

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ЗАКЛАД  
„ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ  
ІМЕНІ ТАРАСА ШЕВЧЕНКА”

Навчально - науковий інститут математики та інформаційних технологій

Кафедра математики та інформатики

**Щербак Михайло Сергійович**

**ДОСЛІДЖЕННЯ ТА РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ  
МЕДИЧНИ КАБІНЕТОМ**

кваліфікаційна робота  
здобувача вищої освіти другого (магістерського) рівня  
освітньої програми «Комп’ютерні науки та інформаційні  
технології»  
за спеціальністю 122 «Комп’ютерні науки»

Особистий підпис – \_\_\_\_\_

Науковий керівник – \_\_\_\_\_ Юрій КОЗУБ д.т.н. доцент

В.о. зав. кафедри – \_\_\_\_\_ Юрій КОЗУБ д.т.н. доцент

Полтава – 2024

## **АНОТАЦІЯ**

**Кваліфікаційна робота:** 71 стор., 27 рис., 1 табл., 27 дж., 2 додатка.

**Об'єкт дослідження** – технології розробки баз даних та програмного забезпечення на C#.

**Мета роботи** – розробити систему управління кабінетом сімейного лікаря “СімЛік”.

**Предмет дослідження** – методи та засоби розробки автоматизованої системи управління кабінетом сімейного лікаря.

**Методи дослідження** – метод моделювання, техніко-економічний із використанням комп'ютерних технологій, технічний аналіз, методи моделювання інформаційних процесів.

Проаналізовано засоби проектування та розробки баз даних та програмного забезпечення, визначено основні об'єкти предметної області, виділено потреби, якими необхідно забезпечити програмну підтримку. Вивчено існуючі СУБД, які можуть забезпечити зберігання даних системи. Обрано базу даних MS SQL для реалізації. Побудовано інфологічну модель даних. Створено інтерфейс ПЗ засобами C# та Windows Forms. Розроблено автоматизовану систему “СімЛік” для автоматизації роботи сімейного лікаря.

**Ключові слова:** ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, БАЗИ ДАНИХ, СІМЛІК, СКБД, ПРЕДМЕТНА ОБЛАСТЬ, ТАБЛИЦЯ, C#, .NET FRAMEWORK, SQL.

## ЗМІСТ

ВСТУП .....	5
РОЗДІЛ 1. ХАРАКТЕРИСТИКА СТАНУ АВТОМАТИЗОВАНОЇ МЕДИЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	8
1.1. Медичні інформаційні системи в Україні.....	8
1.1.1. Підсистема АСУ “Поліклініка” .....	9
1.1.2. Автоматизована система профілактичних оглядів (АСПОН) .....	11
1.1.3. Підсистема АСУ “Стаціонар” .....	13
1.2. Вимоги до системи управління кабінетом сімейного лікаря .....	17
1.3. Висновки до розділу 1 .....	20
РОЗДІЛ 2. АНАЛІЗ ЗАСОБІВ РОЗРОБКИ .....	21
2.1 Система управління базами даних Microsoft SQL Server .....	21
2.2 Середовище розробки Visual Studio 2019.....	23
2.3 Мова програмування C#.....	25
2.4 Технологія .NET .....	28
2.5 Технологія Windows Forms .....	30
2.6 Вимоги до системи.....	32
2.7 Висновки до розділу .....	32
РОЗДІЛ 3. АРХІТЕКТУРА ТА ОПИС РОЗРОБКИ.....	34
3.1 Архітектура додатку для роботи з БД.....	34
3.2. Опис системи.....	38
3.3. Висновки до розділу 3 .....	51
ВИСНОВКИ.....	53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	55
ДОДАТКИ.....	59
Додаток А. Лістинг класу frmConsultation.cs. ....	59
Додаток Б. Лістинг класу frmParent.cs.....	65

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АСОД	-	автоматизована система обробки даних;
АСМОН – ДМПО	-	автоматизована система масових оглядів населення з донозологічною багатoproфільною організацією;
АСОРС	-	автоматизована система оцінки ризику синдромів;
АСПОН	-	автоматизована система профілактичних оглядів;
АСУ	-	автоматизована система управління;
БД	-	база даних;
ІКТ	-	інформаційно - комунікаційні технології;
ІС	-	інформаційні системи;
КАСМОН	-	комплексна автоматизована система для масових оглядів населення;
ЛБ	-	лікарська база;
МІС	-	медична інформаційна система;
МОЗ	-	міністерство охорони здоров'я;
МСЧ	-	медико-санітарна частина;
НТП	-	науково технічний прогрес;
НФ	-	нормальна форма;
ОС	-	операційна система;
ПЗ	-	програмне забезпечення;
ПМД	-	первинна медична допомога;
ПК	-	персональний комп'ютер;
СУБД	-	система управління базами даних
ЕОМ	-	електронні обчислювальні машини
DEC	-	Digital Equipment Corporation;
SQL	-	Structured Query Language;
MGA	-	Multi - Generational Architecture;
MDA	-	Multi Dimensional Array;
ICPC	-	International Classification for Primary Care.

## ВСТУП

Інформаційні процеси, які включають в себе пошук, збирання, зберігання, передавання, опрацювання, використання та захист інформації, є необхідною частиною усіх галузей медицини і сфери охорони здоров'я. Ключовим компонентом цих інформаційних процесів є інформаційні потоки, які визначають ясність функціонування галузі в цілому і ефективність її управління. Починаючи з місць виникнення інформації, ці потоки забезпечують її передачу до місць прийняття рішень. Вони складаються з окремих повідомлень, які відображаються у сигналах і документах, і переміщуються в просторі та часі від джерела інформації до отримувача. Для ефективної роботи з інформаційними потоками застосовуються інформаційні системи (ІС).

3 грудня 2017 року в Україні розпочата медична реформа, а одним з її напрямків є впровадження в практику лікарів первинної ланки міжнародної класифікації первинної медичної допомоги (ІСРС-2-Е) [3]. У березні 2018 року МОЗ України схвалило наказ N504 «Про затвердження Порядку надання первинної медичної допомоги (ПМД)». Згідно з розділом III «Правила надання ПМД» пункту 4: «Відомості про кожен випадок надання ПМД лікар або інший медичний працівник, що входить до команди з надання ПМД, фіксуються в медичній документації відповідно до Міжнародної класифікації первинної допомоги (ІСРС-2-Е)».

Медичні працівники первинної ланки витрачають значну частину свого робочого часу для роботи з паперовими документами та статистикою, що обмежує їхню можливість віддавати достатньо уваги потребам пацієнтів. Згідно з наказом МОЗ України №157 «Про внесення змін до деяких наказів Міністерства охорони здоров'я України»[14], введено в дію нову форму Журналу реєстрації амбулаторних пацієнтів, і з впровадженням електронної системи охорони здоров'я, цей журнал повинен стати електронним. Всі записи в ньому мають автоматично підтягуватись при заповненні амбулаторної картки. Крім того, з початку спалаху COVID-19 і з

впровадженням карантину та обмежувальних заходів пов'язаних із поширенням коронавірусної хвороби обрана тема бакалаврської роботи стає більш актуальною.

Основним завданням у рамках дипломної роботи є розробка автоматизованої системи управління кабінетом сімейного лікаря. Ця система має вирішувати наступні завдання: збирати та зберігати дані, що стосуються об'єкта дослідження; забезпечувати формування загальних та деталізованих звітів за результатами роботи; надавати можливість легко визначати тенденції зміни ключових показників, які впливають на управлінські рішення; забезпечувати отримання інформації без істотних затримок; проводити точний і повний аналіз надходжених даних.

Призначення і мета створення підсистеми:

- Облік пацієнтів сімейного лікаря. Додавання і видалення пацієнтів облік захворювань і призначених процедур.
  - Облік процедур і ліків. Додавання і видалення процедур і ліків.
  - Облік відвідувань Додавання та видалення візитів пацієнтів
- Вивід на друк історії відвідувань.

Завдання впровадження:

- підвищити ефективність управління кабінетом сімейного лікаря;
- підвищити якість та оперативність прийняття рішень у процесі лікування пацієнтів;
- підвищити якість та зменшити тривалість обслуговування пацієнтів;
- підвищити ефективність праці медичного персоналу;
- зменшити терміни й спростити процедури підготовки звітних матеріалів за результатами роботи лікаря.

Робота містить вступ, три розділи, висновки, список використаних джерел, додатки.

В першому розділі розглянуто характеристику стану медичної

інформаційної системи України, вимоги до системи обліку медичного закладу, умови застосування автоматизованих систем для створення медико- комп'ютерних комплексів.

Другий розділ містить задачі, які потрібно виконати протягом роботи, розглянуто програмні засоби, що були застосовані для розробки проекту та обрано відповідну мову програмування та середовище розробки.

У третьому розділі обґрунтовано вибір середовища для розробки модулю. Розглянуто процес створення інтерфейсу додатка “СімЛік”, включаючи опис компонентів за допомогою яких реалізовано інтерфейс.

У додатках представлено фрагменти лістингу коду розробки.

## РОЗДІЛ 1

### ХАРАКТЕРИСТИКА СТАНУ АВТОМАТИЗОВАНОЇ МЕДИЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

#### 1.1. Медичні інформаційні системи в Україні

Головним закладом, який забезпечує надання лікувально-профілактичної допомоги населенню, є об'єднана лікарня, що включає в себе стаціонар і поліклініку (поліклінічне відділення). У таких об'єднаних лікарнях функціонує єдина адміністрація та спільні лікувально-діагностичні відділення, що надають медичну допомогу пацієнтам як у лікарні (у випадку багатoproфільної лікарні - в спеціалізованих відділеннях для лікування певних груп хвороб), так і в поліклініці, а також на дому.

Міська об'єднана лікарня спрямована в основному на надання медичної допомоги робітникам промислових або будівельних організацій. Вона є складовою медико-санітарної частини (МСЧ), що включає стаціонар і поліклініку.

Об'єднаний пологовий будинок є комплексною установою, яка забезпечує жінок, що проживають у районі її діяльності, усіма видами акушерсько-гінекологічної допомоги (поліклінічної, в жіночій консультації і стаціонарі).

Задачі по поліпшенню медичного обслуговування населення передбачають від організаторів охорони здоров'я необхідність вміння аналізувати статистичні дані різних лікувально-профілактичних установ та розробляти конкретні заходи для вдосконалення медичної допомоги [19].

Аналіз діяльності поліклініки і стаціонару об'єднаної міської лікарні (об'єднаного пологового будинку) необхідний:

1. Органам охорони здоров'я, відповідаючим за медичну допомогу населенню визначеної території і здійснюючим контроль за її



постановкою.

2. Керівництву лікарні (пологового будинку) – для оперативного управління установою.

3. Лікарям лікарні – для оцінки якості різних заходів, направлених на охорону здоров'я населення.

Глибокий аналіз дозволяє оцінити основні якісні показники діяльності відповідальної медичної установи в динаміці, виявити недоліки і успіхи у медичному обслуговуванні населення та визначити ключові заходи для його покращення. Використання механізованого обліку, сучасної техніки та проведення вибіркового дослідження для конкретизації окремих аспектів роботи лікувально-профілактичної установи, а також застосування формалізованої оперативно-облікової документації (стат-талон, історія хвороби і інші), які підлягають аналізу головним лікарем та його заступниками, визначається залежно від необхідності та терміновості [3].

Поліклінічна допомога є ведучим та масовим видом медичної допомоги міському населенню, становлячи до 80% від загальної кількості звернень за медичною допомогою. Ця допомога надається в поліклінічних відділах та міських поліклініках. Зважаючи на значущість поліклінічної допомоги для населення, розглянемо більш детально питання автоматизації поліклінічної служби в підсистемі АСУ "Поліклініка" [11].

#### *1.1.1. Підсистема АСУ "Поліклініка"*

Розроблена система АСУ "Поліклініка" знаходить широке застосування в лікувально-профілактичних установах.

З допомогою АСОД (автоматизованої системи обробки даних) АСУ "Поліклініка" забезпечує:

- 1 – оперативний облік роботи всіх лікарів поліклініки;
- 2 – аналіз діяльності поліклініки в цілому;
- 3 – аналіз загальної захворюваності і захворюваності з тимчасовою втратою працездатності;

- 4 – інтенсифікацію праці лікарів в результаті розробки раціональних графіків їх роботи;
- 5 – ефективність організації диспансерного обстеження і т.ін.;
- 6– своєчасний збір і обробку інформації про діяльність лікарів поліклініки з метою забезпечення ритмічності в роботі;
- 7– оптимальний розподіл всієї роботи на профілактичний і лікувальний розділи;
- 8– визначення структури відвідування і т.д.

Дані, отримані АСОД, щомісячно поступають в розпорядження завідуючих відділеннями, головних лікарів і їх замісників, головних спеціалістів Міськохорони здоров'я (по профілям служб).

Автоматизована система обробки даних (АСОД) в складі автоматизованої системи управління "Поліклініка" забезпечує ефективне управління всіма розділами поліклінічної допомоги мешканцям міста. Накопичений досвід використання електронних обчислювальних машин для оперативної обробки стандартизованих документів поліклініки (таких як талон медичного обстеження, формалізована амбулаторна карта і інші) дозволяє впроваджувати єдиний уніфікований талон для запису прийому пацієнтів лікарем, медичних обстежень і диспансерного спостереження. Інформація, отримана з талонів та інших вхідних документів, дозволяє застосовувати комп'ютери для створення різноманітних баз даних, що відображають функціонування поліклінічних служб:

- про особи, підлягаючі медичному огляду;
- по розділу "Диспансеризація";
- про роботу дільничних лікарів і т.ін.

1)

Отримана інформація в рамках автоматизованої системи управління "Поліклініка" може бути використана у системі управління охороною здоров'я більш вищого рівня. Кожна поліклініка, виконуючи свої звичайні

обов'язки, функціонує одночасно як консультативний центр. Таким чином, виникає комплекс завдань "Консультативний центр", який включає в себе: 1 – аналіз різних аспектів роботи лікарів різних спеціальностей; 2 – контроль обґрунтованості направлення пацієнтів на консультації; 3 – аналіз і контроль повноти обстеження і якості діагностики як на місцях, так і в консультативному центрі; 4 – управління диспансеризацією пацієнтів із визначеними захворюваннями. Вирішення цих завдань за допомогою електронно-обчислювальних машин дозволяє:

1. підвищити якість медичного обслуговування;
2. скоротити час очікування на консультації;
3. сприяє формуванню плану диспансеризації [12].

### *1.1.2 Автоматизована система профілактичних оглядів (АСПОН)*

На даний час створено ряд автоматизованих систем для масового огляду населення:

КАСМОН – комплексна автоматизована система для масових оглядів населення.

АСМОН – ДМПО – автоматизована система масових оглядів населення з донозологічною багатопрофільною організацією.

АСОРС – автоматизована система оцінки ризику синдромів. АСПОН – автоматизована система профілактичних оглядів.

