**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Хмельницький політехнічний фаховий коледж**

**Національного університету «Львівська політехніка»**

**Відділення програмної інженерії**

Циклова комісія

програмного забезпечення

**КУРСОВИЙ ПРОЄКТ**

**з дисципліни «Об’єктно-орієнтоване програмування»**

**на тему:**

**Облік ковід-хворих лікарем інфекціоністом**

Виконав студент ***ІІІ*** курсу, групи ***ПІ192***

спеціальності ***121 «Інженерія програмного забезпечення»***

***Рибак Роман Васильович***

Керівник  ***Григоровський Є.С.***

Члени комісії

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_

(підпис) (прізвище та ініціали)

Оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата захисту

Хмельницький 2022

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Хмельницький політехнічний фаховий коледж**

**Національного університету «Львівська політехніка»**

**Відділення** *програмної інженерії*

**Циклова комісія** *програмного забезпечення*

**Спеціальність***121 «Інженерія програмного забезпечення»*

##### Затверджую

Голова циклової комісії

\_\_\_\_\_\_\_\_\_\_ М.В. Миколюк

« 25 » січня 2022 р.

##### Індивідуальне завдання

**на курсовий проект з дисципліни**

**«Об’єктно-орієнтоване програмування»**

видане студенту **III** курсу групи **ПІ192 Рибак Роману Васильовичу**

**Тема проекту – *Облік ковід-хворих лікарем-інфекціоністом***

Керівник курсового проекту ***Григоровський Є.С.***

Затверджено на засіданні циклової комісії програмного забезпечення, протокол №7 від 25.01. 2022р.

**Постановка задачі**

Використовуючи засоби ООП

Реалізувати: Створити програму для оптимізації роботи лікаря та автоматизації створення історій хвороб. Забезпечити додавання лікарів в систему та хворих. Обмежити доступ до хворих лише відповідним лікарям. Створювати мінімальну статистику на основі збережених даних. Створити адміна(аналог завідувача відділення), який матиме доступ до історій усіх хворих.

**Зміст пояснювальної записки**

**Відомість проєкту**

**Реферат**

**Зміст**

**Вступ**

1. Загальний розділ

* 1. Постановка завдання
  2. Опис вхідної інформації
  3. Опис результуючої інформації
  4. Стадії та етапи розробки задачі
  5. Опис існуючих методів та рішень

2. Розробка технічного та робочого проєкту

* 1. Опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних
  2. Розробка структурної схеми проєкту та об’єктної моделі задачі на основі UML діаграм
  3. Об’єктно-орієнтовне проєктування, програмування та тестування

3. Спеціальний розділ

* 1. Інструкція з інсталяції розробленого проєкту
  2. Інструкція з експлуатації проєкту

**Висновки**

**Література**

**Додатки**

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Назва етапу проєктування** | **Термін** | **Примітка** |
|  | Отримання теми курсового проєкту | 25.02 |  |
|  | Аналіз теми та умови задачі. Вступ. Аналітичний розділ. | 01.03 |  |
|  | Побудова інформаційно-математичної моделі. Побудова постановки задачі (технічного завдання);   * + найменування та область застосування;   + призначення розробки;   + вимоги до програмного забезпечення;   + вимоги до програмної документації;   + стадії та етапи розробки. | 05.03 |  |
|  | Визначення структур даних: опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних. | 09.03 |  |
|  | Розробка структурної схеми проєкту та об’єктної моделі задачі а також блок-схем алгоритмів методів:   * + зовнішнє проєктування програми;   + проєктування логіки програми;   + побудова діаграм мовою UML.   + динамічна та функціональна моделі задачі (підзадачі) | 12.03  16.03  22.03  30.03 |  |
|  | Проєктування інтерфейсу та модулів проєкту:   * + головне вікно та допоміжні вікна (головне меню, панель інструментів, контекстне меню, статусний рядок, підказка); | 06.04 |  |
|  | Налагодження та тестування програми. Аналіз результатів (підготовка тестових наборів та результати тестування) | 13.04  20.04 |  |
|  | ПОПЕРЕДНІЙ ЗАХИСТ | 26.04 |  |
|  | Інструкція з експлуатації програмного проєкту:   * + інсталяція програмного забезпечення;   + вимоги до технічного та програмного забезпечення;   + експлуатації програмного продукту. | 29.03  30.04  03.05 |  |
|  | Оформлення ПЗ | 04.05 |  |
|  | ЗДАЧА НА ПЕРЕВІРКУ ПЗ та програмної частини курсового проєкту | 06.05 |  |
|  | Захист (згідно графіка) | 10.05 |  |

