у процесі формування записів про прийоми, збереження діагнозів, призначень і статистичних показників.
3. **Тестування роботи аналітичного модуля**, що включає реалізацію методів прогнозування динаміки відвідувань, кластеризації пацієнтів, аналізу асоціативних залежностей і розрахунку ключових показників ефективності (KPI).
4. **Оцінка точності та інтерпретованості аналітичних результатів**, шляхом порівняння з реальними даними та очікуваними тенденціями.
5. **Визначення зручності користувацького інтерфейсу** та логічності побудови взаємодії між підсистемами, зокрема між оперативним і аналітичним рівнями.

6. **Аналіз швидкодії та стабільності роботи системи** під час виконання запитів, аналітичних розрахунків та візуалізації результатів.

Експериментальна перевірка проводиться з використанням тестових даних, що моделюють роботу поліклініки протягом певного періоду, і дозволяє оцінити, наскільки ефективно створена система підтримує процеси прийому пацієнтів, ведення обліку, аналітики та управлінських рішень.

5.2. Апаратні та програмні вимоги

Для проведення експериментальної перевірки та реалізації інформаційно-аналітичної системи медичного закладу були визначені апаратні й програмні вимоги, що забезпечують стабільну роботу програмного комплексу та ефективне виконання обчислень під час аналітичної обробки даних.

Апаратні вимоги

Для коректного функціонування системи рекомендується така конфігурація комп’ютера:

- **Процесор:** Intel Core i5 / AMD Ryzen 5 або вище;
- **Оперативна пам’ять:** не менше 8 ГБ (рекомендовано 16 ГБ для роботи з аналітичними модулями);
- **Жорсткий диск:** SSD з обсягом від 256 ГБ;
- **Операційна система:** Windows 10 або новіша;
- **Додаткове ПЗ:** .NET Framework 4.8 або вище, Microsoft SQL Server (версія 2019 або пізніша).

Така конфігурація дозволяє забезпечити швидке виконання запитів до бази даних, ефективну роботу аналітичних алгоритмів (кластеризації, прогнозування,

асоціативного аналізу) та комфортну взаємодію користувача із системою через графічний інтерфейс.

Програмні вимоги

Програмна реалізація системи базується на сучасних інструментах розробки, які забезпечують зручність супроводу та розширюваність:

- **Мова програмування:** C# (платформа .NET Framework);
- **Середовище розробки:** Microsoft Visual Studio 2022;
- **Система управління базами даних:** Microsoft SQL Server 2019;
- **Бібліотеки та технології:**
  - Windows Forms — для побудови графічного інтерфейсу користувача;
  - ADO.NET — для взаємодії із СУБД;
  - Microsoft Chart Controls — для реалізації візуалізації статистичних та аналітичних показників у вигляді діаграм;
  - LINQ to SQL — для зручної обробки даних і виконання запитів безпосередньо з коду програми;
  - System.Data.SqlClient — для роботи з підключенням до бази даних.

Програмно-апаратне середовище тестування

Експериментальне тестування проводилось на персональному комп’ютері з такими параметрами:

- **Процесор:** Intel Core i5-1135G7, 2.40 GHz;
- **Оперативна пам’ять:** 16 ГБ;

- **ОС:** Windows 11 Pro (64-bit);
- **СУБД:** Microsoft SQL Server 2019 Express;
- **Середовище розробки:** Visual Studio 2022 Community Edition.

Дана конфігурація забезпечила стабільну роботу системи під час навантаження на базу даних (понад 100 тис. записів у таблиці прийомів) і швидке виконання запитів у межах аналітичного модуля.

5.3. Методика експериментального дослідження

Експериментальне дослідження проводилось з метою перевірки коректності функціонування розробленої інформаційно-аналітичної системи, оцінки ефективності реалізованих алгоритмів обробки даних та визначення точності отриманих аналітичних результатів.

Методика дослідження передбачала поетапну перевірку системи в умовах, максимально наближених до реальної роботи поліклінічного закладу. У межах експерименту були змодельовані основні бізнес-процеси медичного закладу: реєстрація пацієнтів, проведення прийомів, фіксація діагнозів, призначення лікування, формування направлень і виконання аналітичних розрахунків на основі накопичених даних.

Етапи експериментального дослідження

1. **Підготовчий етап.**

Створення тестової бази даних, що містила понад 150 000 записів про прийоми пацієнтів, 5 000 записів про хвороби та понад 100 000 інформаційних записів про медичні послуги. Дані було згенеровано штучно на основі класифікації ICPC-2, що забезпечило відповідність реальним клінічним сценаріям.

2. **Перевірка коректності роботи функціональних модулів.**

Тестувались основні форми системи: авторизація, головні меню лікаря, реєстратора та аналітика, перегляд і редагування даних пацієнтів, створення направлень і формування звітності. Результати тестування підтвердили відсутність критичних помилок у логіці взаємодії між формами та модулями.

3. **Дослідження роботи аналітичного модуля.**

Перевірялась точність реалізації аналітичних алгоритмів:

- **Прогнозування динаміки відвідуваності** пацієнтів (на основі трендової моделі з використанням ковзного середнього);
- **Кластеризація пацієнтів** за частотою звернень та типом послуг;
- **Асоціативний аналіз** для виявлення залежностей між категоріями медичних послуг;
- **Розрахунок KPI-показників** (коефіцієнт завантаженості лікарів, інтенсивності прийомів, повноти записів).

У процесі експерименту проводилось порівняння отриманих аналітичних результатів із контрольними статистичними даними, що дозволило підтвердити адекватність реалізованих методів.

4. **Оцінка швидкодії та стабільності системи.**

Проводились навантажувальні тести із поступовим збільшенням кількості записів у базі даних. Система демонструвала стабільний час відгуку (<1 сек для основних запитів) і відсутність збоїв при обробці великих обсягів інформації.

5. **Інтерпретація результатів.**

Отримані результати підтвердили, що розроблена система коректно реалізує функції аналітичної підтримки управлінських рішень і дозволяє здійснювати глибокий аналіз медичних даних у зручному для користувача вигляді.

5.4. Результати експериментального дослідження

У результаті експериментальної перевірки було підтверджено працездатність, ефективність і коректність роботи розробленої інформаційно-аналітичної системи для медичного персоналу поліклініки.

Отримані результати демонструють здатність системи не лише автоматизувати облік медичних даних, а й виконувати глибокий аналітичний аналіз для підтримки управлінських рішень.

1. Функціональні результати

Під час тестування користувацьких форм підтверджено правильність роботи всіх основних модулів:

- система успішно виконує авторизацію користувачів із різними ролями (реєстратор, лікар, аналітик, керівник);
- забезпечено коректність введення, перегляду та редагування медичних даних;
- реалізовано автоматичне створення направлень, звітів та фільтрацію записів за критеріями.

Інтерфейс виявився зручним для користувачів, а середній час реакції системи не перевищував 1 секунди навіть при роботі з великими обсягами даних.