Програмою передбачено розгляд лише останньої, оскільки решта систем аналогічні по структурі з АСПОН.

АСПОН створюється в вигляді розгалуженої сітки терміналів (дисплеїв) автоматизованих медичних приладів і апаратів з мікропроцесорами, підключеними до міні-ЕОМ [12].

З допомогою АСПОН вирішуються такі задачі:

- автоматизована обробка даних лабораторних досліджень;
- автоматизована розшифровка ЕКГ;

- аналіз біохімічних досліджень;
- аналіз антропометричних досліджень;
- аналіз флюорографічних досліджень;
- обробка анамнестичних даних і результатів лікарських оглядів лікарями-спеціалістами;
- установка діагнозів методами лікарсько-машинної діагностики;
- здійснення планування, обліку і контролю проведенням профілактичних оглядів і диспансеризації в цілому організованих колективів населення;
- визначення професійної придатності;
- складання планів лікувально-оздоровчих заходів;
- обробка матеріалу для статистичної звітності.

З урахуванням пропускнуої здатності автоматизованої системи профілактичних оглядів населення (від 25 до 50 тисяч чоловік щорічно при неперервній роботі), найбільш доцільним варіантом є створення її в міських об'єднаних лікарнях великих підприємств або об'єднань з використанням Інформаційно-вимірювальних комплексів (ІВЦ). Наприклад, на Запорізькому автозаводі "Комунар" діє автоматизована поліклініка для проведення профілактичних оглядів. Щороку більше ста тисяч робітників може пройти огляд в цьому закладі.

У об'єднаних підприємствах "АвтоЗАЗ", за участю Запорізького обласного відділу охорони здоров'я та Міністерства охорони здоров'я, в систему автоматизованої системи медичного обслуговування населення (АСМОН) включені підсистеми, такі як "Реєстратура", "Анамнез" та інші, відповідно до визначених завдань. Після завершення опитувань і обстежень електронно-обчислювальна машина надає табуляції з усіма даними та попереднім висновком щодо стану здоров'я і відповідних рекомендацій (консультації спеціалістів, додаткові обстеження, стаціонарне лікування і т.д.) [16].

### 1.1.3 Підсистема АСУ “Стаціонар”

Згідно з заданою програмою, після обробки та формалізації інформації комп'ютер генерує визначену кількість таблиць з вихідними даними, які щомісячно аналізуються головним лікарем та його заступниками. Табличні зведення дозволяють отримати достовірну інформацію:

1. про склад хворих;
2. про середню тривалість перебування хворих у стаціонарі;
3. про непоказану і непрофільну госпіталізацію;
4. про своєчасність початого лікування та його якість;
5. про якість і кількість обстежень на догоспітальному етапі;
6. про розходження діагнозів;
7. про строки доставки хворих за екстреними показниками;
8. про результат захворювання;
9. про працездатність виписаних хворих.

Характеристика соціального складу вибувших хворих, також враховуючи такі аспекти, як стать, вік, професія, профіль захворювання тощо, на рівні міста, району чи республіки, дозволяє здійснювати планування мережі спеціалізованої допомоги [19].

Обробка формалізованих карт пацієнтів, які покидають стаціонар, призводить до:

1. підвищення інформативності статистичних даних,
2. об'єктивної оцінки показників діяльності стаціонару,
3. розробки та вчасної реалізації комплексу заходів для поліпшення роботи стаціонару.

Підсистема автоматизованої системи управління "Стаціонар" потребує використання методів автоматизованого формування розкладу роботи діагностичних відділів лікарень, що пов'язано з впровадженням нових діагностичних засобів та значними труднощами у традиційному

обслуговуванні заявок. Завдяки комп'ютерам різних типів та комплексів (в залежності від потужності лікарні) вирішуються задачі диспансеризації, ведення статистики, контроль ліжкового фонду, управління аптекою та інші.

Технологія експлуатації комплексу завдань (підсистеми) "Диспансеризація" передбачає взаємодію медичного персоналу відділень, лабораторій та операторної комп'ютерної системи. Відповідні заявки виводяться на відеотермінали за допомогою мультитермінального макетного введення (зазначаються номери карток пацієнтів та числові коди призначених досліджень). Відеотермінали надають можливість візуально переглядати інформацію, виправляти помилки за допомогою програми макетного редагування та змінювати нормативи проведення досліджень.

Завдяки автоматизації діагностичних підрозділів значно зменшується обсяг роботи, пов'язаний з заявками медсестер відділень, а персонал вільний від планування робочого графіку, отримуючи чіткий розклад на основі нормативних вимог.

До речі, слід зазначити, що на основі автоматизованої системи аналізу травматизму, працевтрат і захворюваності (АСАТІЗ) була розроблена система автоматизованого управління періодичними медичними оглядами (АСУ ГМО). У цій системі комп'ютер враховує категорії осіб, які підлягають оглядам (особливо це стосується працівників промислових підприємств), класифікує їх за впливом шкідливих факторів з урахуванням ступеня ризику для основних синдромів можливих захворювань. Визначаються не лише обсяги необхідних додаткових обстежень (залучення фахівців, лабораторно-інструментальні дослідження і т.д.), але й порядок їх проведення (терміновість) та послідовність для кожного обстежуваного.

Час, витрачений на видачу направлень для 600–700 осіб, становить 40–50 хвилин. Після завершення огляду комп'ютер відповідає за контроль за виконанням усіх рекомендованих комісійних призначень. Застосування анкетного методу дозволяє здійснювати динамічний моніторинг стану

здоров'я працівників окремих цехів заводу, підприємства або інших груп на основі кількісних характеристик ризику (в відсотковому співвідношенні).

Запропонована методика визначення рівня ризику за синдромами (від 0 до 1,0) може бути удосконалена шляхом додавання додаткових запитань та розширення переліку аналізованих синдромів. Осіб із середнім рівнем ризику (від 0,5 до 0,9) слід розглядати як "групу уваги", а тих, чий ризик становить від 0,99 до 1,0, можна визначити як осіб з високим (оптимальним) ступенем ризику (фактично хворі). Отримані таким чином дані можуть бути використані для формування систем управління медичними обстеженнями (АСУ ГМО або АСУ "Диспансеризація").

Сами ці системи можуть бути використані для ефективного управління процесом диспансеризації за умови збору інформації про кожного обстеженого в пам'яті комп'ютерної системи. У медичній службі Цивільної оборони лікарня (стаціонар) розглядається як лікувальний заклад, включений до складу лікарняної бази, яка призначена для надання висококваліфікованої та спеціалізованої медичної допомоги та лікування заражених хворих в умовах можливого військового нападу.

У цьому випадку АСУ "Стаціонар" може бути використана для обчислення даних з медичної бази (ЛБ) – це етап медичної евакуації в системі Цивільної оборони. Медична база представляє собою сукупність лікувальних установ, які є частиною єдиного керівництва, розташованих в приміській зоні в евакуаційному напрямку. Це забезпечує спеціалізовану медичну допомогу та лікування враженим і хворим до досягнення визначеного результату враження (захворювання).

Математичне забезпечення виступає організуючим принципом всієї сучасної технології діагностики і лікування. Останнім часом в стаціонарах виникли автоматизовані робочі місця. Малогабаритний персональний комп'ютер із вбудованою системою, яка використовується в цьому контексті, одночасно виконує функції:

1. довідкового відділу,
2. архівного відділу,
3. бібліографічного каталогу,
4. електронної записної книги,
5. щотижневого звіту,
6. обчислювальної системи тощо. Це допомагає лікареві витратити

менше часу на рутинні аспекти роботи, дозволяючи більше уваги приділяти творчому процесу лікування та глибокому аналізу власної діяльності.

Автоматизовані робочі місця об'єднуються в одну мережу, яка включається в систему автоматизованого управління стаціонаром (АСУ "Стаціонар"). Ця мережа робить ресурси, пам'ять, програми і технічні пристрої кожного робочого місця доступними для всіх осіб, які працюють з персональними комп'ютерами в мережі. Такий підхід дозволяє створити єдину базу даних, яка містить інформацію про кожного хворого.

Завдяки цій системі можна отримати відомості про хворих, які перебувають на лікуванні з приводу недостатності серця і легень, виявити можливі ускладнення та порівняти різні методи лікування та їх результати за кілька хвилин.

З використанням вдосконалених автоматизованих прогностичних моделей можна здійснити наступні завдання:

1. визначити значення кожного фактора та їх комбіновану дію на формування здоров'я;
2. оцінити ефективність вжитих оздоровчих заходів як в комплексі, так і окремо для кожного спостережуваного;
3. розрахувати оптимальний комплекс лікувально-оздоровчих процедур для індивідууму та різних профільних груп;
4. прогнозувати стан здоров'я різних контингентів за введення в систему різних параметрів;
5. керувати диспансеризацією на різних рівнях.



На даний момент утруднено створення прогностичних моделей на основі автоматизованих систем з використанням ЕОМ через вирішення проблеми інтегрованого (збагаченого) показника ефективності, який повинен відповідати різноманітним вимогам, таким як адекватність, комплектність, наукове обґрунтування, об'єктивність і інше.

Використання багатокрокового кореляційно-регресивного аналізу дозволяє описати виявлені залежності за допомогою рівнянь регресії, які є організаційно-статистичною моделлю. В цьому процесі автоматизовані системи відіграють важливу роль.

Використання автоматизованих систем для оцінки ризику основних патологічних синдромів дозволяє вирішувати завдання прогнозування стану здоров'я великих груп населення. У цьому випадку рекомендовано використовувати середні значення ризиків для популяційних оцінок. Застосування математичних методів обробки часових рядів для аналізу динаміки середніх ризиків дозволяє отримувати короткострокові (до 5 років) прогнози у формі очікуваних значень середніх ризиків для основних захворювань (передзахворювань) та загальної характеристики стану здоров'я населення. Ця система надає можливість:

1. точно враховувати обслуговуючі групи населення і на цій основі оптимізувати розподіл ресурсів;
2. автоматизувати планування всіх видів обстежень, диспансерний контроль за хворими на хронічні захворювання та пацієнтами з факторами ризику, а також періодичні профілактичні огляди працездатного населення [19].

## **1.2 Вимоги до системи управління кабінетом сімейного лікаря**

Система обліку відвідування є ефективним засобом контролю використання послуг і робочого часу. Основним методом досягнення поставленої мети є створення банку спеціалізованих даних, що зберігають

інформацію про пацієнтів, видах послуг, графіку відвідувань, лікарів. Дослідження підприємства направлено на виявлення можливостей підвищення ефективності управління підприємством на базі використання засобів обчислювальної техніки і сучасних економіко-математичних методів, а так само на збільшення обсягу та підвищення якості послуг, що надаються при існуючих технічних засобах.

Вирішення завдань управління передбачає інтенсивний обмін інформацією між всіма учасниками системи, як між людьми, так і між людьми та персональними комп'ютерами (ПК). Існують два основних види обміну інформацією - документований і не документований.

Документований обмін - це передача документів, головним чином на папері, які підготовлені і заповнені людьми або ПК у вигляді роздруківок.

Не документований обмін інформацією може бути використаний для отримання негайних відомостей від джерела, усної передачі розпоряджень, підтвердження виконання отриманих наказів чи команд, повідомлення про ситуації, які вимагають термінового втручання і т. д. Він відзначається швидкістю отримання інформації, відсутністю обмежень на зміст питань і відповідей, а також можливістю введення будь-яких уточнень і деталізації.

Основним недоліком не документованих повідомлень є відсутність правової та юридичної відповідальності джерела інформації за достовірність і точність виданої інформації, а також неможливість подальшої перевірки змісту раніше надісланого повідомлення. У випадках, пов'язаних з матеріальною, фінансовою або іншою підвищеною відповідальністю, використовуються виключно документовані повідомлення.

Основні типи документів з точки зору відділів їх ведення класифікуються на: вхідні, вихідні та внутрішні. До внутрішніх можна віднести організаційно-розпорядчу документацію. В системі організаційно-розпорядчої документації виділені три групи документів: організаційна, в яку входять статuti, інструкції, правила; розпорядча -

постанови, розпорядження, накази, рішення; довідково- інформаційна - листи, довідки, протоколи, акти [17].

В процесі виконання об'єднання потоків вхідних і вихідних документів формуються типові групи документів: договірні документи, бухгалтерські документи, документи підтримки бізнес і впровадження нових технологічних процесів, звітні документи.

В сучасних системах управління є програмне забезпечення, що дозволяє автоматизувати створення форми документа за мінімальною інформації, що повідомляється користувачем.

Дані про відвідувача вносяться в документи зі слів відвідувача. У базі даних необхідно зберігати наступну інформацію про пацієнта:

- прізвище;
- ім'я;
- по батькові;
- адреса;
- номери стаціонарного і мобільного телефонів.

Основним користувачем програмного комплексу є адміністратори.

Часто виконуються запити з боку адміністраторів:

- інформація про послуги;
- інформація про пацієнтів;
- інформація про відвідування;
- інформація про оплатах;
- дані про співробітників.

З огляду на вищесказане, можливо визначити завдання, які ставить перед собою оператор, звертаючись до бази даних:

- пошук пацієнта;
- пошук інформації про оплату;
- отримання звітів [2, 3, 25].

Таким чином, приходимо до висновку, що для реалізації

автоматизації обліку необхідно розробити базу даних і програмне забезпечення для роботи з нею.

### **1.3 Висновки до розділу 1**

Медико-комп'ютерні комплекси можуть організовуватись на базі мікро - ЕОМ в поліклініках, лікарнях та інших установах охорони здоров'я. Здійснення поставлених задач в повному об'ємі можливо при умові застосування автоматизованих систем і електронно-обчислювальної техніки, створення медико-комп'ютерних комплексів. Багато питань застосування автоматизованих систем з метою диспансеризації вимагають подальшого їх вирішення. Необхідно перебороти скептицизм, психологічний бар'єр і проїнятися духом часу, тим більше, що є всі передумови і реальні можливості створення автоматизованих систем для масових профілактичних оглядів і диспансеризації. Таким чином, для реалізації автоматизації управління кабінетом сімейного лікаря необхідно розробити базу даних і програмне забезпечення для роботи з нею.

## РОЗДІЛ 2

### АНАЛІЗ ЗАСОБІВ РОЗРОБКИ

При розробці програмного продукту важливим фактором є правильний вибір засобів програмної реалізації та технологій, які впливають на час розробки, якість та надійність продукту.

При створенні програми було обрано такі засоби:

- система керування базами даних Microsoft SQL Server 2019;
- середовище розробки Microsoft Visual Studio 2019;
- мову програмування C#;
- програмну технологію .NET Framework 4.5;
- технологію Windows Forms;

Для розробки було використано ОС Windows 10, основними перевагами якої є:

- зручний інтерфейс;
- стабільність роботи системи;
- популярність використання.

Також операційна система Windows 10 має точки відновлення, тому в разі збою в роботі системи чи непередбачених обставин можливе повне відновлення документів та інших даних. Система є багатофункціональною, при встановленні має стандартний пакет драйверів.