**Видано** «25» «січня» *2022* року

**Термін виконання** «06» «травня» *2022* року

Індивідуальне завдання одержав Рибак Роман Васильович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ «\_\_\_\_» «» *2022* року

(підпис студента) (дата)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Григоровський Є.С.)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№ рядка** | **Формат** | **Позначення** | **Найменування** | **Кіл.**  **арк.** | **№ екз** | **Примітка** |
| **1** | **А4** |  | **Завдання на**  **проект** | **2** | **1** |  |
|  |  |  |  |  |  |  |
| **2** | **А4** | КП.ПО.02.ПІ.192.36.ПЗ | **Облік ковід-хворих лікарем інфікціоністом** | **33** | **1** |  |
|  |  |  |  |  |  |  |
| **3** | **А4** | КП.ПО.02.ПІ.192.36.Д1 | **Додаток А**  **Програмний код** | **58** | **1** |  |
|  |  |  |  |  |  |  |
| **4** | **Архів** |  | **Рибак\_ПІ-192\_КП.zip** |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Облік ковід-хворих лікарем-інфікціоністом**

Відомість проекту

*Змін.*

*Аркуш*

*№ докум.*

*Підпис*

*Дата*

*Розробив*

*Перевірив*

*Затвер.*

*Н.контр.*

*Літ*

*Аркуш*

*Аркушів*

**ХПК**

КП.ПО.02.ПІ.192.36.ВП

1

1

Н

Григоровський Є.С.

Рибак Р.В.

**РЕФЕРАТ**

Даний курсовий проєкт з розробки програмного забезпечення - “Облік ковід-хворих лікарем-інфікціоністом”, було розроблено із застосуванням об’єктно-орієнтованої мови програмування C# в середовищі для розробки програмного забезпечення Visual Studio. У системі реалізовано додавання пацієнта, його редагування, видалення, систему акаунтів, створення статистики на основі отриманих даних за певний період, формування історій хворіб.

За період розробки курсового проєкту було виділені деякі етапи такі як:

* формулювання вимог до програми та проєктування;
* створення UML-діаграм для полегшеної розробки проєкту;
* кодування;
* тестування;
* написання пояснювальної записки.

Записка складається з таких розділів:

* загальний розділ;
* розробка технічного та робочого проєкту,
* спеціальний розділ.

Пояснювальна записка містить 3 розділи. В загальному розділі вказана постановка задачі, описані вхідні і вихідні дані та розглянуті існуючі методи та їх вирішення.

У розділі «Розробка технічного та робочого проєкту» описана структура організації вхідних даних, розробка структурної схеми та моделі проєкту на основі UML-діаграм, та описаний процес об’єктно-орієнтованого проєктування та програмування.

В спеціальному розділі є інструкція з інсталяції та експлуатації проєкту.

Отже, це програмне забезпечення дозволяє полегшити роботу лікарям-інфекціоністам.