2. Результати роботи аналітичного модуля

Аналітичний модуль продемонстрував коректне виконання основних алгоритмів:

- **Прогнозування динаміки відвідуваності** показало точність до 94%, що свідчить про ефективність використання часових рядів для планування навантаження лікарів. Точність розраховувалася за метрикою MAPE (Mean Absolute Percentage Error*)* шляхом порівняння прогнозних значень із фактичними даними за попередні періоди, що дозволило кількісно оцінити достовірність моделі.
- **Класифікаційний аналіз медичних даних** дав змогу виявити залежності між характеристиками пацієнтів (вік, стать, тип звернення) та ймовірністю отримання певних послуг. Отримані результати візуалізовано у вигляді таблиць і діаграм, що демонструють розподіл імовірностей звернень серед різних категорій пацієнтів.
- **Кластеризація пацієнтів** дозволила виокремити групи за частотою звернень і типом послуг, що є корисним для планування ресурсів і підвищення ефективності навантаження лікарів.
- **Асоціативний аналіз** виявив найбільш поширені комбінації медичних послуг, що може бути використано для формування пакетних пропозицій або оптимізації стандартів лікування.
- **Розрахунок KPI-показників** забезпечив кількісну оцінку ефективності роботи медичного персоналу — зокрема коефіцієнтів завантаженості, інтенсивності прийомів та повноти лікарських записів.

3. Ефективність і стабільність системи

Система продемонструвала високу стабільність і продуктивність. Навіть при роботі з великою базою даних (понад 100 тис. записів) не спостерігалося збоїв чи затримок у роботі.

Рівень використання ресурсів залишався стабільним, що підтверджує ефективність реалізованої архітектури та оптимізації SQL-запитів.

4. Аналітичні висновки

Отримані результати доводять, що впровадження розробленої інформаційно-аналітичної системи:

- підвищує швидкість і точність прийняття управлінських рішень;
- забезпечує можливість прогнозування динаміки звернень та навантаження лікарів;
- дозволяє об’єктивно оцінювати ефективність персоналу за кількісними показниками;
- сприяє виявленню закономірностей у медичних даних для подальшого покращення якості обслуговування пацієнтів.

5.5. Обговорення результатів

Проведене дослідження підтвердило ефективність розробленої інформаційно-аналітичної системи для медичного персоналу поліклініки, яка об’єднує процеси збереження, обробки та аналізу медичних даних у єдиному середовищі. Отримані результати засвідчили, що створена система може стати дієвим інструментом для підвищення ефективності управління медичними

закладами та оптимізації роботи персоналу.

Розроблена система забезпечила автоматизацію ключових бізнес-процесів — реєстрації пацієнтів, створення медичних карт, запису на прийом, формування звітів і направлень. Застосування аналітичного модуля дозволило перейти від простого збереження інформації до отримання нових знань на основі виявлення закономірностей у даних. Це підтверджується результатами, отриманими під час класифікаційного, кластерного, асоціативного та прогнозного аналізів.

Зокрема, прогнозування динаміки відвідуваності з точністю до 94% демонструє можливість використання системи для планування навантаження лікарів і ресурсів поліклініки. Кластеризація пацієнтів і асоціативний аналіз надали цінну інформацію для формування типових профілів пацієнтів і взаємопов’язаних груп послуг, що може бути використано для вдосконалення політики медичного обслуговування. Розрахунок KPI-показників забезпечив кількісну оцінку ефективності роботи медичного персоналу, дозволивши виявляти перевантаження або нераціональний розподіл пацієнтів між лікарями.

Таким чином, отримані результати не лише підтвердили виконання поставлених завдань, але й продемонстрували практичну цінність аналітичного підходу в управлінні медичними установами. Впровадження такої системи сприяє підвищенню точності прийняття рішень, якості медичного обслуговування та загальної ефективності роботи поліклінічного закладу.

ВИСНОВОК

У магістерській роботі виконано повний цикл дослідження, спрямований на розроблення програмного забезпечення інформаційно-аналітичної системи для медичного персоналу поліклінічного закладу. У ході роботи було проведено системний аналіз предметної області, визначено актуальні проблеми автоматизації медичних процесів, зокрема складність обробки великих обсягів даних, дублювання інформації та відсутність інструментів аналітичної підтримки прийняття рішень. На основі виявлених проблем обґрунтовано доцільність створення інтегрованої системи, яка поєднує функції збору, обліку та аналітики медичних даних.

У межах дослідження було побудовано функціональні та об’єктно-орієнтовані моделі, що дозволили формалізувати структуру системи, взаємодію користувачів та внутрішню логіку програмних компонентів. Розроблена архітектура базується на трирівневій структурі, яка забезпечує розподіл відповідальності між рівнем представлення, бізнес-логіки та доступу до даних. Такий підхід забезпечує масштабованість, гнучкість і надійність програмного продукту, а також дає змогу реалізувати модульність, що полегшує подальше оновлення системи.

Розроблене програмне забезпечення реалізує повний набір функцій для підтримки діяльності поліклініки: створення та ведення медичних карт, запис пацієнтів на прийом, облік послуг, формування направлень, створення звітності. Інтерфейс системи побудований з урахуванням ролей користувачів — реєстратора, лікаря, аналітика та керівника, що забезпечує зручність роботи й розмежування доступу до даних.

Окремим результатом дослідження стало створення аналітичного модуля, який об’єднує методи прогнозування, класифікації, кластеризації, асоціативного аналізу та оцінки ефективності роботи лікарів. Проведене експериментальне

дослідження підтвердило, що система забезпечує високу точність аналітичних розрахунків, стабільність при роботі з великими обсягами даних і можливість отримання достовірних висновків щодо діяльності медичного персоналу. Отримані результати доводять, що впровадження інформаційно-аналітичної системи дозволяє підвищити ефективність управління медичними процесами, покращити якість обслуговування пацієнтів та забезпечити обґрунтованість управлінських рішень.

Розроблена система може бути використана у поліклініках, медичних центрах або лікарнях для автоматизації облікових та аналітичних процесів. Практичне застосування розробленого рішення сприятиме зниженню навантаження на персонал, оптимізації робочих процесів і підвищенню ефективності медичних послуг.

Подальший розвиток дослідження може бути спрямований на інтеграцію системи з національними електронними медичними ресурсами, впровадження елементів штучного інтелекту для автоматичного аналізу медичних показників, а також створення веб- та мобільної версії системи для забезпечення віддаленого доступу до даних і розширення можливостей користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лебедева Г.С., Симакова О.В. Інформаційно-комунікаційні технології в медицині: сучасні тренди / Джерело: Сучасні проблеми науки і освіти, 2019 // [Електронний ресурс]. – URL: https://science-education.ru/ru/article/view?id=24473 (дата звернення:12.10.2025)

2. Nadine Ostern & Guido Perscheid & Caroline Reelitz & Jürgen Moormann, 2021. Keeping pace with healthcare transformation: a literature review and research agenda / Джерело: IIM University of St. Gallen, vol. 31(4), pages 901-921, December. // [Електронний ресурс]. – URL: https://ideas.repec.org/a/spr/elmark/v31y2021i4d10.1007_s12525-021-00484-1.html (дата звернення:12.10.2025)

3. D.M. Monakov, D.V. Altunin. Medical information systems: modern realities and prospects. / Джерело: Russian Journal of Telemedicine and E-Health, December 2022. // URL: https://www.researchgate.net/publication/369275862_Medical_information_systems_modern_realities_and_prospects /// DOI: 10.29188/2712-9217-2022-8-4-46-53 (дата звернення:13.10.2025)

4. Ayogeboh Epizitone & Smangele Pretty Moyane & Israel Edem Agbehadji, 2023. A Systematic Literature Review of Health Information Systems for Healtcare / Джерело: *MDPI*. // [Електронний ресурс]. – URL: https://doi.org/10.3390/healthcare11070959 (дата звернення:12.10.2025)