Для розробки інтерфейсу користувача було обрано середовище Microsoft Visual Studio 2019, використана база даних, що створена в MS SQL Server 2019. Для розробки графічного середовища й інтерфейсу користувача використано технологію Windows Forms.

#### 2.1 Система управління базами даних Microsoft SQL Server

Система управління базами даних Microsoft SQL Server вважається однією з найпопулярніших у світі. Вона є відмінним вибором для

різноманітних проектів, включаючи невеликі додатки та великі високонавантажені проекти.

Розроблену компанією Microsoft, SQL Server вперше вийшов у 1987 році. Починаючи з версії 2016, ця СУБД стала доступною також для операційних систем Linux. Важливими характеристиками SQL Server є:

- Висока продуктивність, яка дозволяє їй ефективно працювати.
- Надійність та безпека, підтримка шифрування даних.
- Простота в адмініструванні і використанні.

Однією з ключових складових MS SQL Server, як і в будь-якій СУБД, є база даних. База даних - це сховище даних, організованих за реляційною моделлю, яка була розроблена ще Едгаром Коддом у 1970 році і зараз є стандартом організації баз даних.

Модель даних, заснована на взаємозв'язках, включає у себе зберігання інформації у вигляді таблиць, кожна з яких має свої рядки та стовпці. Кожен рядок відображає окремий об'єкт, а атрибути цього об'єкта представлені в стовпцях.

Для унікальності кожного рядка в межах таблиці використовується первинний ключ (primary key), який може складатися з одного або кількох стовпців. Використовуючи первинний ключ, ми можемо вказати на конкретний рядок в таблиці, і два рядки не можуть мати однаковий первинний ключ.

Ключі дозволяють встановлювати зв'язки між різними таблицями, що дозволяє організувати відносини між ними. Крім того, таблиця може бути представлена у вигляді відносин ("relation").

Одним із основних способів взаємодії з базою даних SQL Server є використання запитів, які описуються мовою SQL. Запит є декларативним і вказує, які дані потрібно отримати. Процесор запитів обробляє запит і визначає послідовність кроків для отримання необхідної інформації, що

називається планом запиту. План може бути обраний серед кількох можливих варіантів для оптимізації виконання запиту [4].

Основною мовою запитів для SQL Server є Transact-SQL, розроблена спільно Microsoft із Sybase. Transact-SQL є реалізацією стандарту ANSI/ISO для структурованої мови запитів (SQL) з розширеннями, що значно розширюють можливості мови SQL і рекомендуються для використання у різноманітних завданнях.

Залежно від завдання, яке виконує команда T-SQL, вона може відноситися до одного з наступних типів:

а) DDL (Data Definition Language / Мова визначення даних), яка охоплює команди для створення, зміни та видалення об'єктів бази даних, таких як таблиці, індекси та збережені процедури.

б) DML (Data Manipulation Language / Мова маніпуляції даними), яка включає команди для вибірки, оновлення, додавання та видалення даних.

в) DCL (Data Control Language / Мова управління доступом до даних), що включає команди для керування правами доступу до даних, такими як надання та відкликання прав.

Особливості T-SQL включають обмеження щодо використання псевдонімів в операторах WHERE та HAVING, вимогу явної вказівки приналежності поля до таблиці у разі використання JOIN, та підтримку різних видів з'єднань, таких як EXCEPT і INTERSECT [1, 5]..

## 2.2 Середовище розробки Visual Studio 2019

Середовище Microsoft Visual Studio — продукт розроблений компанією Microsoft, до якого входить інтегроване середовище розробки програмного забезпечення та ряд інших засобів розробки (рисунок 3.1).

Даний продукт дозволяє розробляти:

— додатки з графічним інтерфейсом з підтримкою технології Windows Forms;

- консольні додатки;
- веб-додатки;
- веб-служби;
- веб-сайти;

Ці додатки використовуються для всіх платформ, що підтримують Windows, .NET Framework, Silverlight та Windows Phone .NET Compact Framework.

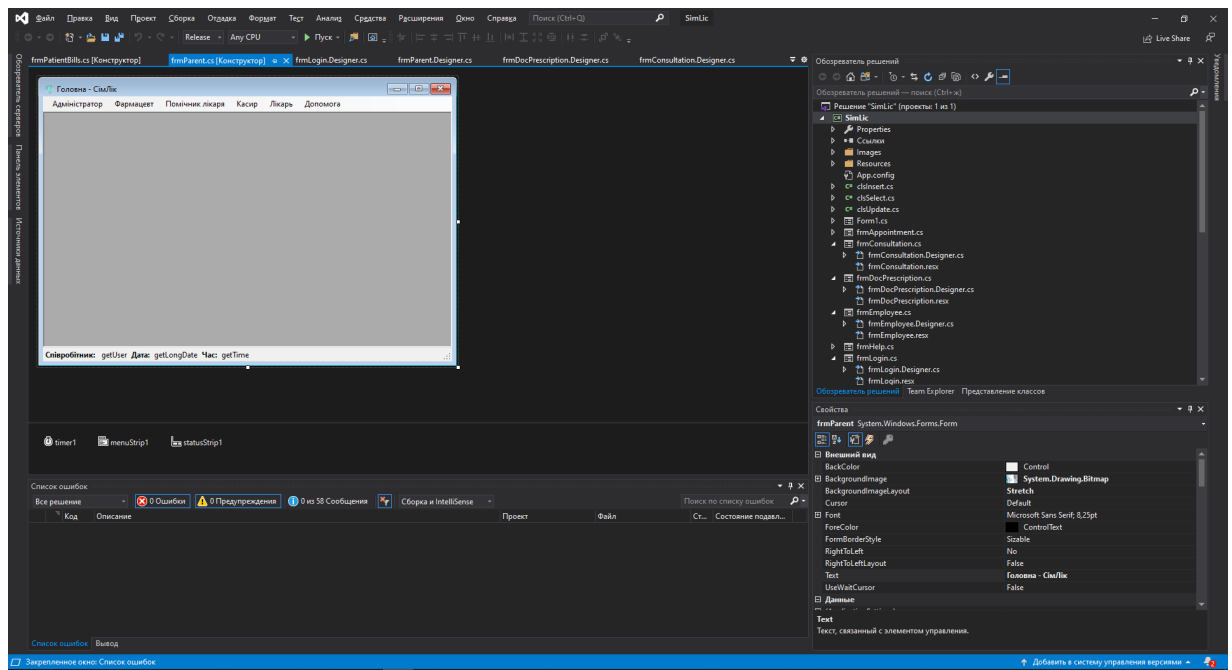


Рис. 2.1. Середовище розробки Microsoft Visual Studio 2019

У Visual Studio представлено різноманітні інструменти, включаючи редактор коду з можливостями рефакторінгу та підтримкою технології IntelliSense. Також доступний редактор форм для зручного створення графічного інтерфейсу користувача, дизайнер схеми бази даних, дизайнер класів та веб-редактор. Вбудований відладчик працює як на рівні машинного коду, так і на рівні вихідного коду.

Visual Studio дозволяє створювати та підключати сторонні додатки для розширення можливостей на різних рівнях, додаючи нові інструменти, наприклад, для візуального дизайну та редагування. Також воно підтримує



системи контролю версій вихідного коду та надає інструменти для інших аспектів розробки програм, таких як клієнт для роботи з Team Foundation Server — Team Explorer.

Архітектура Visual Studio розрахована на використання доповнень (Add-Ins) — плагінів від сторонніх розробників, що надає можливість розширювати функціональність середовища розробки.

### 2.3 Мова програмування C#

Мова програмування C# є об'єктно-орієнтованою мовою, призначеною для розробки на платформі .NET, і вона має безпечну систему типізації. Розроблена командою, до якої входили Скот Вілтамут, Пітер Гольде і Андерс Гейлсберг під егідою Microsoft Research (у складі компанії Microsoft) [21]. Синтаксис C# є схожим до Java та C++, і він успадкував ряд особливостей від інших мов програмування, таких як Delphi, C++, Smalltalk і Модуль.

Мова C# виключає деякі моделі, які виявилися проблематичними при розробці програмних систем, наприклад, множинне наслідування класів (на відміну від C++). Вона підтримує поліморфізм, перевантаження операторів, строгу статичну типізацію, вказівники на функції-члени класів, події, атрибути, винятки та коментарі у форматі XML.

C# була спеціально розроблена як мова прикладного рівня для Common Language Runtime (CLR), і вона залежить від можливостей самої CLR. Властивості мови обумовлені тим, чи може конкретна мовна функція бути трансльована у відповідні конструкції CLR. CLR пропонує C# та іншим мовам багато функціональних можливостей, які відсутні у "класичних" мовах програмування. Наприклад, в самому C# відсутня реалізація збірки сміття, яка здійснюється CLR для програм, написаних на C#. Такі можливості взаємодії можуть продовжити розвиватися, але ця послідовність може бути порушена при виході C# 3.0, який є розширенням мови, незалежним від розширень платформи .NET.

Мова програмування C# компілюється за допомогою компілятора Microsoft Visual C#. Однак існують інші компілятори C#, які часто включають реалізації Common Language Infrastructure та бібліотеки класів .NET. До них належать:

- Проект SharpDevelop, що є альтернативою Visual Studio і належить компанії icsharpcode. Він надає повну реалізацію Common Language Infrastructure.
- Проект Microsoft Rotor (тепер називається Shared Source Common Language Infrastructure), який ліцензований лише для дослідницького та навчального використання. Він надає реалізацію CLR runtime та компілятора C#, відповідно до специфікації ECMA, але тільки для Windows XP до C# 2.0.
- Проект DotGNU, який також має відкритий компілятор C# та повну реалізацію Common Language Infrastructure з підтримкою бібліотек до .NET 2.0.
- Проект Mono, що надає відкритий компілятор C#, відкриту та повну реалізацію Common Language Infrastructure, а також бібліотеки фреймворка відповідно до специфікації ECMA.
- DotNetAnywhere Micro Framework CLR, спрямований на вбудовані системи, підтримує практично всі специфікації C# 2.0.

Розглянемо деякі особливості мови програмування C#:

- **Портативність:** C# була створена як мова програмування на прикладному рівні для Common Language Runtime (CLR), і, отже, вона перш за все залежить від функціоналу CLR. Це особливо стосується типізації в C#, де наявність певних особливостей мови залежить від їх трансляції в конкретні конструкції CLR. Розвиток CLR від версії 1.1 значно розширив можливості мови програмування C#, і подібної взаємодії можна чекати в майбутньому, за винятком C# 3.0, яке є розширенням мови, не базованим на розширенні платформи .NET.
- **Типи даних:** Другою особливістю є типи даних в мові C#. Головні типи даних розглядаються у таблиці 2.1.
- **Метапрограмування:** Третьою особливістю є метапрограмування, яке досягається за допомогою використання атрибутів у C#. Деякі з цих атрибутів

відображають функціональні особливості директив препроцесора, спрямованих на платформу Visual C++ та GCC [8, 26].

Таблиця 2.1 — Основні типи даних мови програмування C#

Type	Represents	Range	Default Value
Bool	Boolean value	True or False	False
Byte	8-bit unsigned integer	0 to 255	0
Char	16-bit Unicode character	U +0000 to U +ffff	'\0'
Decimal	128-bit precise decimal values with 28-29 significant digits	$(-7.9 \times 10^{28} \text{ to } 7.9 \times 10^{28}) / 100 \text{ to } 28$	0.0M
Double	64-bit double-precision floating point type	$(+/-)5.0 \times 10^{-324} \text{ to } (+/-)1.7 \times 10^{308}$	0.0D
Float	32-bit single-precision floating point type	$-3.4 \times 10^{38} \text{ to } +3.4 \times 10^{38}$	0.0F
Int	32-bit signed integer type	-2,147,483,648 to 2,147,483,647	0
Long	64-bit signed integer type	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0L
Sbyte	8-bit signed integer type	-128 to 127	0
Short	16-bit signed integer type	-32,768 to 32,767	0
UInt	32-bit unsigned integer type	0 to 4,294,967,295	0
Type	Represents	Range	Default Value
Ulong	64-bit unsigned integer type	0 to 18,446,744,073,709,551,615	0
Ushort	16-bit unsigned integer type	to 65,535	0

## 2.4 Технологія .NET

Технологія Microsoft .NET є програмною платформою, розробленою компанією Microsoft для створення як веб-застосунків, так і звичайних програм. Багато з принципів та ідей винесено з технології Java, і однією з ключових концепцій .NET є сумісність служб, написаних різними мовами [23]. Хоча Microsoft виділяє цю можливість як перевагу .NET, аналогічна можливість також існує у платформі Java.

Програми, розроблені на мові C#, виконуються в середовищі .NET Framework (див. рис. 2.2) — інтегрованому компоненті для операційної системи Windows. Це середовище включає віртуальну систему виконання (CLR-середовище), уніфікований набір бібліотек класів для всіх мов платформи .NET, відомий як BCL (Base Class Library), а також бібліотеку класів FCL (Framework Class Library) з розширеними компонентами, такими як ASP.NET, ADO.NET, Windows Forms, WPF.

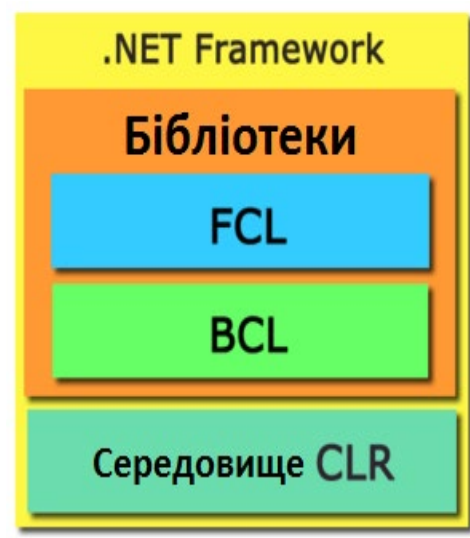


Рис. 2.2. Компоненти платформи .NET Framework

Кожна бібліотека в .NET має свою власну версію, що допомагає уникнути можливих конфліктів між різними версіями збірок. Технологія .NET є крос-платформенною, і наразі існують реалізації для платформ Microsoft Windows, FreeBSD (надана Microsoft), а також варіант технології для операційної системи Linux в рамках проекту Mono (згідно з угодою між Microsoft і Novell).

Забезпечення захисту авторських прав в .NET пов'язане з створенням середовища виконання (CLR — Common Language Runtime) для програм .NET. Компілятори для .NET надаються фірмами у вільному доступі для різних мов.

.NET розділяється на дві основні частини: середовище виконання (фактично, віртуальна машина) та інструментарій розробки. Середовище CLR (див. рисунок 2.3) є комерційною реалізацією інфраструктури CLI (common language infrastructure), що є міжнародним стандартом та базою для середовищ виконання та розробки з тісною взаємодією мов і бібліотек.

Вихідний код, написаний мовою C#, компілюється в проміжну мову (IL) згідно зі специфікацією CLI. Код IL і ресурси, такі як растрові зображення і рядки, зберігаються на диску у виконуваному файлі-збірці з розширенням EXE або DLL у більшості випадків.

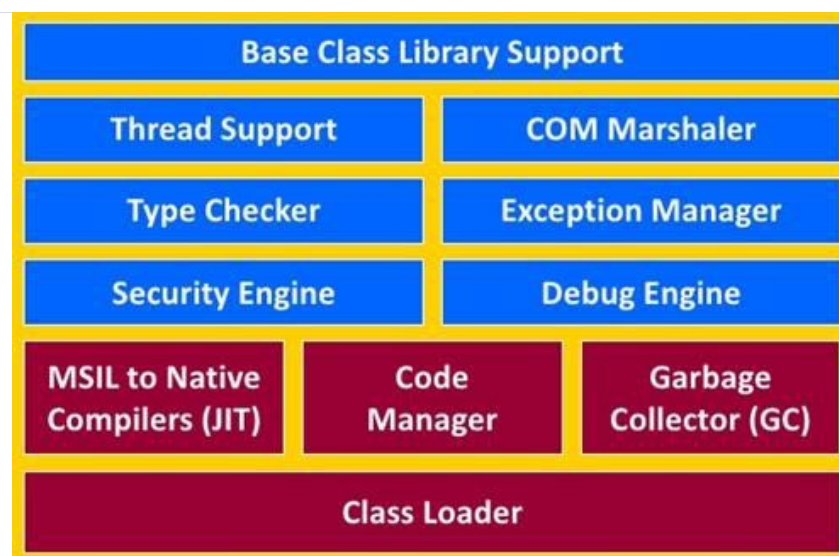


Рис. 2.3. Архітектура середовища CLR

При виконанні C#-програми, згідно з інформацією у маніфесті, збірка завантажується в збірку CLR. Після цього, якщо вимоги безпеки виконані, середовище CLR використовує JIT-компіляцію для перетворення коду IL в інструкції машинного коду. Крім того, середовище CLR забезпечує інші служби, такі як автоматичне видалення сміття, обробка винятків та управління ресурсами.

Також важливо зазначити, що один із перших компіляторів Java був розроблений компанією Microsoft. Зараз використовується HotSpot для Java, що є більш вдосконаленою багаторівневою системою компіляції. Якість конкретного компілятора часто визначає швидкодію, і сучасна технологія динамічної компіляції дозволяє досягти аналогічного рівня швидкодії порівняно з традиційними "статичними" компіляторами, такими як C++ [21, 23] .

## 2.5 Технологія Windows Forms

Технологія Windows Forms — інтерфейс програмування додатків (API), що відповідає за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки для існуючого Win32 API в керованому коді. Причому керований код — класи, що реалізують API для Windows Forms, не залежать від мови розробки. Тобто програміст однаково може використовувати Windows Forms як при написанні ПЗ на C#, C ++, так і на VB.Net, J# та інших.

З одного боку, Windows Forms розглядається як заміна більш старої і складної бібліотеки MFC, спочатку написаної на мові C++. З іншого боку, WF не пропонує парадигму, порівнянну з MVC. Для виправлення цієї ситуації і реалізації даної функціональності в WF існують сторонні бібліотеки.

Усередині .NET Framework, Windows Forms реалізується в рамках простору імен System.Windows.Forms.

Додаток Windows Forms, який базується на Microsoft .NET Framework, орієнтований на події. У порівнянні з пакетними програмами, значна частина часу витрачається на очікування взаємодії користувача, такої як введення тексту в текстове поле або натискання кнопки мишею. Для подальшого розвитку Microsoft досягла успіху, використовуючи WinFormi з XAML у фреймворках, таких як WPF та UWP. Однак перетягування графічних компонентів інтерфейсу, залишаючись в стилі WinForms, продовжується за допомогою XAML, замінюючи кореневий елемент XAML сторінки або вікна за допомогою інтерфейсу користувача Canvas. За внесенням цієї зміни користувач може конструювати вікно аналогічно для WinForms, використовуючи безпосереднє пертягування компонентів за допомогою графічного інтерфейсу Visual Studio.

Хоча XAML забезпечує можливість перетягування та зворотню сумісність місць розташування через контрольну панель Canvas, слід відзначити, що елементи керування XAML, хоа схожі на WinForm Controls, несумісні між собою. Вони мають подібні функції та мають схожий зовнішній вигляд, але їх властивості та методи відрізняються настільки, що виникає необхідність у перенаправленні API зодного на інший [4, 21, 23].

На рис. 2.4 показано приклад вікна побудованого за допомогою Windows Forms.

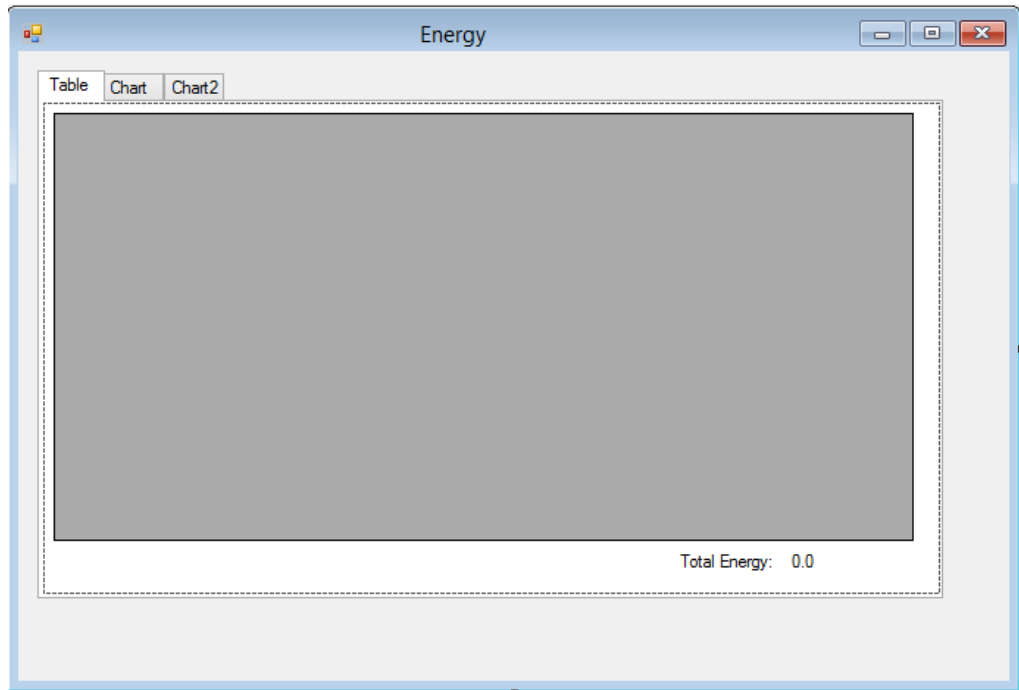


Рис. 2.4. Приклад Windows Forms

## 2.6 Вимоги до системи

Рекомендованими версіями операційних систем для нормальної роботи додатку є Windows 7, Windows 8 та Windows 10. Для запуску програми на персональному комп'ютері має бути встановлена версія .NET Framework 4.5. не нижче

## 2.7 Висновки до розділу

У цьому розділі розглянуті інструменти, які були використані при розробці проєкту. Оскільки розроблена система призначена для операційної системи Microsoft Windows, були вибрані відповідна мова програмування та середовище розробки. Вибір технологій ґрунтувався на їхній зручності використання, відкритості вихідних кодів та актуальності. Узгоджене використання цих технологій надає можливість створити якісний та надійний продукт, який захищений від можливих патентних вимог з боку розробників, оскільки всі ці технології мають ліцензії, що виключають такі претензії і надають доступ до вихідних кодів відповідних проєктів.





## РОЗДІЛ 3

### АРХІТЕКТУРА ТА ОПИС РОЗРОБКИ

#### 3.1 Архітектура додатку для роботи з БД

Для роботи з серверною частиною використовується принцип "Onion Architecture".

Більшість традиційних архітектур висувають ключові питання стосовно жорсткого зв'язку та розділення проблем. Модель архітектури Onion, розроблена Джеффри Палермо, була представлена з метою надання ефективного підходу до розробки додатків, сприяючи покращенню тестованості, ремонтпридатності та надійності в перспективі. Onion Architecture вирішує труднощі, що виникають при використанні трирівневих та n-ярусних архітектур, а також пропонує рішення загальних проблем. Взаємодія між шарами цибульової архітектури реалізується через використання інтерфейсів.

Основна ідея Onion-архітектури ґрунтується на принципі інверсії управління. Архітектура цибулі складається з кількох концентричних шарів, що взаємодіють між собою відносно ядра, яке втілює домен. На відміну від класичних багаторівневих архітектур, Onion Architecture не залежить від рівнів даних, а враховує реальні моделі доменів.

Відповідно до традиційної архітектури, взаємодія інтерфейсу користувача (UI) з бізнес-логікою і шарами даних веде до змішування всіх шарів та їх сильної взаємозалежності. У 3-рівневій та n-рівневій архітектурі жоден із шарів не функціонує незалежно, що породжує проблеми розмежування. Такі системи складно розуміти та утримувати через інтенсивні зв'язки між шарами. Головним недоліком цієї традиційної архітектури є надмірна взаємозалежність.

Архітектура Onion розглядає програму як розподілену лише на рівні.

Центральним елементом є один незалежний рівень, що визначає взаємодію з другим рівнем, другий – з третім тощо. Іншими словами, перший незалежний рівень оточений другим, який залежить від першого. За цим слідує, що навколо першого незалежного рівня нашаровується другий, залежний від першого. А подібно утворюється інше нашарування навколо другого рівня, залежного від обох попередніх.

Образно це може бути виражено у вигляді цибулини, в якому також є серцевина, навколо якого нашаровуються всі інші верстви, аж до лушпиння (рис. 3.1).

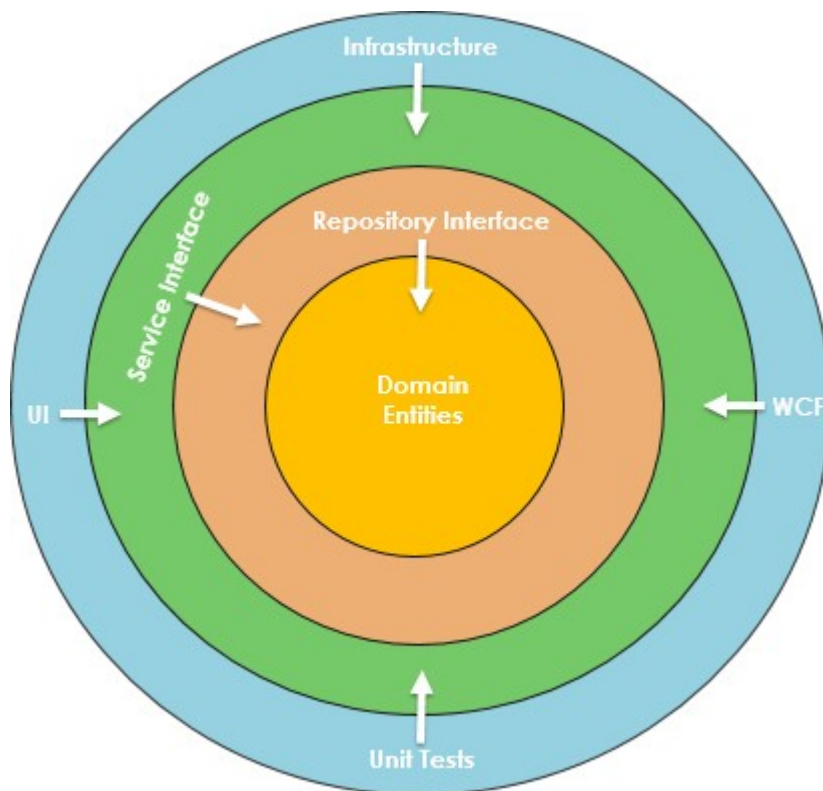


Рисунок 3.1. Onion-архітектура

Архітектура Onion успішно подолала ці проблеми, визначаючи верстви від ядра до інфраструктури. Застосовується основне правило, яке спрямовує всі зв'язки до центрального елемента. Ця архітектура чітко орієнтована на об'єктно-орієнтоване програмування, надаючи пріоритету об'єктам перед іншими. У центрі архітектури цибулі розташовується

доменна модель, що представляє бізнес і об'єкти поведінки. Навколо доменного шару є інші шари з великим набором поведінкових можливостей.

У центрі архітектури Onion знаходиться шар домену, що є об'єктами бізнесу та їх поведінкою. Основна ідея полягає в тому, щоб усі об'єкти вашого домену знаходились у цьому ядрі. В рамках шару домену також можлива наявність інтерфейсів. Важливо, що об'єкти домену немає зовнішніх залежностей, і вони рівноправні один перед одним, без необхідності у важкому коді чи зовнішніх залежностях.

Шар сховища створює абстракцію між учасниками домену та бізнес-логікою програми. У цьому шарі зазвичай включаються інтерфейси, які забезпечують збереження та отримання об'єктів, переважно через використання бази даних. Цей компонент складається з шаблону доступу до даних, який представляє собою більш гнучкий підхід до отримання даних. Також розробляється загальний репозиторій, а також додаються запити для отримання даних з джерела, порівнюючи дані з джерела інформації з сутністю господарювання та зберігаючи внесені зміни в бізнес-об'єкт у джерелі даних.

Рівень обслуговування містить інтерфейси з основними операціями, такими як додавання, збереження, редагування та видалення. Крім того, цей шар використовується для взаємодії між інтерфейсним та сховищевим компонентами. Рівень обслуговування також може включати в себе бізнес-логіку для сутностей. У цьому шарі інтерфейси обслуговування зберігаються окремо від їхньої реалізації, забезпечуючи свободу зв'язку та відокремлення проблем.

Шар інтерфейсу користувача є найзовнішнішим шаром і відповідає за обробку периферійних завдань, таких як реалізація інтерфейсу та проведення тестів. У випадку веб-застосунку він може представляти собою проект веб-API або тестовий проект. Цей шар впроваджує принцип ін'єкції залежностей,

що дозволяє побудувати додаток з вільно зв'язаною структурою та передавати його до внутрішнього рівня через інтерфейси.

Керівництво з архітектури Onion не встановлює конкретних вимог щодо реалізації шарів. Архітектор вільно вибирає спосіб втілення та має можливість визначити рівні класів, пакетів, модулів або будь-яких інших компонентів, які необхідні для додавання до рішення.

Ось деякі переваги реалізації архітектури "Onion":

Позитивні аспекти:

- Пластини цибулі архітектури взаємодіють через інтерфейси.
- Ін'єкція здійснюється під час роботи.
- Архітектура додатків конструюється поверх моделі домену.
- Всі зовнішні залежності, такі як доступ до бази даних та виклики сервісів, представлені у зовнішніх шарах.
- Відсутність залежностей внутрішнього шару від зовнішніх шарів.
- Гнучка та стійка портативна архітектура.
- Немає потреби в створенні загальних або спільних проектів.
- Швидка перевірка можлива, оскільки ядро програми не має залежностей.