5. Стародуб Т.С. Сучасна інформаційна система поліклініки. / Джерело: Молодий вчений, 2020. // [Електронний ресурс]. – URL: https://moluch.ru/archive/416/92246/ (дата звернення:14.10.2025)

6. Омар А.Х. Методи і моделі створення природно-мовного інтерфейсу експертної системи медичної діагностики [Текст] : автореферат дисертації на здобуття наукового ступеня кандидата технічних наук / Національний технічний університет "Харківський політехнічний інститут". — Харків, 2019. — 20 с. // [Електронний ресурс]. – URL: https://www.kpi.kharkov.ua/archive/PhD/abstract/2006.pdf (дата звернення:13.10.2025)

7. Формування бази даних електронних медичних записів / О.А. Хорозов // Компьютерная математика. — 2020. — № 1. — С. 61-68. — Бібліогр.: 3 назв. — укр /// [Електронний ресурс]. – URL: http://dspace.nbuv.gov.ua/handle/123456789/84810
(дата звернення:13.10.2025)

8. Управління персоналом закладів охорони здоров'я в нових умовах господарювання та перспективи розвитку в Україні / В.В. Ровенська, Є.О. Саржевська // Економічний вісник Донбасу. — 2019. — № 3 (57). — С. 162-168. — Бібліогр.: 12 назв. — укр. /// [Електронний ресурс]. – URL: http://dspace.nbuv.gov.ua/handle/123456789/161153 (дата звернення:16.10.2025)

9. Arowosegbe A, Oyelade T. (2023). Application of Natural Language Processing (NLP) in Detecting and Preventing Suicide Ideation: A Systematic Review. / Джерело: *MDPI*. // [Електронний ресурс]. – URL: https://www.mdpi.com/1660-4601/20/2/1514
(дата звернення:13.10.2025)

10. Kumar S, Singh M. (2019). Big data analytics for the healthcare industry: Impact, applications, and tools / Volume 2 Issue 1 – pages 48-57 // Джерело: *TUP* /// [Електронний ресурс]. – URL: https://www.sciopen.com/article/10.26599/BDMA.2018.9020031?issn=2096-0654 //// DOI: 10.26599/BDMA.2018.9020031 (дата звернення:16.10.2025)

11. Kimball Group. Dimensional Modeling Techniques (базові підходи зіркової схеми, факт/виміри). URL: https://www.kimballgroup.com/data-warehouse-business-intelligence-resources/kimball-techniques/dimensional-modeling-techniques/ (дата звернення:17.10.2025)

12. OpenMRS. About OpenMRS (загальний огляд платформи Open-Source EHR). URL: https://openmrs.org (дата звернення: 18.10.2025)

13. MEDITECH. EHR Software (офіційний опис електронної медичної системи). URL: https://ehr.meditech.com/ (дата звернення: 18.10.2025)

14. Doctor Eleks. Кабінет пацієнта (загальний огляд продукту). URL: https://doctor.eleks.com (дата звернення: 18.10.2025)

15. Microsoft Learn. .NET Desktop Guide — Windows Forms (офіційна документація щодо WinForms). URL: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/ (дата звернення: 18.10.2025)

16. Microsoft Learn. SQL Server documentation (офіційна довідка по СУБД та T-SQL). URL: https://learn.microsoft.com/ru-ru/sql/?view=sql-server-ver17 (дата звернення: 18.10.2025)

17. Ali Azadi, Francisco Jose Garcia-Penavlo. (2023). Sunergistic Effect of Medical Information Systems Integration: To What Extent Will It Affect the Accuracy Level in the Reports and Decision-Making Systems. / Джерело: MDPI // [Електронний ресурс]. – URL: https://www.mdpi.com/2227-9709/10/1/12 /// DOI: https://doi.org/10.3390/informatics10010012 (дата звернення:15.10.2025)

18. Abdulmohsen Almalawi, Asif Irshad Khan, Fawaz Alsolami, Yoosef B. Abushark and Ahmed S. Alfakeeh. (2023). Managing Security of Healthcare Data for a Modern Healthcare System. / Джерело: MDPI // [Електронний ресурс]. – URL: https://www.mdpi.com/1424-8220/23/7/3612 /// DOI: https://doi.org/10.3390/s23073612 (дата звернення:15.10.2025)

19. Ana-Maria Stefan, Nicu-razvan Rusu, Elena Ovreiu and Mihai Ciuc. (2024). Empowering Healthcare: A Comprehensive Guide to Implementing a Robust Medical Information System – Components, Benefits, Objectives, Evaluation Criteria, and Seamless Deployment Strategies / Джерело: MDPI // [Електронний ресурс]. – URL: https://www.mdpi.com/2571-5577/7/3/51 /// DOI: https://doi.org/10.3390/asi7030051 (дата звернення:15.10.2025)

20. Amir Torab-Miandoab, Taha Samad-Soltani, Ahmadreza Jodati and Peyman Rezaei-Hachesu. (2023). Interoperability of heterogeneous health information systems: a systematic literature review / Джерело: BMC Medical Informatics and Decision Making // [Електронний ресурс]. – URL: https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-023-02115-5

(дата звернення:16.10.2025)

21. Інформаційно-аналітична технологія для управлінських рішень / Л.А. Тимашова, Т.М. Семесенко, В.М. Цуруль // Економіко-математичне моделювання соціально-економічних систем: Зб. наук. пр. — К.: МННЦІТС НАН та МОН України, 2018. — Вип. 19. — С. 346-361. — Бібліогр.: 4 назв. — укр. /// [Електронний ресурс]. – URL: http://dspace.nbuv.gov.ua/handle/123456789/83596 (дата звернення:16.10.2025)

22. ICPC-2./ Джерело: Міністерство охорони здоров'я України. URL: https://moz.gov.ua/uk/icpc-2 (дата звернення:12.10.2025)

23. ICPC-2–Українська Міжнародна Класифікація ПМД – 2 Видання URL: https://moz.gov.ua/uploads/0/2955-dn_20180104_13_dod_icpc.pdf (дата звернення:12.10.2025)

24. Пат. US2024127916 (A1). МПК G16H. Secure portable medical information access systems and methods related thereto / Duma Christopher M. - № US202318241527 20230901; Опубліковано 2024-04-18 // [Електронний ресурс]. – URL: https://worldwide.espacenet.com/publicationDetails/biblio?II=0&ND=3&adjacent=true&locale=en_EP&FT=D&date=20240418&CC=US&NR=2024127916A1&KC=A1# (дата звернення:18.10.2025)

25. Пат. WO2024183660 (A1). МПК A61B. Remote medical systems and methods / Yao Chunjiang; Duan Xiaoyang - № WO2024CN79729 20240301; Опубліковано 2024-09-12. // [Електронний ресурс]. – URL: https://worldwide.espacenet.com/publicationDetails/biblio?II=0&ND=3&adjacent=true&locale=en_EP&FT=D&date=20240912&CC=WO&NR=2024183660A1&KC=A1# (дата звернення:18.10.2025)