Також слід врахувати кілька недоліків архітектури "Onion":

- Для початківців не завжди легко зрозуміти.
- Розподіл обов'язків між шарами часто викликає плутанину серед архітекторів.
- Значна використаність інтерфейсів.

Архітектура цибулі широко використовується в промисловості і представляє собою вискоєфективний підхід, що тісно пов'язаний з двома іншими архітектурними стилями — Шарованою та Гексагональною. Зауважимо, що ця архітектура викликає більший інтерес серед програмістів мови програмування C# у порівнянні з представниками мови програмування Java. [24].

### 3.2. Опис системи

Система являє собою програмний засіб для автоматизації та комп'ютеризації роботи сімейного лікаря та побічних операцій, таких як: створення та перевірка розкладу прийомів та заходів працівника, зберігання та перевірка історій хвороби, облік медикаментів та обладнання, фіксація фінансових операцій відповідно до статті доходу, збір та зберігання даних про пацієнтів та їх обстежень, структуризація підрозділів та ведення обліку колективу працівників, керування доступом на основі ролей, резервне копіювання.

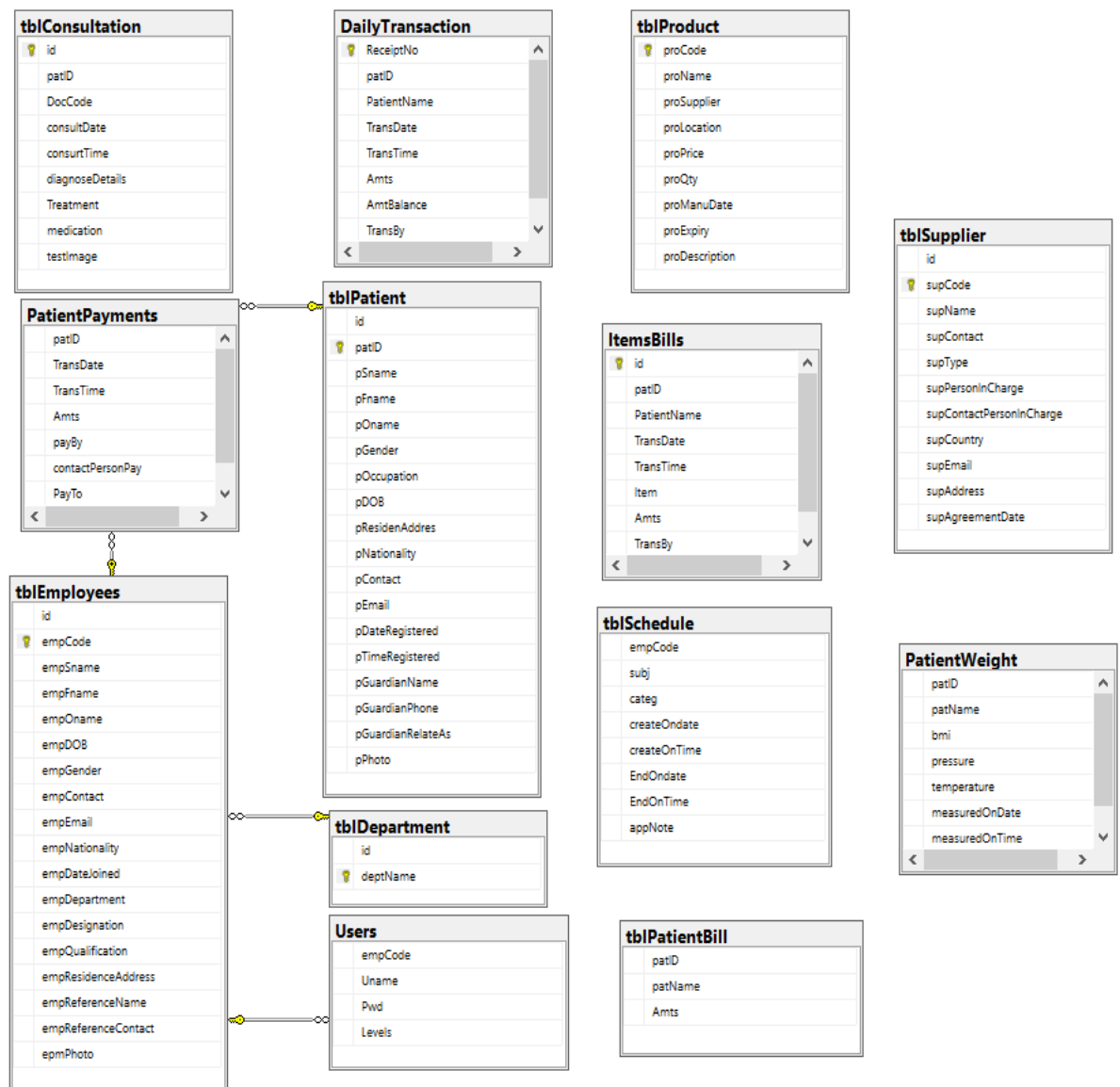
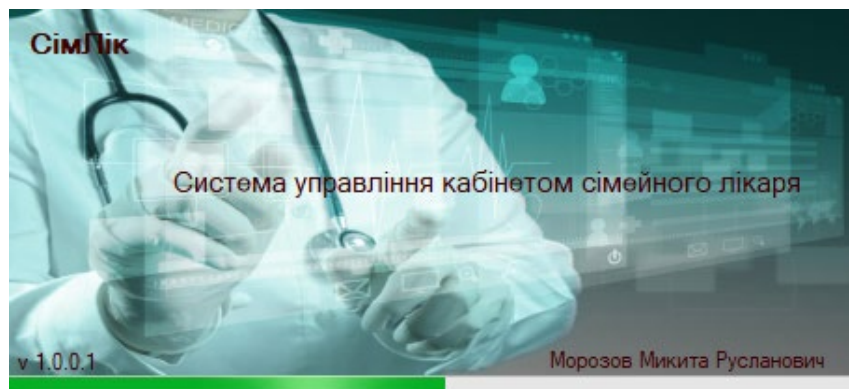


Рис. 3.2. Діаграма схеми бази даних

База даних складається з 13 основних (рис. 3.2) та 5 допоміжних таблиць:

- tblConsultation – таблиця консультацій хворих с деталізацією;
- DailyTransaction – таблиця фінансових операцій;
- tblProduct – таблиця товарів та майна;
- PatientPayments – таблиця оплат пацієнтів;
- tblPatient – таблиця пацієнтів;
- ItemsBills – деталізована таблиця фінансових операцій;
- tblSupplier – таблиця постачальників;
- tblEmployees – таблиця співробітників;
- tblDepartment – таблиця відділів;
- tblSchedule – таблиця запланованих заходів;
- PatientWeight – таблиця діагностичних даних пацєнтів;
- Users – таблиця користувачів;
- tblPatientBill – таблиця балансу.

Запуск системи ілюструється екраном завантаження після паузи надається форма авторизації до системи управління кабінетом сімейного лікаря. Система має базовий поділ користувачів на основі ролей.



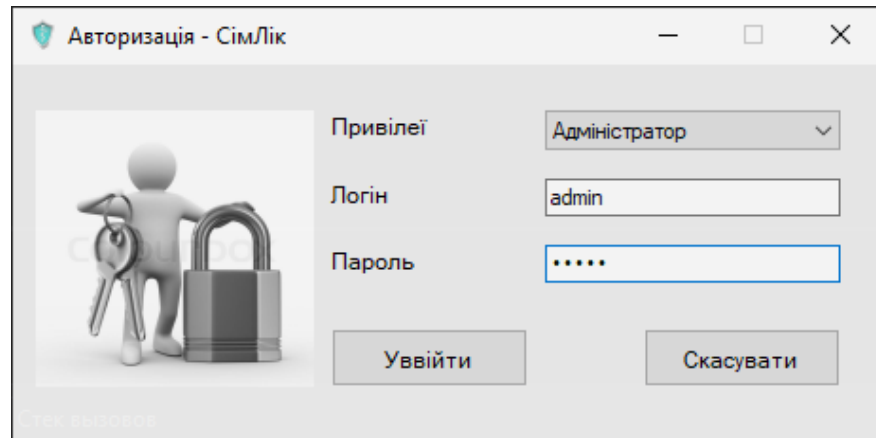


Рис 3.3. Екран завантаження та форма авторизації

Після авторизації користувач потрапляє на головну форму додатку (рис 3.4), керуюча панель з випадаючими меню, що відрізняються в залежності від рівню доступу. Знизу представлена інформативна панель яка показує поточного користувача, дату та час.

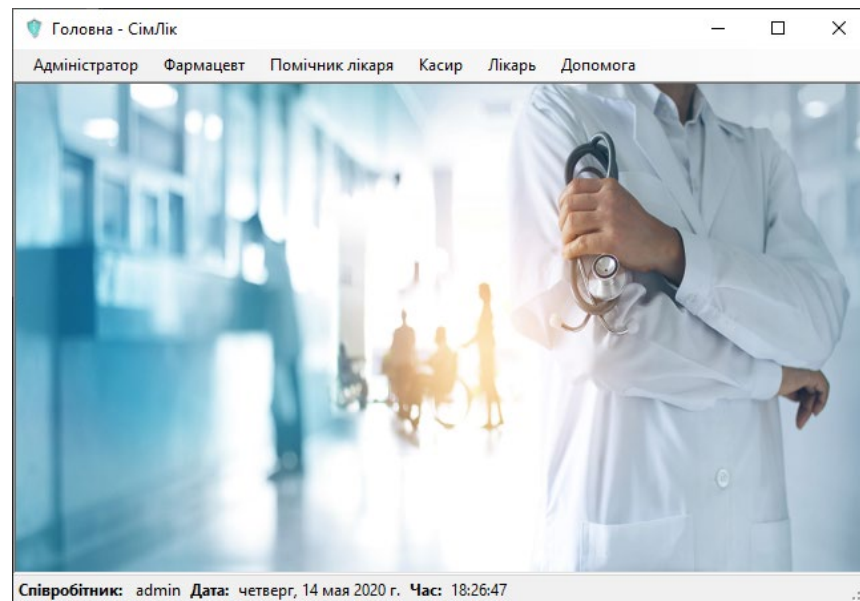


Рис. 3.4. Головне меню

Для зручності структуризації роботи створено 5 типів користувацьких ролей, у яких активні відповідні пункти меню на головному екрані додатку:

- Адміністратор

Має доступ до всіх операцій та форм з додатковим меню



адміністратора додатку.

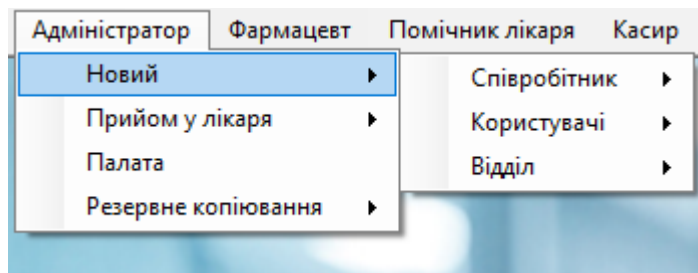


Рис. 3.5. Випадаюче меню адміністратора

– Касир

Має доступ до фінансових операцій.

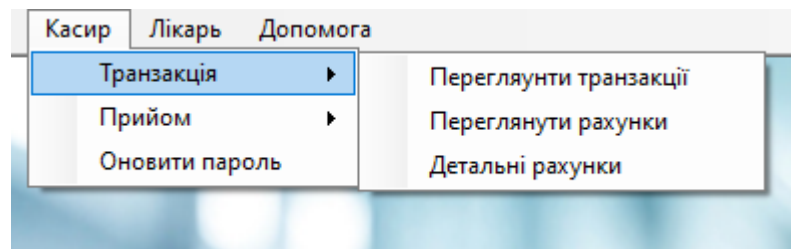


Рис. 3.6. Випадаюче меню касира

– Лікарь

Проведення консультацій та перегляд базових даних.

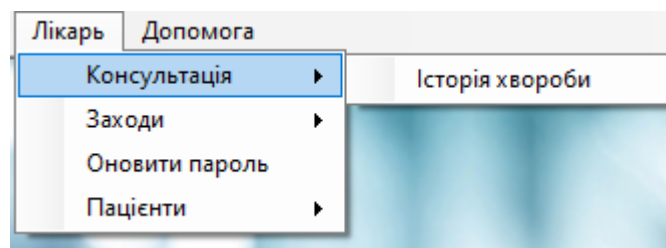


Рис. 3.7. Випадаюче меню лікаря

– Помічник лікаря

Основна функція – створення картки пацієнта та проведення базової діагностики.

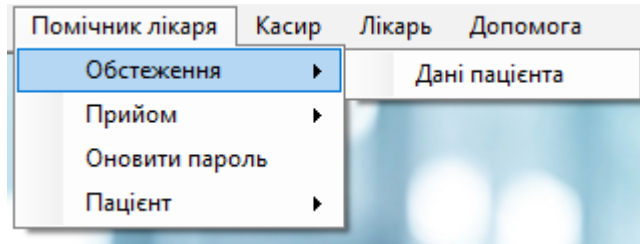


Рис. 3.8. Випадаюче меню помічника лікаря

– Фармацевт

Завідуючий процесами обліку медикаментів та інвентару.

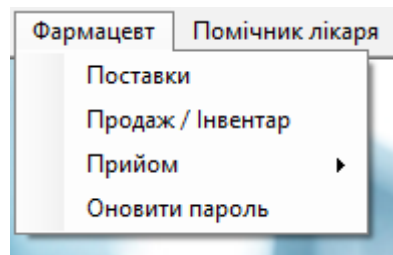


Рис. 3.9. Випадаюче меню фармацевта

Частина система, яка використовується усіма рівнями ролей, яка дозволяє записувати та фіксувати час роботи співробітників, що характеризується такими видами діяльності: прийом, консультація, симпозіум, збори, платіж, телефонний дзвінок, презентація, поставка.