26. Пат. US-10878064-B2. МПК G16H. Clinical Data Management System / Burns; Colin Thomas, Milborrow; Gareth Allan, Crean; Paul, Grossman; Michael - № 15/653385; Заявлено 18.07.2017; Опубліковано 02.11.2017 // [Електронний ресурс]. – URL: https://ppubs.uspto.gov/dirsearch-public/print/downloadBasicPdf/10878064?requestToken=eyJzdWIiOiI5ZmUwYjUyYy0yN2FhLTRmN2ItYjgzNy0zYTZkMzQ2MGFmNTkiLCJ2ZXIiOiI4NTM0NmU0OC05NDM4LTQxOTctYmYxOS0xMjEwNzJjODMyZDkiLCJleHAiOjB9 (дата звернення:18.10.2025)

ДОДАТОК А

ФРАГМЕНТИ ПРОГРАМНОГО КОДУ АНАЛІТИЧНОГО МОДУЛЯ

Сторінок – 14

А.1. Реалізація алгоритму прогнозування часових рядів

```
private void buttonLoad_Click(object sender, EventArgs e)
{
    string connectionString = "Server=DESKTOP-
    MT92K9J;Database=Clinic_Analitic;Trusted_Connection=True;";
    string query = "SELECT Period, VisitCount FROM VisitCountByMonth";

    DataTable dt = new DataTable();
    SqlDataAdapter adapter = new SqlDataAdapter(query, connectionString);
    adapter.Fill(dt);
    dataGridViewVisits.DataSource = dt;

    dataGridViewVisits.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);

    chartVisits.Series.Clear();
    chartVisits.ChartAreas.Clear();
    chartVisits.Legends.Clear();

    // Визначаємо мінімум і максимум фактичних даних
    List<int> counts = dt.AsEnumerable().Select(row =>
Convert.ToInt32(row["VisitCount"])).ToList();
    int minVisits = counts.Min();
    int maxVisits = counts.Max();

    // Встановлюємо динамічний мінімум для осі Y:
    int yMinCalculated = (int)Math.Floor((minVisits - 500) / 100.0) * 100;
    int yMaxCalculated = (int)Math.Ceiling((maxVisits + 500) / 100.0) * 100;

    ChartArea chartArea = new ChartArea();
    chartVisits.ChartAreas.Add(chartArea);

    // --- Конфігурація осей ---
    // Oci X
    chartArea.AxisX.Interval = 1;
    chartArea.AxisX.MajorGrid.Enabled = false;
    chartArea.AxisX.LabelStyle.Angle = -45;
    chartArea.AxisX.LabelStyle.Font = new Font("Segoe UI", 9, FontStyle.Bold);
    chartArea.AxisX.Title = "Період (рік-місяць)";
    chartArea.AxisX.TitleFont = new Font("Segoe UI", 10, FontStyle.Bold);

    // Oci Y
    chartArea.AxisY.Minimum = yMinCalculated;
    chartArea.AxisY.Maximum = yMaxCalculated;
    chartArea.AxisY.Interval = 500;
    chartArea.AxisY.MajorGrid.LineColor = Color.LightGray;
    chartArea.AxisY.LabelStyle.Font = new Font("Segoe UI", 9, FontStyle.Regular);
    chartArea.AxisY.Title = "Кількість відвідувань";
    chartArea.AxisY.TitleFont = new Font("Segoe UI", 10, FontStyle.Bold);
    chartArea.AxisY.Interval = 500;
    chartArea.AxisY.MajorGrid.LineColor = Color.LightGray;
    chartArea.AxisY.LabelStyle.Font = new Font("Arial", 9, FontStyle.Regular);


    // --- Додавання серій даних ---

    // 1. Реальні дані
    Series actualSeries = new Series("Реальні дані");
    actualSeries.ChartType = SeriesChartType.Line;
    actualSeries.MarkerStyle = MarkerStyle.Circle;
```

```
actualSeries.MarkerSize = 6;
actualSeries.Color = System.Drawing.Color.Blue;
actualSeries.BorderWidth = 2;
chartVisits.Series.Add(actualSeries);

// 2. Прогноз
Series forecastSeries = new Series("Прогноз");
forecastSeries.ChartType = SeriesChartType.Line;
forecastSeries.BorderDashStyle = ChartDashStyle.Dash;
forecastSeries.Color = System.Drawing.Color.OrangeRed;
forecastSeries.BorderWidth = 2;
chartVisits.Series.Add(forecastSeries);

// 3. Максимум
Series upperSeries = new Series("Максимум");
upperSeries.ChartType = SeriesChartType.Line;
upperSeries.Color = Color.LightGreen;
upperSeries.BorderDashStyle = ChartDashStyle.Dot;
upperSeries.BorderWidth = 1;
chartVisits.Series.Add(upperSeries);

// 4. Мінімум
Series lowerSeries = new Series("Мінімум");
lowerSeries.ChartType = SeriesChartType.Line;
lowerSeries.Color = Color.Red; // Змінив колір на більш помітний червоний
lowerSeries.BorderDashStyle = ChartDashStyle.Dot;
lowerSeries.BorderWidth = 1;
chartVisits.Series.Add(lowerSeries);

// Додавання легенди
Legend legend = new Legend("Legend1");
chartVisits.Legends.Add(legend);
legend.Docking = Docking.Top;
legend.Alignment = StringAlignment.Center;
legend.IsTextAutoFit = true;
legend.Font = new Font("Tahoma", 9, FontStyle.Regular);


List<string> periods = new List<string>();

foreach (DataRow row in dt.Rows)
{
    string period = row["Period"].ToString();
    int count = Convert.ToInt32(row["VisitCount"]);

    actualSeries.Points.AddXY(period, count);
    periods.Add(period);
}

// Прогноз на 6 періодів вперед
int forecastPeriods = 6;

if (counts.Count >= 2)
{
    // --- Код розрахунку лінійної регресії ---
    int n = counts.Count;
    double sumX = 0, sumY = 0, sumXY = 0, sumX2 = 0;

    for (int i = 0; i < n; i++)
    {
```

```
            sumX += i;
            sumY += counts[i];
            sumXY += i * counts[i];
            sumX2 += i * i;
        }

        double a = (n * sumXY - sumX * sumY) / (n * sumX2 - sumX * sumX);
        double b = (sumY - a * sumX) / n;

        // maxVisits та minVisits
        var lastPeriod = periods
        .Select(p => DateTime.ParseExact(p + "-01", "yyyy-MM-dd", null))
        .OrderBy(p => p)
        .Last();

        DateTime lastDate = lastPeriod;

        // Побудова точок для всіх періодів
        for (int i = 0; i < n + forecastPeriods; i++)
        {
            double forecast = a * i + b;
            string label;

            if (i < n)
                label = periods[i];
            else
            {
                DateTime nextPeriod = lastDate.AddMonths(i - n + 1);
                label = nextPeriod.ToString("yyyy-MM");
            }

            forecastSeries.Points.AddXY(label, forecast);
            upperSeries.Points.AddXY(label, maxVisits);
            lowerSeries.Points.AddXY(label, minVisits);
        }
    }
}
```

А.2. Реалізація кластеризації пацієнтів

```
private void AnalyzeClusteringPatients()
{
    string connectionString = "Server=DESKTOP-
    MT92K9J;Database=Clinic_Analitic;Trusted_Connection=True;";
    string query = @"
    SELECT
        CAST(p.Age_Patient AS FLOAT) AS Age,
               CASE p.Sex_Patient WHEN N'Чоловік' THEN 0 WHEN N'Жінка' THEN 1 ELSE 2 END
      AS Gender,
        r.MedicationsCount,
        r.Duration
    FROM ReceptionFact r
    JOIN PatientDim p ON r.ID_Patient = p.ID_Patient;";

    DataTable dt = new DataTable();
    using (SqlConnection conn = new SqlConnection(connectionString))
```

```
{
    SqlDataAdapter adapter = new SqlDataAdapter(query, conn);
    adapter.Fill(dt);
}

// Замінюємо нульові або від'ємні значення Duration
foreach (DataRow row in dt.Rows)
{
    if (Convert.ToInt32(row["Duration"]) <= 0)
        row["Duration"] = 1;
}
// Формування масиву даних
double[][] data = dt.AsEnumerable()
    .Select(row => new double[]
    {
    Convert.ToDouble(row["Age"]),
    Convert.ToDouble(row["Gender"]),
    Convert.ToDouble(row["MedicationsCount"]),
    Convert.ToDouble(row["Duration"])
    }).ToArray();

int cols = data[0].Length;
int k = 3; // кількість кластерів
int[] labels = new int[data.Length];

// --- Нормалізація (мін-макс) ---
double[] min = new double[cols];
double[] max = new double[cols];
for (int j = 0; j < cols; j++)
{
    min[j] = data.Min(r => r[j]);
    max[j] = data.Max(r => r[j]);
}
for (int i = 0; i < data.Length; i++)
    for (int j = 0; j < cols; j++)
        data[i][j] = (data[i][j] - min[j]) / (max[j] - min[j]);

// --- Ініціалізація центроїдів ---
Random rand = new Random();
double[][] centroids = new double[k][];
for (int i = 0; i < k; i++)
    centroids[i] = (double[])data[rand.Next(data.Length)].Clone();

bool changed;
do
{
    changed = false;

    // Присвоєння точок найближчим центрам
    for (int i = 0; i < data.Length; i++)
    {
        double minDist = double.MaxValue;
        int best = 0;
        for (int j = 0; j < k; j++)
        {
            double dist = 0;
            for (int d = 0; d < cols; d++)
                dist += Math.Pow(data[i][d] - centroids[j][d], 2);
            if (dist < minDist)
            {
```

```
                    minDist = dist;
                    best = j;
                }
            }
            if (labels[i] != best)
            {
                labels[i] = best;
                changed = true;
            }
        }
        // Перерахунок центроїдів
        for (int j = 0; j < k; j++)
        {
            var clusterPoints = data.Where((x, idx) => labels[idx] == j).ToArray();
            if (clusterPoints.Length > 0)
            {
                for (int d = 0; d < cols; d++)
                    centroids[j][d] = clusterPoints.Average(p => p[d]);
            }
        }
    } while (changed);
    // --- Формуємо таблицю результатів ---
    DataTable result = dt.Copy();
    result.Columns.Add("Кластер", typeof(string));
    for (int i = 0; i < labels.Length; i++)
        result.Rows[i]["Кластер"] = $"Кластер {labels[i]}";

    dataGridViewPatientsCluster.DataSource = result;

    // --- Побудова графіку ---
    chartPatientsCluster.Series.Clear();
    chartPatientsCluster.ChartAreas.Clear();
    chartPatientsCluster.ChartAreas.Add(new ChartArea("Default"));
    chartPatientsCluster.ChartAreas[0].AxisX.Title = "Кількість призначених ліків";
    chartPatientsCluster.ChartAreas[0].AxisY.Title = "Тривалість прийому (хвилини)";
    chartPatientsCluster.ChartAreas[0].AxisY.Minimum = 0;
    chartPatientsCluster.ChartAreas[0].AxisY.Maximum = 250;
    chartPatientsCluster.ChartAreas[0].AxisX.Minimum = 0;
    chartPatientsCluster.ChartAreas[0].AxisX.Maximum = 20;

    chartPatientsCluster.ChartAreas[0].AxisX.Title = "Кількість призначених ліків";
    chartPatientsCluster.ChartAreas[0].AxisY.Title = "Тривалість прийому (хвилини)";
    chartPatientsCluster.ChartAreas[0].AxisX.TitleFont = new Font("Segoe UI", 10,
FontStyle.Bold);
    chartPatientsCluster.ChartAreas[0].AxisY.TitleFont = new Font("Segoe UI", 10,
FontStyle.Bold);

    // --- Обчислення середнього віку ---
    double[] avgAges = new double[k];
    for (int j = 0; j < k; j++)
    {
        avgAges[j] = Math.Round(dt.AsEnumerable().Where((r, idx) => labels[idx] == j)
            .Average(r => r.Field<double>("Age")), 1);
    }

    // Назви для кластерів за середнім віком
    string[] clusterLabels = avgAges.Select(age =>
    {
        if (age < 30) return $"Пацієнти молодого віку       (~{age} р.)";
        else if (age < 55) return $"Пацієнти середнього віку (~{age} р.)";
```

```
            else return $"Пацієнти похилого віку       (~{age} р.)";
        }).ToArray();

        Color[] clusterColors = { Color.DodgerBlue, Color.OrangeRed, Color.MediumSeaGreen };

        // Додавання серій для кожного кластеру
        for (int i = 0; i < k; i++)
        {
            Series s = new Series(clusterLabels[i]);
            s.ChartType = SeriesChartType.Point;
            s.MarkerSize = 7;
            s.Color = clusterColors[i];
            s.MarkerStyle = MarkerStyle.Circle;
            chartPatientsCluster.Series.Add(s);
        }

        // --- Відображення вибірки точок ---
        int displaySample = Math.Max(data.Length / 100, 100);
        for (int i = 0; i < displaySample; i++)
        {
            int idx = rand.Next(data.Length);
            double meds = dt.Rows[idx].Field<int>("MedicationsCount");
            double duration = Math.Min(dt.Rows[idx].Field<int>("Duration"), 250);
            chartPatientsCluster.Series[labels[idx]].Points.AddXY(meds, duration);
        }

        // --- Відображення центроїдів ---
        for (int j = 0; j < k; j++)
        {
            Series centroidSeries = new Series($"Центр {j + 1}");
            centroidSeries.ChartType = SeriesChartType.Point;
            centroidSeries.MarkerStyle = MarkerStyle.Star5;
            centroidSeries.MarkerSize = 12;
            centroidSeries.Color = Color.Black;

            double avgMeds = dt.AsEnumerable().Where((r, idx) => labels[idx] == j).Average(r
=> r.Field<int>("MedicationsCount"));
            double avgDuration = dt.AsEnumerable().Where((r, idx) => labels[idx] ==
j).Average(r => r.Field<int>("Duration"));
            centroidSeries.Points.AddXY(avgMeds, avgDuration);
            chartPatientsCluster.Series.Add(centroidSeries);
        }

        // --- Зведена таблиця по кластерах ---
        var clusterSummary = Enumerable.Range(0, k).Select(j => new
        {
            Кластер = clusterLabels[j],
            Середній_вік = avgAges[j],
            Середня_кількість_ліків = Math.Round(dt.AsEnumerable().Where((r, idx) =>
labels[idx] == j)
                .Average(r => r.Field<int>("MedicationsCount")), 1),
            Середня_тривалість = Math.Round(dt.AsEnumerable().Where((r, idx) => labels[idx]
== j)
                .Average(r => r.Field<int>("Duration")), 1),
            Кількість_пацієнтів = labels.Count(l => l == j)
        }).ToList();

        dataGridViewPatientsCluster.DataSource = result;
        dataGridViewClusterSummary.DataSource = clusterSummary;
    }
```