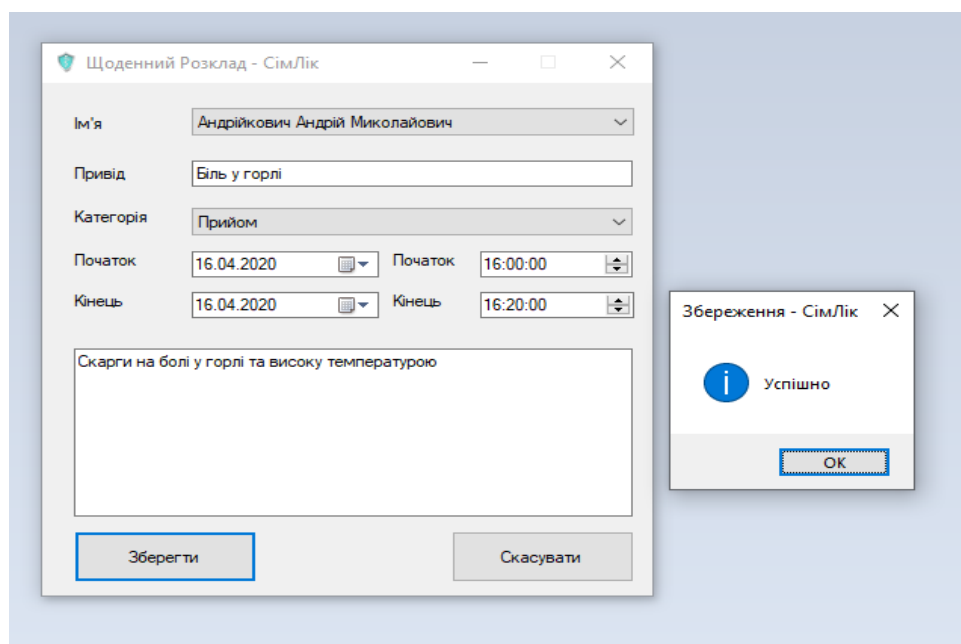


Рис. 3.10. Форма створення розкладу

Якщо за розкладом необхідно провести медичну консультацію, то

для успішного проведення її у додатку необхідно мати введенними у систему картки пацієнтів. Для взаємодії з підсистемою були розроблені форми реєстрації (рис. 3.11), списку та перегляду (рис. 3.12). До даних якими оперує система належить базова інформація про пацієнта (повне ім'я, дата народження, стать, громадянство, місце роботи, адреса та контакти) та інформація про попечителя(піклувальника), який може асистувати пацієнту та до якого можна звернутися у випадку нестабільності та ургентного стану пацієнта.

Форма реєстрації пацієнта

ID: Pat-8

Особисті подробиці

Ім'я:  Прізвище:

По батькові:  Стать: Чоловіча

Д. / н.: 14.05.2020 Вік: 0

Місце роботи:  Національність: Україна

Телефон:  Адреса:

Email:

Попечитель

Ім'я:

Фамілія:

Телефон:

Ідентифікація:

Зберегти Скасувати

Рис. 3.11. Форма реєстрації пацієнта

Перегляд пацієнта - СімЛік

Ім'я пацієнта: Собуренко Петр Юрійович Посада: Студент

Телефон: 0205585817 Країна: Україна

Адреса: Київ Піклувальник: Собуренко Миколай

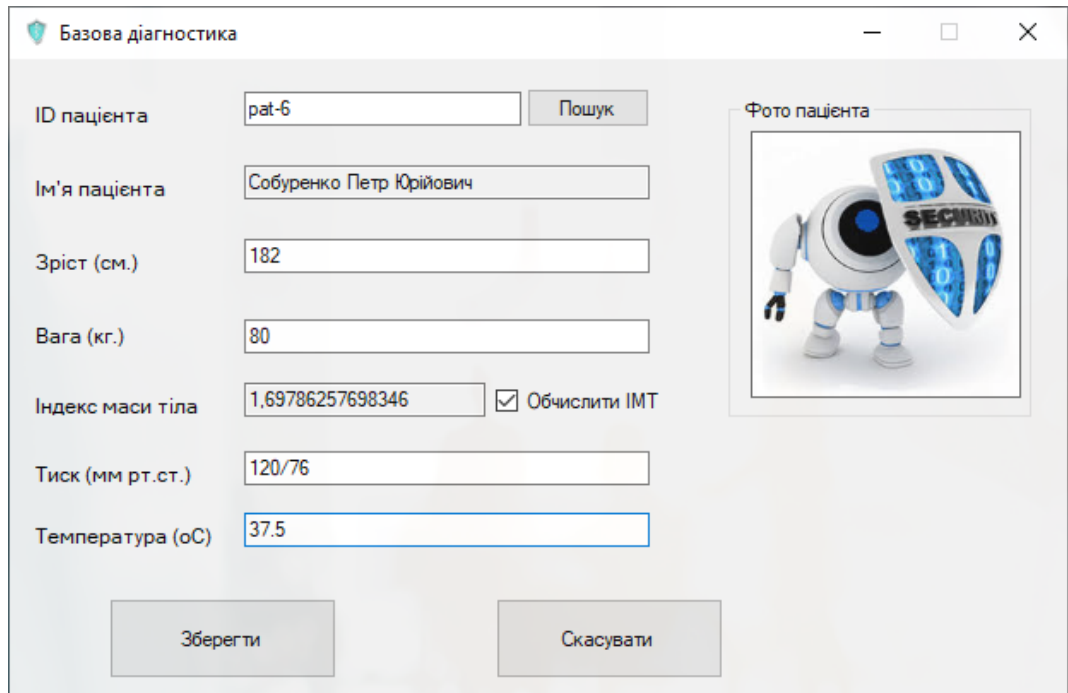
Email: example@gmail.com Телефон п.: 0205585817

ID	Прізвище	Ім'я	По батькові	Стать	Посада	Дата народження	Адреса
Pat-5	Сидоренко	Федор	Володимирович	Чоловіча	Викладач	24.11.1976	Київ
Pat-6	Собуренко	Петр	Юрійович	Чоловіча	Студент	15.06.2000	Київ
Pat-7	Брежнева	Наталія	Романівна	Жіноча	Актриса	24.01.1986	Москва

Закрити

Рис. 3.12. Форма перегляду інформації про пацієнтів

Базова діагностика проводиться через форми зображені на рис. 3.13. Вікно ілюструє збір антропометричних даних, показники температури і тиску. Для покращеної навігації використовується фото пацієнта, що має бути попередньо завантаженим, та пошук по ID картки пацієнта. Для перегляду проведених діагностик можна звернутися до форми “Перегляд діагностичних даних пацієнта” (рис. 3.14).



Базова діагностика

ID пацієнта:

Ім'я пацієнта:

Зріст (см.):

Вага (кг.):

Індекс маси тіла:  ☒ Обчислити ІМТ

Тиск (мм рт.ст.):

Температура (оC):

Фото пацієнта




Рис. 3.13. Форма вводу діагностичних даних пацієнтів

Перегляд діагностичних даних пацієнта

Фільтр пошуку  
Ім'я пацієнта: Сидоренко Федор Володимирови

	ID	Ім'я	ІМТ	Тиск	Температура	Дата	Час
▶	pat-5	Сидоренко Фед...	2152,04	120/200	40	22.11.2020	14:22
	pat-5	Сидоренко Фед...	5711,88	100/150	35	23.11.2020	14:28
	pat-5	Сидоренко Фед...	3802,06	20	36	24.11.2020	14:31
	pat-6	Собуренко Петр...	976,39	23	45	15.01.2020	9:06
*							

Оновити Закрити

Рис. 3.14. Форма перегляду діагностичних даних пацієнтів

Для безпосередньої консультації пацієнтів створена спеціальна форма, в якій, обрав попередньо створеного пацієнта, можливо зафіксувати усі деталі прийому: діагностичні дані, процедуру лікування, рецепт, час консультації і лікування, а також можливість завантажити лабораторні дослідження у форматі скан-копії.

Консультація - СімЛік

Лікар: admin

Календар: май 2020 г.

ID пацієнта: Pat-6 Ім'я пацієнта: Собуренко Петр Юрійович

Діагностика: Підозра на остеохондроз

Лікування: Протизапальні, спазмолітичні та препарати, що викликають стимуляцію мікроциркуляції крові. Через 4 дні на діагностику до вузького спеціаліста

Інформація про ліки (препарати, дозування, кількість): Пантогаматоген, дроговерин, трентал. Кожень день по одній таблетці 4

Інтервал часу: Дні

Завантажити скан-копію

Зберегти Закрити

[Імпортувати лаб. дослідження](#)

Рис. 3.15. Форма консультацій хворих с деталізацією

При подальшому спостереженні необхідно своєчасно і грамотно, базуючись на історіях хвороби, призначати лікування. Щоб це забезпечити необхідно мати доступ до цієї інформації, що надається формою зображеною на рис. 3.16.

**Історія лікування пацієнта - СімЛік**

**Діагноз**  
У пацієнта ГРВІ. Запалені дихальні шляхи з високою температурою тіла, ознобом, а також втратаю апетиту. Слабкість та біль у м'язах.

**Препарати**  
По одній таблетці на день між прийомами їжі. Потрібно прихити до лікарні через три дні для огляду.

**Лікування**  
Тилорон, Ібупрофен, інгаляції з бронхосекретолітиками, Лібексин

**Результати досліджень**

ID	Лікар	Дата	Час	Діагноз	Лікування	Медикаменти
Pat-5	Макаров Іван В...	14.01.2020	11:39	Діагноз	Лікування	Препарати та ку...
Pat-5	Макаров Іван В...	14.01.2020	11:52	Герпесвірус	Аміксин	По одній таблет...
Pat-5	Макаров Іван В...	14.01.2020	12:06	H1N1	Гропрінозін	По дві таблетці ...
Pat-5	Макаров Іван В...	14.01.2020	12:06	Бронхіт	Флокссум, сель...	По одній таблет...
Pat-5	Андрійкович Ан...	14.01.2020	10:28	У пацієнта ГРВІ...	Тилорон, Ібупро...	По одній таблет...
Pat-7	Андрійкович Ан...	15.01.2020	12:40	Підозра на запа...	Знеболочі, пода...	Знеболочі зас...

Закрити

Рис. 3.16. Форма історій хвороби пацієнтів

При необхідності співробітник кожного рівня доступу має можливість звертатися з розкладом не тільки записів до лікаря, але й заходів на рівні медичного закладу тощо (рис 3.17).

Перегляд прийомів - СімЛік

	Працівник	Привід	Тип	Дата початку	Час початку	Дата кінця	Час кінця	Опис
▶	Джавара Мішель	Мітинг з розробн...	Зустріч	01.01.2020	12:31	01.01.2020	15:31	Складання
	Джавара Мішель	Обговорення си...	Презентація	01.01.2020	12:12	01.01.2020	15:12	Вступ до ви...
	Макаров Іван В...	Прийом хворих	Прийом	18.01.2020	12:50	18.01.2020	13:15	Запис з дер...
	Макаров Іван В...	Прийом хворих	Прийом	18.01.2020	14:40	18.01.2020	15:00	-
	Андрійкович Ан...	Прийом хворих	Прийом	25.01.2020	16:35	25.01.2020	16:50	Залізняк з г...
	Андрійкович Ан...	Прийом хворих	Прийом	16.04.2020	16:00	16.04.2020	16:20	Скарги на б...

Закрити

Рис. 3.17. Форма перегляду існуючих прийомів та заходів

Система надає можливість вести облік медичної продукції та обладнання за допомогою спеціальної підсистеми. Призначені для взаємодії з підсистемою форми (рис. 3.18 – рис. 3.20) дозволяють редагувати дані з безпосереднім оприбуткуванням та реалізацією продукції. У формі продажу є необхідність обрання попередньо створеного пацієнта завдяки чому можна звернутися до виписаного лікарем рецептом.

Товар

Додавання товару | Редагування | Перегляд складу | Продаж

Редагування

Номер

Найменування

Постачальник

Ціна

Кількість

Інформація

Дата виробництва

Місцезнаходження

Термін придатності

Оновити

Закрити

Товар

Додавання товару | Редагування | Перегляд складу | Продаж

Додати

Найменування

Постачальник

Ціна

Кількість

Інформація

Дата виробництва

Місцезнаходження

Термін придатності

Додати

Закрити

Рис. 3.18. Форми редагування та додавання медикаментів та обладнання

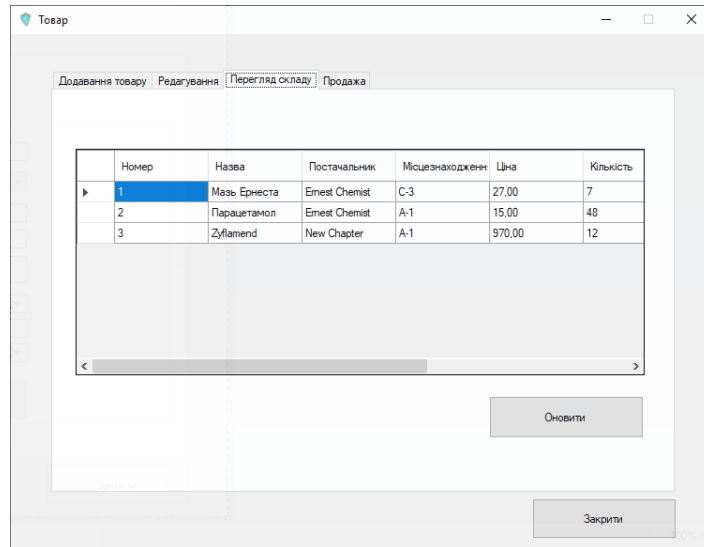


Рис. 3.19. Форма перегляду існуючих медикаментів та обладнання

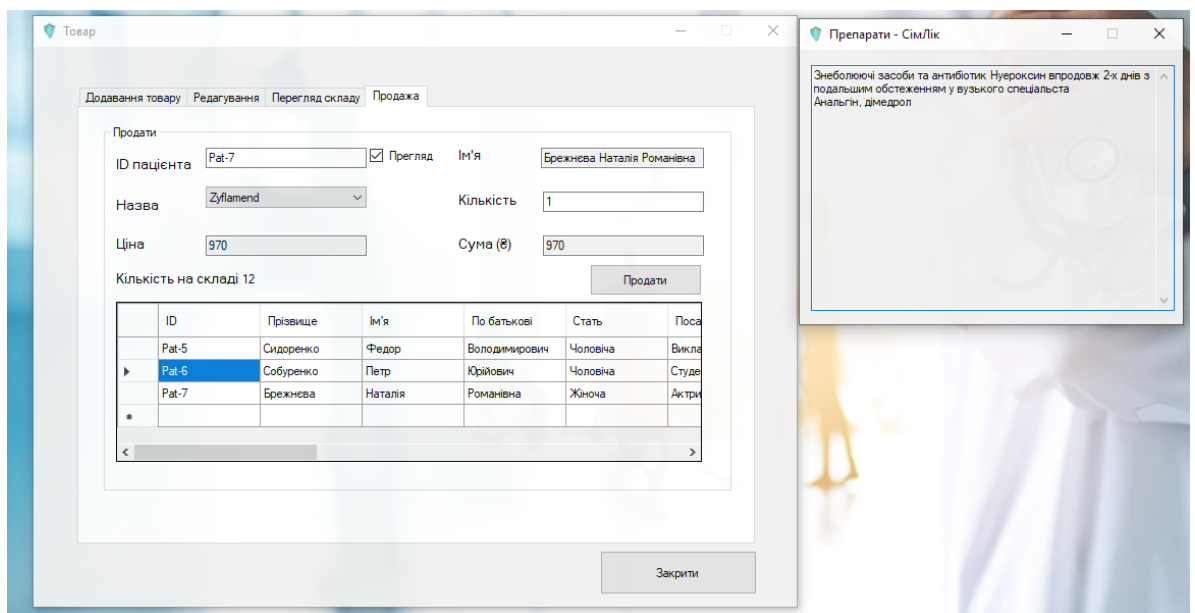


Рис. 3.20. Форма реєстрації продаж медикаментів та обладнання з впливаючим вікном рецепту лікаря

За кожною продукцією має бути закріплений постачальник. Перегляд та редакція бази постачальників виконується за формами наведеними на рис. 3.20. Через них надається можливість указувати назву підприємства, контактні номери керівництва, вказати тип продукції що поставляється, відповідальна особа (відділ продаж) та її контактні дані з боку



постачальника, а також адреса та дата укладання угоди про поставки.