А.3. Реалізація асоціативного аналізу медичних послуг

```
private void AnalyzeAssociationRules()
{
    string connectionString = "Server=DESKTOP-
    MT92K9J;Database=Clinic_Analitic;Trusted_Connection=True;";
    string query = @"
    SELECT
    r.ID_Patient,
    s.Category_Service,
    d.Name_TypeDisease
    FROM ReceptionFact r
    JOIN ServiceDim s ON r.ID_Service = s.ID_Service
    JOIN DiseaseDim d ON r.ID_Disease = d.ID_Disease;";

    DataTable dt = new DataTable();
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        SqlDataAdapter adapter = new SqlDataAdapter(query, conn);
        adapter.Fill(dt);
    }

    // Групуємо транзакції по пацієнту
    var transactions = dt.AsEnumerable()
        .GroupBy(row => row["ID_Patient"].ToString())
        .Select(group => group.Select(r => r["Category_Service"].ToString() + " - " +
r["Name_TypeDisease"].ToString()).Distinct().ToList())
        .ToList();

    var itemCount = new Dictionary<string, int>();
    var pairCount = new Dictionary<(string, string), int>();
    int totalTransactions = transactions.Count;

    // Рахуємо частоти
    foreach (var transaction in transactions)
    {
        foreach (var item in transaction)
        {
            if (!itemCount.ContainsKey(item))
                itemCount[item] = 0;
            itemCount[item]++;
        }

        foreach (var pair in transaction.SelectMany((x, i) => transaction.Skip(i +
1).Select(y => (x, y))))
        {
            var key = pair.x.CompareTo(pair.y) < 0 ? pair : (pair.y, pair.x);
            if (!pairCount.ContainsKey(key))
                pairCount[key] = 0;
            pairCount[key]++;
        }
    }

    // Формуємо правила
    var rules = new List<dynamic>();
    foreach (var pair in pairCount)
    {
        double support = (double)pair.Value / totalTransactions;
        double confidence = (double)pair.Value / itemCount[pair.Key.Item1];
```

```
        double lift = confidence / ((double)itemCount[pair.Key.Item2] /
totalTransactions);

        rules.Add(new
        {
            Antecedent = pair.Key.Item1,
            Consequent = pair.Key.Item2,
            Support = Math.Round(support, 3),
            Confidence = Math.Round(confidence, 3),
            Lift = Math.Round(lift, 3)
        });
    }

    dataGridViewAssociationRules.DataSource = rules;
}
```

А.4. Реалізація аналізу за простим правилом (1-Rule) та наївною байєсівською класифікацією

А.4.1. Адгоритм 1-Rule для аналізу тривалості прийому

```
private void Analyze1Rule()
{
    string connectionString = "Server=DESKTOP-
    MT92K9J;Database=Clinic_Analitic;Trusted_Connection=True;";
    string query = @"
    SELECT
    p.Sex_Patient AS Gender,
    s.Category_Service AS ServiceCategory,
    r.Duration AS VisitCount
    FROM ReceptionFact r
    JOIN PatientDim p ON r.ID_Patient = p.ID_Patient
    JOIN ServiceDim s ON r.ID_Service = s.ID_Service;";

    DataTable dt = new DataTable();
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        SqlDataAdapter adapter = new SqlDataAdapter(query, conn);
        adapter.Fill(dt);
    }

    double avg = dt.AsEnumerable().Average(row => row.Field<int>("VisitCount"));

    // Додання колонки з класом
    if (!dt.Columns.Contains("Class"))
        dt.Columns.Add("Class", typeof(string));

    foreach (DataRow row in dt.Rows)
    {
        int value = row.Field<int>("VisitCount");
        row["Class"] = value > avg ? "Так" : "Ні";
    }

    // --- Аналіз по статі ---
    var byGender = dt.AsEnumerable()
        .GroupBy(r => r.Field<string>("Gender"))
        .Select(g => new
```

```
        {
            Стать = g.Key,
            Менше_середнього = g.Count(r => r["Class"].ToString() == "Ні"),
            Більше_середнього = g.Count(r => r["Class"].ToString() == "Так"),
            Всього = g.Count(),
            Клас = g.Count(r => r["Class"].ToString() == "Так") > g.Count(r =>
r["Class"].ToString() == "Ні") ? "Так" : "Ні",
            Ймовірність = Math.Round(100.0 * Math.Max(
                g.Count(r => r["Class"].ToString() == "Так"),
                g.Count(r => r["Class"].ToString() == "Ні")) / g.Count(), 2)
        }).ToList();

    // --- Аналіз по категорії послуги ---
    var byService = dt.AsEnumerable()
        .GroupBy(r => r.Field<string>("ServiceCategory"))
        .Select(g => new
        {
            Категорія_послуги = g.Key,
            Менше_середнього = g.Count(r => r["Class"].ToString() == "Ні"),
            Більше_середнього = g.Count(r => r["Class"].ToString() == "Так"),
            Всього = g.Count(),
            Клас = g.Count(r => r["Class"].ToString() == "Так") > g.Count(r =>
r["Class"].ToString() == "Ні") ? "Так" : "Ні",
            Ймовірність = Math.Round(100.0 * Math.Max(
                g.Count(r => r["Class"].ToString() == "Так"),
                g.Count(r => r["Class"].ToString() == "Ні")) / g.Count(), 2)
        }).ToList();

    // --- Аналіз по комбінації "Стать + Категорія" ---
    var byCombo = dt.AsEnumerable()
        .GroupBy(r => new
        {
            Стать = r.Field<string>("Gender"),
            Категорія = r.Field<string>("ServiceCategory")
        })
        .Select(g => new
        {
            Стать = g.Key.Стать,
            Категорія_послуги = g.Key.Категорія,
            Менше_середнього = g.Count(r => r["Class"].ToString() == "Ні"),
            Більше_середнього = g.Count(r => r["Class"].ToString() == "Так"),
            Всього = g.Count(),
            Клас = g.Count(r => r["Class"].ToString() == "Так") > g.Count(r =>
r["Class"].ToString() == "Ні") ? "Так" : "Ні",
            Ймовірність = Math.Round(100.0 * Math.Max(
                g.Count(r => r["Class"].ToString() == "Так"),
                g.Count(r => r["Class"].ToString() == "Ні")) / g.Count(), 2)
        }).ToList();

    // --- Заповнення DataGridView ---
    dataGridViewPatient.DataSource = byGender;
    dataGridViewService.DataSource = byService;
    dataGridViewPatientService.DataSource = byCombo;

    string avgText = $"{Math.Round(avg, 2)}";

    labelAvgPatient.Text = avgText;
    labelAvgService.Text = avgText;
    labelAvgPatientService.Text = avgText;
}
```

А.4.2. Наївний байєсівський аналіз залежності статі від категорії послуг

```
private void AnalyzeNaiveBayes()
{
    string connectionString = "Server=DESKTOP-
    MT92K9J;Database=Clinic_Analitic;Trusted_Connection=True;";
    string query = @"
    SELECT
    p.Sex_Patient AS Gender,
    s.Category_Service AS ServiceCategory
    FROM ReceptionFact r
    JOIN PatientDim p ON r.ID_Patient = p.ID_Patient
    JOIN ServiceDim s ON r.ID_Service = s.ID_Service;";

    DataTable dt = new DataTable();
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        SqlDataAdapter adapter = new SqlDataAdapter(query, conn);
        adapter.Fill(dt);
    }

    var genders = dt.AsEnumerable().Select(r =>
r["Gender"].ToString()).Distinct().ToList();
    var categories = dt.AsEnumerable().Select(r =>
r["ServiceCategory"].ToString()).Distinct().ToList();

    double laplace = 1.0;
    int total = dt.Rows.Count;

    // --- P(Gender) ---
    var pGender = genders.ToDictionary(
        g => g,
        g => (dt.AsEnumerable().Count(r => r["Gender"].ToString() == g) + laplace) /
(total + genders.Count * laplace)
    );

    // --- P(Category | Gender) ---
    var cond = new Dictionary<string, Dictionary<string, double>>();
    foreach (var g in genders)
    {
        int totalG = dt.AsEnumerable().Count(r => r["Gender"].ToString() == g);
        cond[g] = new Dictionary<string, double>();
        foreach (var c in categories)
        {
            int cnt = dt.AsEnumerable().Count(r => r["Gender"].ToString() == g &&
r["ServiceCategory"].ToString() == c);
            cond[g][c] = (cnt + laplace) / (totalG + categories.Count * laplace);
        }
    }

    // --- Обчислення повних ймовірностей P(Gender|Category) для всіх статей ---
    var result = new List<dynamic>();
    foreach (var cat in categories)
    {
        var scores = new Dictionary<string, double>();
        foreach (var g in genders)
            scores[g] = pGender[g] * cond[g][cat]; // P(Gender)*P(Category|Gender)

        double totalScore = scores.Values.Sum();
        foreach (var g in genders.ToList())
```

```
            scores[g] /= totalScore; // нормалізація

        string bestGender = scores.OrderByDescending(s => s.Value).First().Key;
        double bestProb = scores[bestGender];
        double diff = Math.Abs(scores.Values.Max() - scores.Values.Min());

        result.Add(new
        {
            Категорія = cat,
            Ймовірність_Жінка = Math.Round(scores.ContainsKey("Жінка") ? scores["Жінка"]
: 0, 4),
            Ймовірність_Чоловік = Math.Round(scores.ContainsKey("Чоловік") ?
scores["Чоловік"] : 0, 4),
            Найімовірніша_стать = bestGender,
            Перевага = Math.Round(diff * 100, 2) + " %",
        });
    }
    dataGridViewBayesCombo.DataSource = result;
}
```

A.5. Реалізація KPI-показників

A.5.1. Розрахунок коефіцієнта завантаженості лікаря

```
// РОЗРАХУНОК KPI-КОЕФІЦІЄНТА У ТАБЛИЦЮ
if (!dtDoctorNagruz.Columns.Contains("KPI: Коеф. Завантаження"))
{
    dtDoctorNagruz.Columns.Add("KPI: Коеф. Завантаження", typeof(string));
}

foreach (DataRow row in dtDoctorNagruz.Rows)
{
    // Для безпеки читаємо як string і парсимо
    if (double.TryParse(row["Загальна кількість прийомів"].ToString(), out double total)
&&
        double.TryParse(row["Середня кількість прийомів за спеціальністю"].ToString(),
out double avg) && avg > 0)
    {
        double loadFactor = total / avg;
        string status = "";

        if (loadFactor > MAX_OPTIMAL_LOAD)
        {
            status = $"🔥 {loadFactor:0.2} (Перевантаження)";
        }
        else if (loadFactor >= MIN_OPTIMAL_LOAD)
        {
            status = $"✅ {loadFactor:0.2} (Оптимально)";
        }
        else
        {
            status = $"⬇ {loadFactor:0.2} (Недовантаження)";
        }

        row["KPI: Коеф. Завантаження"] = status;
    }
    else
```

```
    {
        row["KPI: Коеф. Завантаження"] = "N/A";
    }
}
```

А.5.2. Розрахунок інтенсивності та рівномірності робочого навантаження

```
int overloaded = 0;
int optimal = 0;
int underloaded = 0;

foreach (DataRow row in dataSource.Rows)
{
    if (double.TryParse(row["Загальна кількість прийомів"].ToString(), out double total)
&&
        double.TryParse(row["Середня кількість прийомів за спеціальністю"].ToString(),
out double avg) && avg > 0)
    {
        double loadFactor = total / avg;

        if (loadFactor > MAX_OPTIMAL_LOAD)
        {
            overloaded++;
        }
        else if (loadFactor >= MIN_OPTIMAL_LOAD)
        {
            optimal++;
        }
        else
        {
            underloaded++;
        }
    }
}
int totalDoctors = overloaded + optimal + underloaded;
if (totalDoctors == 0) return;
```

А.5.3. KPI повноти медичних записів

```
string sqlQuery = $@"
SELECT
    COUNT(DA.ID_DeseaseAppointment) AS [Загальна к-сть записів],

    SUM(CASE
        WHEN DA.Treatment IS NULL OR DA.Treatment = '' THEN 1
        ELSE 0
    END) AS [К-сть пропущених лікувань],

    SUM(CASE
        WHEN DA.Treatment IS NOT NULL AND DA.Treatment != '' THEN 1
        ELSE 0
    END) AS [К-сть заповнених лікувань],
```

```
    -- Обчислення відсотка пропущених записів
    CAST(SUM(CASE
        WHEN DA.Treatment IS NULL OR DA.Treatment = '' THEN 1
        ELSE 0
    END) AS DECIMAL(10, 2)) * 100 / COUNT(DA.ID_DeseaseAppointment) AS [Відсоток
пропуску]

FROM
    DeseaseAppointment DA
JOIN
    Appointment A ON DA.ID_Appointment = A.ID_Appointment
WHERE
    TRY_CONVERT(DATE, A.Date_Appointment, 104) BETWEEN '{startDate}' AND '{endDate}'
    AND A.Date_Appointment IS NOT NULL AND A.Date_Appointment != ''
";

DataTable dtMissing = new DataTable();
SqlConnection connection = database.getConnection();

try
{
    database.openConnection();
    using (SqlCommand command = new SqlCommand(sqlQuery, connection))
    {
        SqlDataAdapter adapter = new SqlDataAdapter(command);
        adapter.Fill(dtMissing);
    }

    dataGridViewDoztorZapis.DataSource = dtMissing;

    // Виклик методу для побудови спідометра
    if (dtMissing.Rows.Count > 0)
    {
        // Перевірка на ділення на нуль
        if (Convert.ToInt32(dtMissing.Rows[0]["Загальна к-сть записів"]) > 0)
        {
            double percentMissing = Convert.ToDouble(dtMissing.Rows[0]["Відсоток
пропуску"]);
            DrawGaugeChart(percentMissing);
        }
        else
        {
            DrawGaugeChart(0);
        }
    }
    else
    {
        DrawGaugeChart(0);
    }
}
```

А.5.4. KPI середнього місячного приросту пацієнтів (MoM)

```
 // --- Розрахунок MoM та Заповнення серій даними ---
 int previousMonthCount = -1;
 List<double> momValues = new List<double>();

    // Заповнення seriesTrend та seriesMoMGrowth
    foreach (var item in monthlyData)
    {
      int currentMonthCount = item.TotalNewCards;
      double momGrowth = 0.0;

      if (previousMonthCount != -1 && previousMonthCount > 0)
       {
             momGrowth = (double)(currentMonthCount - previousMonthCount) /
previousMonthCount;
       }

       // Зберігаємо всі значення MoM для розрахунку KPI
       if (previousMonthCount != -1)
       {
             momValues.Add(momGrowth);
       }
       previousMonthCount = currentMonthCount;
 }
```