The image shows two windows from a web application titled 'Постачальник' (Supplier).

The left window, 'Створення картки постачальника' (Create supplier card), has a tab 'Додати постачальника' (Add supplier) selected. It contains a form with the following fields:
 

- ID постачальника:
- Ім'я постачальника:
- Контакт постачальника:
- Тип поставок:
- Відповідальна особа:
- Контактні дані:
- Країна:
- Email:
- Адреса постачальника:
- Дата угоди:

 A 'Додати' (Add) button is at the bottom right.

The right window, 'Переглянути постачальників' (View suppliers), has a search bar 'Введіть ID:' and a 'Пошук' (Search) button. Below it is a table of suppliers:

ID	Назва	Контакт	Тип поставок	Продавець	Телефон продавця
вир-1	Emest Chemist	0549123432	Медикаменти	Joey Bee	05491234
вир-2	Київський вітам...	0444972112	Медикаменти	Борис Володим...	04449721
вир-4	M.BIOTECH LIMI...	0202039939	Медикаменти	Jonas Key	02020399
вир-5	Stahlmed Laborat...	0928282882	Обладнання	Gratson Bens	09282828
вир-7	Борщівський ...	0509939399	Медикаменти	Сергій Олексан...	05099393
вир-8	Біофарма	0444972148	Медикаменти	Олександр Сепр...	04449721

Below the table is a pagination bar showing 'Загалом: 6' (Total: 6) and an 'Оновити' (Refresh) button.

Рис. 3.21. Форми вводу та перегляду постачальників

Серед сервісних функцій були введені операція та форма для зміни паролю, а також функція резервного копіювання бази даних з користувацького інтерфейсу додатку. Резервне копіювання зберігає базу даних у форматі файлу резервних копій “\*.BAK”.

The image shows a window titled 'Оновлення паролю' (Change password). It contains the following fields:
 

- ID співробітника:
- Логін:
- Поточний пароль:
- Новий пароль:
- Повтор нового паролю:

 An 'Оновити' (Update) button is located at the bottom right.

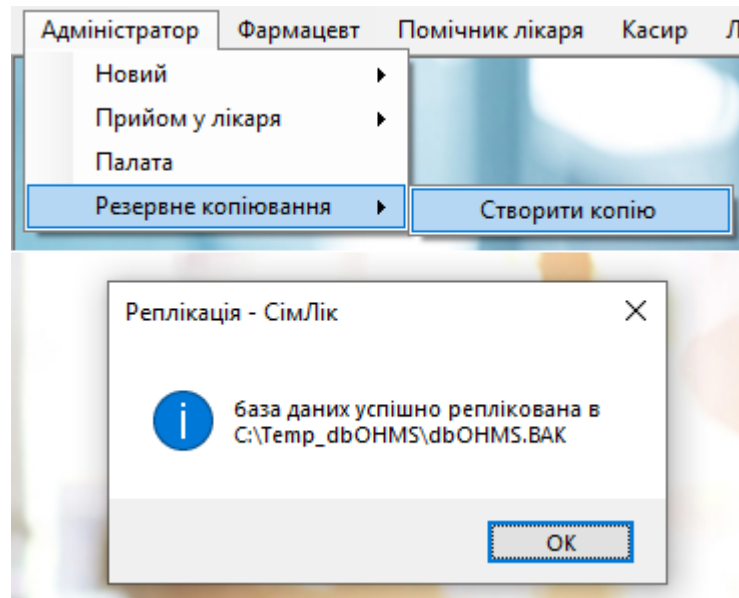


Рис. 3.22. Ілюстрація роботи сервісних функцій

Для зв'язку з розробником або адміністратором передбачено інформаційне вікно (рис. 3.23).

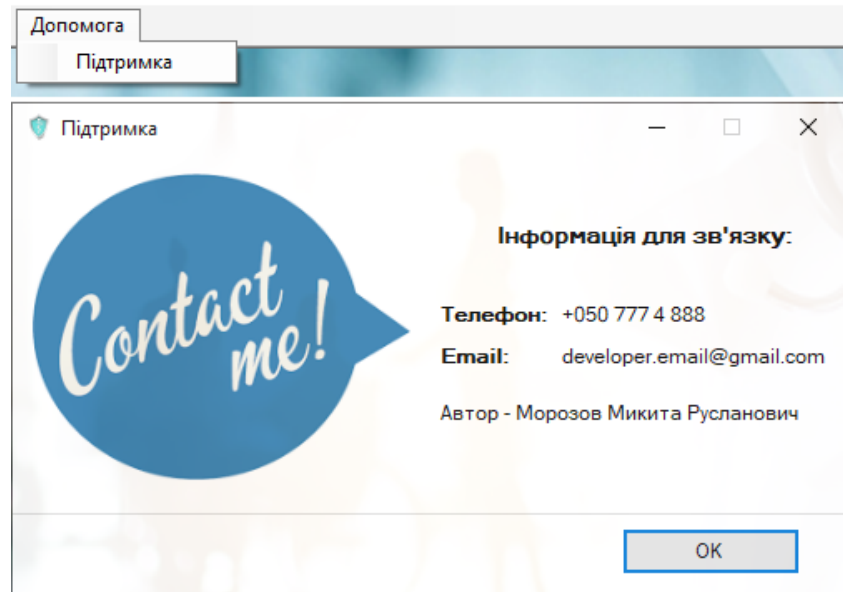


Рис. 3.23. Інформаційне вікно підтримки

### 3.3. Висновки до розділу 3

В цьому розділі дипломної роботи описано створення інформаційної моделі системи управління кабінетом сімейного лікаря “СімЛік”. Ця модель складається з бази даних, сервера бази даних та клієнтської частини.

Виконано програмну реалізацію проекту засобами мови програмування C#, технологією побудови графічного інтерфейсу WindowsForm і бази даних: Microsoft SQL Server (MSSQL).

Програмний засіб має регламент керування доступом на основі наступних ролей:

- Адміністратор;
- Касир;
- Лікарь;
- Помічник лікаря;
- Фармацевт.

Було розглянуто складові підсистем: форми, підпрограми та змінні.

Спроековано та розроблено бази даних стосовно обліку пацієнтів та лікарів, яка складається з 13 таблиць. Розроблено програмний засіб, що забезпечує ефективне ведення обліку інформації про лікарів, клієнтів і послуги, і дозволяє автоматизувати формування стандартних та нестандартних звітів стосовно обліку.

Також розроблено інтуїтивно-зрозумілий інтерфейс програмного засобу, виконаний у вигляді вікон. Основне вікно ПЗ має основні функціональні елементи у вигляді панелі випадаючих меню. Кожне меню відображає різноманітну інформацію стосовно розкладу відвідувань, послуг, видів послуг та пацієнтів та обліку відповідно до рівню доступу користувача.

Розроблена система може бути впроваджена для будь-якої медичної установи, що передбачає прийом пацієнтів за записом. Для розподіленої роботи з програмним засобом екземпляр бази даних Microsoft SQL Server повинен бути опублікований для доступу з робочого комп'ютеру, а параметри підключення відповідно налаштовані.

В планах подальшої розробки є створення підсистеми керування лікарняною палатою, були створені функції-заглушки та допоміжні таблиці в базі даних; введення підсистеми лабораторії для аналізів для більш детальної діагностики.

## ВИСНОВКИ

У дипломній роботі розроблено систему обліку інформації про лікарів, клієнтів і лікарські послуги сімейного лікаря. Програмний комплекс складається з бази даних і програми- клієнта.

Вивчено предметну область медичного закладу, інформаційні потреби користувачів системи, на основі яких спроектовано інфологічну модель бази даних. Обґрунтовано вибір архітектури клієнт-сервер для створення інформаційної системи. Для реалізації бази даних обрана промислова СУБД Microsoft SQL Server, що є однією з найбільш поширених і надійних СУБД. Реалізовано даталогічне проектування БД.

Спроектовано базу даних для обліку відвідувань і зберігання всієї необхідної для функціонування організації і обслуговування відвідувачів інформації. Визначено обмеження на формат даних, що вводяться.

Розроблено базу даних, що призначено для зберігання інформації про:

- лікарів;
- послуги та їх видах;
- медичинські препарати та обладнання;
- постачальників препаратів та обладнання;
- відвідини процедур та історії хвороби;
- пацієнтів;
- фінансові операції.

У кваліфікаційній роботі детально описано інтерфейс програмної частини системи обліку, а так само наведено сценарії використання найбільш важливих процедур програми зі зазначенням всіх використаних компонентів.

Програма-клієнт має низку позитивних сторін: простий інтерфейс, захист від некоректного введення даних, використання програми не вимагає спеціального досвіду і тощо.

Система обліку включає в себе базу даних для зберігання інформації

про відвідувачів і програму-клієнта, за допомогою якої користувач працює з нею. Передбачається, що база даних буде зберігатися на сервері, а програма-клієнт звертається до неї для збору інформації та внесення нових даних. В основі системи лежить використання дворівневої архітектури - сервера бази даних і клієнтської частини, яка формує запити, і яка має певної функціональної незалежністю, яка дозволяє ефективно вирішувати проблеми управління даними.

Розроблена система обліку відвідувань сімейного лікаря відповідає завданням на дипломну роботу.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. MySQL. Справочник по языку. М.: *Издательский дом «Вильямс»*, 2018. 432 с.
2. Гусев, А., Романов Ф., Дуданов И. Медицинские информационные системы: анализ рынка. URL: [http://www.pcweek.ru/idea/article/detail\\_print.php?ID=75038&print=Y](http://www.pcweek.ru/idea/article/detail_print.php?ID=75038&print=Y).
3. Дабагов, А.Р. Информатизация здравоохранения и некоторые проблемы построения интегрированных медицинских информационных систем URL: <http://jre.cplire.ru/mac/sep11/2/text.html>.
4. Деордица Ю.С. Информационные системы и базы данных: Учеб. пособие. Луганск: *Изд-во ВУГУ*, 1999. 140 с.
5. Диго С.М. Проектирование и использование баз данных; Учебник. М.: *Финансы и статистика*, 1995. 208 с.: ил.
6. Закон України «Про державні фінансові гарантії медичного обслуговування населення» [Електронний ресурс]: закон від 19.10.2017. URL: <https://zakon.rada.gov.ua/laws/show/2168-19>
7. Закон України «Про підвищення доступності та якості медичного обслуговування у сільській місцевості» [Електронний ресурс]: закон від 14.11.2017. URL: <https://zakon.rada.gov.ua/laws/show/2206-19>
8. Кириллов В.В. Основы проектирования реляционных баз данных. Учеб. пособ. Санкт-Петербург: *Изд-во ГИТМО (технический университет)*, 1998. 96 с.
9. Методические рекомендации по оснащению медицинских учреждений компьютерным оборудованием и программным обеспечением для регионального уровня единой государственной информационной системы в сфере здравоохранения, а также функциональные требования к ним. URL: <http://www.minzdravsoc.ru/docs/mzsr/informatics/47>.
10. Методические рекомендации по составу и техническим

требованиям к сетевому телекоммуникационному оборудованию учреждений системы здравоохранения для регионального уровня единой государственной информационной системы в сфере здравоохранения, а также функциональные требования к ним. URL: <http://www.minzdravsoc.ru/docs/mzsr/informatics/46>.

11. Методические рекомендации по составу создаваемых в 2011 – 2012 годах в рамках реализации региональных программ модернизации здравоохранения прикладных компонентов регионального уровня единой государственной информационной системы в сфере здравоохранения, а также функциональные требования к ним. URL: <http://www.minzdravsoc.ru/docs/mzsr/informatics/45>.

12. Морозов Д.В., Лысенко К.И., Баранов Л.И. Информатизация медицины. Новый этап развития. URL : [http://zhenilo.narod.ru/main/ips/2009\\_medicine.pdf](http://zhenilo.narod.ru/main/ips/2009_medicine.pdf).

13. Наказ МОЗ України від 04.01.2018 № 13 "Про деякі питання застосування Україномовного варіанту Міжнародної класифікації первинної медичної допомоги(ICPC-2-E)". [Електронний ресурс]: Міністерство охорони здоров'я України. URL:<http://moz.gov.ua/article/ministry-mandates/nakaz-moz-ukraini-vid-04012018--13-pro-dejaki-pitannja-zastosuvannja-ukrainomovnogo-variantu-mizhnarodnoi-klasifikacii-pervinnoi-medichnoi-dopomogiicpc-2-e>

14. Наказ МОЗ України від 08.08.2014 № 549 "Про внесення змін до деяких наказів Міністерства охорони здоров'я України". [Електронний ресурс]: Міністерство охорони здоров'я України. URL: [http://old.moz.gov.ua/ua/print/dn\\_20140808\\_0549.html](http://old.moz.gov.ua/ua/print/dn_20140808_0549.html)

15. Наказ МОЗ України від 19.03.2018 № 504 "Про затвердження Порядку надання первинної медичної допомоги)". [Електронний ресурс]: Міністерство охорони здоров'я України. URL:



<http://moz.gov.ua/article/ministry-mandates/nakaz-moz-ukraini-vid-19032018--504-pro-zatverdzhennja-porjadku-nadannja-pervinnoi-medichnoi-dopomogi>

16. Об опыте разработки и внедрения медицинских информационных систем. URL: <http://www.zdrav.ru/articles/practice/detail.php?ID=77693>.

17. Організація медичної допомоги. Первинна медико-санітарна допомога, впровадження сімейної медицини. URL: [http://www.medcollege.te.ua/sayt1/Lecturs/soc\\_meducuna\\_ta\\_statustuka\\_lection/Lection\\_5.htm](http://www.medcollege.te.ua/sayt1/Lecturs/soc_meducuna_ta_statustuka_lection/Lection_5.htm).

18. Орлов, Г.М. Типовая медицинская информационная система персонифицированного учета оказания медицинской помощи [Текст]. *Врач и информационные технологии*. 2009. № 2, С. 38–43.

19. Павлов В.В., Закамсков А.В., Рахманова З.Б. Актуальные вопросы внедрения и использования медицинских информационных систем (МИС). Проблемы и ошибки при внедрении и использовании МИС. *Информационно-измерительные и управляющие системы*. 2010. Т. 8, № 12. С. 82–85.

20. Перов В. Н. Информационные системы. СПб.: Питер, 2002. 688 с.

21. Мартин Р., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C# . Издательство: Символ-Плюс; серия: High tech. 2011. 757 с.

22. Проект наказу МОЗ України "Про затвердження Концепції інформатизації сфери охорони здоров'я України". URL: [http://www.moz.gov.ua/ua/portal/dn\\_20121226\\_pp.html](http://www.moz.gov.ua/ua/portal/dn_20121226_pp.html).

23. Виссер Джуст. Разработка обслуживаемых программ на языке C# : [пер. с англ. Р. Н. Рагимова]. Москва: ДМК Пресс, 2017. 191 с

24. Сравнительная характеристика различных подходов к автоматизации деятельности стационара. URL:

<http://ucr.gs.com.ua/support/files/hcr&meds.doc>.

25. СТО МОСЗ 91500.16.0002-2004. Стандарт организации «Информационные системы в здравоохранении. Общие требования». URL: [http://www.miacso.ru/UPLOAD/fck/files/Doc\\_all/Standart.pdf](http://www.miacso.ru/UPLOAD/fck/files/Doc_all/Standart.pdf).

26. Хортон А. VisualC++ 2010: полный курс. М.: ООО «И.Д. Вильямс», 2011. 1216 с

27. Четвериков В.Н. Ревунков Г.И., Самохвалов Э.Н. Базы и банки данных: Учеб. для вузов по спец. «АСУ»/ Под ред. В.Н. Четверикова. М.: Высш. шк., 1987, 248 с.: ил.

## ДОДАТКИ

## Додаток А. Лістинг класу frmConsultation.cs.

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace SimLic
{
    public partial class frmConsultation : Form
    {
        clsSelect selectClass = new clsSelect();
        clsInsert varInsert = new clsInsert();
        DateTimePicker sysdate = new DateTimePicker();
        double consultBills = 40; //consultation bill
        double AddBill = 0;
        ErrorProvider err = new ErrorProvider();
        public string docName;

        public frmConsultation()
        {
            InitializeComponent();

            private void textBox5_TextChanged(object sender, EventArgs e)
            {
                validateMedication((Control)sender);
            }

            private void frmConsultation_Load(object sender, EventArgs e)
            {
                clearAll();
            }

            private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClicked
            EventArgs e)
            {
                selectClass.ImageUpload(pictureBox1);
            }

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

            void clearAll()
            {
                selectClass.selectIdForname(cboPatcode);
                cboFor.SelectedIndex = 0;
            }
        }
    }

```

```

        cboPatcode.SelectedIndex = 0;
        txtDiagnose.ResetText();
        txtTreatment.ResetText();
        txtMedications.ResetText();
        pictureBox1.Image = Properties.Resources.labs;
    }

    private void btnSaveResult_Click(object sender, EventArgs e)
    {
        validateDiagnosis(txtDiagnose);
        validateTreatment(txtTreatment);
        validateMedication(txtMedications);

        if (err.GetError(txtDiagnose).Length != 0)
        {
            err.SetError(txtDiagnose, "Поле не може бути порожнім");
        }
        else if (err.GetError(txtTreatment).Length != 0)
        {
            err.SetError(txtTreatment, "Поле не може бути порожнім");
        }
        else if (err.GetError(txtMedications).Length != 0)
        {
            err.SetError(txtMedications, "Поле не може бути порожнім");
        }
        else
        {
            try
            {
                UpdateBalance();
                varInsert.insertIntoConsultation(cboPatcode, txtDocName.Text,
                    t.ToString(), sysdate, sysdate, txtDiagnose.Text, txtTreatment.Text, txtMedications.Text + ", " + cboFor.SelectedItem.ToString(), pictureBox1);
                varInsert.ItemsBills(cboPatcode.SelectedItem.ToString(), txtPatName.Text, sysdate, sysdate, "Consultation", consultBills, txtDocName.Text);
                cboPatcode.SelectedIndex = 0;
                selectClass.selectIdForname(cboPatcode);
                cboFor.SelectedIndex = 0;

                txtDiagnose.ResetText();
                txtTreatment.ResetText();
                txtMedications.ResetText();
                pictureBox1.Image = Properties.Resources.labs;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}

```

```

private void cboPatcode_SelectedIndexChanged(object sender, EventArgs e)
{
    selectClass.selectname(cboPatcode.SelectedItem.ToString());
    txtPatName.Text = selectClass.fullName;
}

//Update Balance
void UpdateBalance()
{
    string updateBillString;
    SqlConnection con;
    // SqlCommand cmd;
    try
    {
        con = new SqlConnection(varInsert.dbPath);

        // ADD CONSULTATION FEE WHEN IGNORE CONSULTATION FEE IS UNCHECKED

        if (chkNocharge.Checked == false)
        {
            try
            {
                selectClass.selectname(cboPatcode);
                AddBill = consultBills + selectClass.patientBills;
                updateBillString = "update tblPatientBill set Amt = '" + AddBill.ToString() + "' where patID = '" + cboPatcode.SelectedItem.ToString() + "' And patName = '" + txtPatName.Text + "'";
                con.Open();
                SqlCommand cmd = new SqlCommand(updateBillString, con);

                cmd.ExecuteNonQuery();

                MessageBox.Show("Успішно", "Збереження - Сімлік", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
            con.Close();
        }

        //IF CHECKED
    else
    {
        try
        {
            selectClass.selectname(cboPatcode);
            AddBill = selectClass.patientBills;

```

```

        updateBillString = "update tblPatientBill set Amt = '" + A
ddBill + "' where patID = '" + cboPatcode + "'And patName= '" + txtPatName.Text + "'";

        con.Open();
        SqlCommand cmd = new SqlCommand(updateBillString, con);

        cmd.ExecuteNonQuery();

        MessageBox.Show("Успішно", "Збереження - Сімлік", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    con.Close();
}

}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}

private void chkNocharge_CheckedChanged(object sender, EventArgs e)
{
}

//diagnose
private void txtDiagnose_TextChanged(object sender, EventArgs e)
{
    validateDiagnosis((Control)sender);
}

private void txtDiagnose_Leave(object sender, EventArgs e)
{
    validateDiagnosis((Control)sender);
}

//TREATMENT
private void txtTreatment_TextChanged(object sender, EventArgs e)
{
    validateTreatment((Control)sender);
}

private void txtTreatment_Leave(object sender, EventArgs e)
{
    validateTreatment((Control)sender);
}

```

```

//VALIDATIONS
//DIAGNOSE
void validateDiagnosis(Control ctrl)
{
    if (txtDiagnose.Text.Trim() == string.Empty)
    {
        err.SetError(txtDiagnose, "Поле не може бути порожнім");
        return;
    }
    else {
        err.SetError(txtDiagnose, string.Empty);
    }
}

//TREAT
void validateTreatment(Control ctrl)
{
    if (txtTreatment.Text.Trim() == string.Empty)
    {
        err.SetError(txtTreatment, "Поле не може бути порожнім");
        return;
    }
    else
    {
        err.SetError(txtTreatment, string.Empty);
    }
}

//medication
void validateMedication(Control ctrl)
{
    if (txtMedications.Text.Trim() == string.Empty)
    {
        err.SetError(txtMedications, "Поле не може бути порожнім");
        return;
    }
    else
    {
        err.SetError(txtMedications, string.Empty);
    }
}

private void txtMedications_Leave(object sender, EventArgs e)
{
    validateMedication((Control)sender);
}

```

}  
}  
}



### Додаток Б. Лістинг класу frmParent.cs.

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SimLic
{
    public partial class frmParent : Form
    {
        //private frmLogin login;
        clsSelect selectClass = new clsSelect();
        clsUpdate theUpdates = new clsUpdate();
        public frmParent()
        {
            InitializeComponent();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            DateTime dt = new DateTime();
            dt = DateTime.Now;

            getDate.Text = dt.Date.ToLongDateString();
            toolStripStatusLabel4.Text = dt.ToLongTimeString();
        }

        private void frmParent_Load(object sender, EventArgs e)
        {
            clsSelect selectClass = new clsSelect();

            timer1.Start();
        }
        private void label2_Click(object sender, EventArgs e)
        {
        }

        private void frmParent_FormClosing(object sender, FormClosingEventArgs
e)
        {
            Application.Exit();
        }

        private void employeeToolStripMenuItem_Click(object sender, EventArgs e
)
        {
            frmEmployee employee = new frmEmployee();
            employee.Show();
        }
    }
}

```

```

private void usersToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmUsers users = new frmUsers();
    users.Show();
}

private void supplierToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmSupplier supplier = new frmSupplier();
    supplier.Show();
}

private void productSaleInventoryToolStripMenuItem_Click(object sender,
EventArgs e)
{
    frmProduct product = new frmProduct();
    product.empName = this.getEmpCodes.Text;
    product.Show();
}

private void checkupsToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmNurseTest Nursetest = new frmNurseTest();
    Nursetest.Show();
}

private void transactionToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmPayments payment = new frmPayments();
    payment.cachierName = this.getEmpCodes.Text;
    payment.Show();
}

private void appointmentToolStripMenuItem4_Click(object sender, EventArgs e)
{
}

private void appointmentToolStripMenuItem_Click(object sender, EventArgs e)
{
}

private void appointmentToolStripMenuItem1_Click(object sender, EventArgs e)
{
}

private void appointmentToolStripMenuItem2_Click(object sender, EventArgs e)

```

```

    {
        frmAppointment appointment = new frmAppointment();
        appointment.Show();
    }

    private void appointmentToolStripMenuItem3_Click(object sender, EventArgs e)
    {
        frmAppointment appointment = new frmAppointment();
        appointment.Show();
    }

    private void consultationToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmConsultation consult = new frmConsultation();
        consult.txtDocName.Text = this.getEmpCodes.Text;
        consult.Show();
    }

    private void updatePasswordToolStripMenuItem4_Click(object sender, EventArgs e)
    {
    }

    private void updatePasswordToolStripMenuItem3_Click(object sender, EventArgs e)
    {
        frmUpdatePassword uPassword = new frmUpdatePassword();
        uPassword.Show();
    }

    private void updatePasswordToolStripMenuItem2_Click(object sender, EventArgs e)
    {
        frmUpdatePassword uPassword = new frmUpdatePassword();
        uPassword.Show();
    }

    private void updatePasswordToolStripMenuItem1_Click(object sender, EventArgs e)
    {
        frmUpdatePassword uPassword = new frmUpdatePassword();
        uPassword.Show();
    }

    private void updatePasswordToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmUpdatePassword uPassword = new frmUpdatePassword();
        uPassword.Show();
    }

    private void addAppointmentToolStripMenuItem_Click(object sender, EventArgs e)
    {

```

```

        frmAppointment appointment = new frmAppointment();
        appointment.Show();
    }

    private void viewAppointmentToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmViewAppointment viewAppointment = new frmViewAppointment();
        viewAppointment.Show();
    }

    private void addAppointmentToolStripMenuItem1_Click(object sender, EventArgs e)
    {
        frmAppointment appointment = new frmAppointment();
        appointment.Show();
    }

    private void viewAppointmentToolStripMenuItem1_Click(object sender, EventArgs e)
    {
        frmViewAppointment viewAppointment = new frmViewAppointment();
        viewAppointment.Show();
    }

    private void viewAppointmentToolStripMenuItem2_Click(object sender, EventArgs e)
    {
        frmViewAppointment viewAppointment = new frmViewAppointment();
        viewAppointment.Show();
    }

    private void addAppointmentToolStripMenuItem2_Click(object sender, EventArgs e)
    {
        frmAppointment appointment = new frmAppointment();
        appointment.Show();
    }

    private void wardToolStripMenuItem_Click(object sender, EventArgs e)
    {
        /*
        *
        * THIS FORM IS NOT YET WORKED ON
        *
        * */

        //frmWard ward = new frmWard();
        //ward.Show();
    }

    private void addPatientToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmPatient patient = new frmPatient();
        patient.nurseName = this.getEmpCodes.Text;
        patient.Show();
    }

```

```

    }

    private void viewPatientToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmViewPatient patientView = new frmViewPatient();
        patientView.Show();
    }

    private void viewPatientWeightToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmViewPatientWeight viewWeight = new frmViewPatientWeight();
        viewWeight.Show();
    }

    private void updatePasswordsToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmUpdatePassword uPassword = new frmUpdatePassword();
        uPassword.Show();
    }

    private void viewUsersToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmViewUsers userView = new frmViewUsers();
        userView.Show();
    }

    private void departmentToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Form1 department = new Form1();
        department.Show();
    }

    private void viewTransactionToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmviewDailyTrans viewTrans = new frmviewDailyTrans();
        viewTrans.Show();
    }

    private void viewDepartmentToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmViewDept viewDept = new frmViewDept();
        viewDept.Show();
    }

    private void viewBillsToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmPatientBills bills = new frmPatientBills();
        bills.Show();
    }

```

```

private void supportToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmHelp help = new frmHelp();
    help.Show();
}

private void updatePatientToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmUpdatePatient patUpdate = new frmUpdatePatient();
    patUpdate.Show();
}

private void updateEmployeeToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmupdateEmployee upEmployee = new frmupdateEmployee();
    upEmployee.Show();
}

private void viewEmployeeToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmViewEmployee viewEmp = new frmViewEmployee();
    viewEmp.Show();
}

private void nowToolStripMenuItem_Click(object sender, EventArgs e)
{
    theUpdates.BackUp();
}

private void patientHistoryToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmDocPrescription prescrib = new frmDocPrescription();
    prescrib.Show();
}

private void viewPatientToolStripMenuItem1_Click(object sender, EventArgs e)
{
    frmViewPatient viewPatient = new frmViewPatient();
    viewPatient.Show();
}

private void viewPatientWeightToolStripMenuItem1_Click(object sender, EventArgs e)
{
    frmViewPatientWeight viewWeight = new frmViewPatientWeight();
    viewWeight.Show();
}

private void viewAppointmentToolStripMenuItem3_Click(object sender, EventArgs e)

```

```

    {
        frmViewAppointment vAppointment = new frmViewAppointment();
        vAppointment.Show();
    }

    private void viewItemBillingsToolStripMenuItem_Click(object sender, EventArgs e)
    {
        frmViewBills bills = new frmViewBills();
        bills.Show();
    }

    private void viewAppointmentToolStripMenuItem4_Click(object sender, EventArgs e)
    {
        frmViewAppointment viewAppoint = new frmViewAppointment();
        viewAppoint.Show();
    }

    private void administratorsToolStripMenuItem_Click(object sender, EventArgs e)
    {
    }

    private void viewPatientWeightToolStripMenuItem2_Click(object sender, EventArgs e)
    {
        frmViewPatientWeight viewWeight = new frmViewPatientWeight();
        viewWeight.Show();
    }
}

```