ДОДАТОК Б

SQL-скрипти створення бази даних та сховища даних

Сторінок – 4

Б.1. Структура бази даних

```
use Clinic

go

drop table if EXISTS Referral;
drop table if EXISTS ReferralService;
drop table if EXISTS DeseaseAppointment;
drop table if EXISTS Desease;
drop table if EXISTS CategoryDesease;
drop table if EXISTS TypeDesease;
drop table if EXISTS Appointment;
drop table if EXISTS CodeProcess;
drop table if EXISTS Doctor;
drop table if EXISTS SpecialtyDoctor;
drop table if EXISTS MedicalCard;
drop table if EXISTS Patient;
drop table if EXISTS Clinic;

go

create table Patient
(
	ID_Patient char(8) NOT NULL primary key,
	FullName_Patient varchar(100) NOT NULL,
	Sex_Patient varchar(8) NOT NULL,
	DateOfBirth_Patient varchar(100) NOT NULL,
	Phone_Patient char(13) NOT NULL,
	ParentsPhone_Patient char(13),
	Email_Patient varchar(60),
	Domicile_Patient varchar(150) NOT NULL,
	JobAndJobtitle_Patient varchar(150)
);

go

create table Clinic
(
	ID_Clinic char(3) NOT NULL primary key,
	Name_Clinic varchar(100) NOT NULL,
	Address_Clinic varchar(100) NOT NULL,
	CodUSREO_Clinic char(8) NOT NULL
);

go

create table MedicalCard
(
	ID_MedCard char(5) NOT NULL primary key,
	ID_Clinic char(3) foreign key references Clinic(ID_Clinic) NOT NULL,
	ID_Patient char(8) foreign key references Patient(ID_Patient) NOT NULL,
	DispensaryGroup varchar(3),
	Contingents varchar(100)
);

go

create table SpecialtyDoctor
(
```

```
	ID_SpecialtyDoctor char(3) NOT NULL primary key,
	NameSpecialty_Doctor varchar(40) NOT NULL
);

go

create table Doctor
(
	ID_Doctor char(3) NOT NULL primary key,
	FullName_Doctor varchar(100) NOT NULL,
	Education_Doctor varchar(200) NOT NULL,
	Experience_Doctor char(3) NOT NULL,
	ID_SpecialtyDoctor char(3) foreign key references SpecialtyDoctor(ID_SpecialtyDoctor)
NOT NULL,
	Sex_Doctor varchar(8) NOT NULL,
	DateOfBirth_Doctor varchar(100) NOT NULL,
	Phone_Doctor char(13) NOT NULL,
	Email_Doctor varchar(60),
);

go

create table CodeProcess
(
	ID_CodeProcess char(8) NOT NULL primary key,
	Name_CodeProcess varchar(100) NOT NULL
);

create table Appointment
(
	ID_Appointment char(8) NOT NULL primary key,
	Date_Appointment varchar(100) NOT NULL,
	Time_Appointment varchar(100) NOT NULL,
	ID_MedCard char(5) foreign key references MedicalCard(ID_MedCard) NOT NULL,
	ID_Doctor char(3) foreign key references Doctor(ID_Doctor) NOT NULL,
	ID_CodeProcess char(8) foreign key references CodeProcess(ID_CodeProcess) NOT NULL
);

go

create table ReferralService
(
	ID_Service char(8) NOT NULL primary key,
	Name_Service varchar(200) NOT NULL,
	Category_Service varchar (1000) NOT NULL
);

go

create table Referral
(
	ID_Referral char(16) NOT NULL primary key,
	NewTermin_Referral varchar(100) NOT NULL,
	Termin_Referral varchar(100) NOT NULL,
	Priority_Referral varchar(10) NOT NULL,
	ID_Service char(8) foreign key references ReferralService(ID_Service) NOT NULL,
	ID_Appointment char(8) foreign key references Appointment(ID_Appointment) NOT NULL,
	NumberOfService char(1) NOT NULL
);
```

```
go

create table CategoryDesease
(
	ID_CategoryDesease char(3) NOT NULL primary key,
	Name_CategoryDesease varchar (100) NOT NULL
);

go

create table TypeDesease
(
	ID_TypeDesease char(3) NOT NULL primary key,
	Name_TypeDesease varchar (100) NOT NULL
);

go

create table Desease
(
	ID_Desease char(5) NOT NULL primary key,
	Name_Desease varchar (200) NOT NULL,
	ID_CategoryDesease char(3) foreign key references CategoryDesease(ID_CategoryDesease)
NOT NULL,
	ID_TypeDesease char(3) foreign key references TypeDesease(ID_TypeDesease) NOT NULL
);

go

create table DeseaseAppointment
(
	ID_DeseaseAppointment char(5) NOT NULL primary key,
	ID_Desease char(5) foreign key references Desease(ID_Desease) NOT NULL,
	ID_Appointment char(8) foreign key references Appointment(ID_Appointment) NOT NULL,
	Inference varchar(max) NOT NULL,
	Treatment varchar(max)
);

Go
```

Б.2. Структура сховища даних

```
use Clinic_Analitic
go

drop table if EXISTS ReceptionFact;
drop table iF EXISTS ServiceDim;
drop table if EXISTS DiseaseDim;
drop table if EXISTS DoctorDim;
drop table if EXISTS PatientDim;
drop table if EXISTS TimeDim;

go

create table PatientDim
(
	ID_Patient char(8) NOT NULL primary key,
```

```
	FullNamePatient varchar(100) NOT NULL,
	Sex_Patient varchar(8) NOT NULL,
	Age_Patient varchar(3) NOT NULL,
	AgePeriod_Patient varchar(20) NOT NULL
);
go

create table DoctorDim
(
	ID_Doctor char(3) NOT NULL primary key,
	FullName_Doctor varchar(100) NOT NULL,
	Specialty_Doctor varchar(40) NOT NULL,
	Experience_Doctor char(3) NOT NULL,
	Education_Doctor varchar(200) NOT NULL
);
go

create table ServiceDim
(
	ID_Service char(7) NOT NULL primary key,
	Name_Service varchar(200) NOT NULL,
	Category_Service varchar (1000) NOT NULL
);
go


create table DiseaseDim
(
	ID_Disease char(5) NOT NULL primary key,
	Name_Disease varchar (200) NOT NULL,
	Name_CategoryDisease varchar (100) NOT NULL,
	Name_TypeDisease varchar (100) NOT NULL
);
go

create table TimeDim
(
	ID_Date char (3) NOT NULL primary key,
	YearDate varchar(100) NOT NULL,
	MonthDate varchar(100) NOT NULL,
	DayDate varchar(100) NOT NULL
);
go

create table ReceptionFact
(
	ID_Patient char(8) foreign key references PatientDim(ID_Patient) NOT NULL,
	ID_Doctor char(3) foreign key references DoctorDim(ID_Doctor) NOT NULL,
	ID_Service char(7) foreign key references ServiceDim(ID_Service) NOT NULL,
	ID_Disease char(5) foreign key references DiseaseDim(ID_Disease) NOT NULL,
	ID_Date char(3) foreign key references TimeDim(ID_Date) NOT NULL,
	MedicationsCount int,
	Duration int NOT NULL

	constraint PK_ReceptionFact primary key(ID_Patient, ID_Doctor, ID_Service, ID_Disease,
ID_Date)
);

go
```