**ЗМІСТ**

[Вступ 1](#_Toc102769143)

[1 Загальний розділ 3](#_Toc102769144)

[1.1 Постановка задачі 3](#_Toc102769145)

[1.2 Опис вхідної інформації 4](#_Toc102769146)

[1.3 Опис результуючої інформації 7](#_Toc102769147)

[1.5 Опис існуючих методів та рішень 8](#_Toc102769148)

[2. Розробка технічного та робочого проекту 10](#_Toc102769149)

[2.1 Опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних 10](#_Toc102769150)

[2.2 Розробка структурної схеми проекту та об’єктної моделі задачі на основі UML діаграм 10](#_Toc102769151)

[2.3 Об’єктно-орієнтоване проектування, програмування та тестування 17](#_Toc102769152)

[3 Спеціальний розділ 21](#_Toc102769153)

[3.1 Інструкція з інсталяції розробленого проекту 21](#_Toc102769154)

[3.2 Інструкція з експлуатації проекту 22](#_Toc102769155)

[Висновки 29](#_Toc102769156)

[Список використаних джерел 30](#_Toc102769157)

[Додатки 31](#_Toc102769158)

## 

**ВСТУП**

Понад 500 мільйонів людей у всьому світі заразилися вірусом COVID-19, у простонародді «ковід», і понад 6 мільйонів людей померли від захворювань, викликаних цим збудником. Але звідки взявся цей високо інфекційний вірус, що зумів увірвався в нашу буденність швидко та безпринципно і буквально протягом декількох місяців зруйнувати економіку, а також залишити незліченну кількість людей без засобів до існування?

Саме в невеличкому китайському місті Ухань зародилася ця глобальна пандемія. В 2019 році він швидкими темпами розповсюдився по всьому світу, створивши чимало проблем для суспільства. Саме тоді професія лікаря почала по-справжньому цінуватись у всьому світі. На плечі одного хороброго борця із цією світовою хворобою звалилися десятки, а то й сотні хворих, які довірили йому своє життя.

Вести паперовий облік усіх пацієнтів – це далеко не легка справа. Необхідно пам’ятати усі їхні показники, щоб на цій основі продовжувати якісне та правильне лікування. Але коли у тебе сотні пацієнтів з різними показниками самостійно з цим впоратись майже не можливо. Також велику роль відіграє постійна тяганина із історіями хворіб.

Оскільки ми живемо в сучасному світі, де технології направлені на полегшення роботи та створення комфорту, я вирішив оптимізувати цю без перебільшень титанічну працю та полегшити життя лікарю-інфекціоністу.

Всі ці можливості надає нам мова програмування С# та об’єктно-орієнтоване програмування.

**1 Загальний розділ**

**1.1 Постановка задачі**

На підставі індивідуального завдання, затвердженого на засіданні циклової комісії програмного забезпечення, протокол № 7 від 25.01.2022р.  необхідно розробити проєкт “Облік ковід-хворих лікарем інфекціоністом”.

Метою курсового проекту є розробка програми з використанням об’єктно-орієнтованого програмування для автоматизації обліку хворих, створення меню для роботи з програмою, введення даних про пацієнта, збереження інформації, редагування та видалення, пошук необхідної інформації, вивід необхідної інформації.

У програмі необхідно передбачити :

* створення зручного меню для роботи;
* додавання пацієнта;
* видалення пацієнта;
* додавання лікаря;
* видалення лікаря;
* можливість зміни паролю та логіну для лікаря;
* можливість зміни пароля для адміна;
* реалізувати пошук та фільтрацію даних на стороні адміна;
* редагування стану пацієнта;
* пошук по заданому критерію;
* збереження даних у файли;
* зручність програми;
* простота програми;
* декілька облікових записів лікарів з обмеженою видимістю та взаємодією з пацієнтом;
* обліковий запис для адміна з відповідними можливостями;
* створення мінімальної статистики на основі збережених даних;
* можливість перегляду інформації про кожного пацієнта;
* створення зручного меню для перегляду інформації про пацієнта;
* формування історії хвороби по заданому пацієнту.

Отже у програмі, я повинен передбачити всі пункти, які вказав вище.

**1.2 Опис вхідної інформації**

Вхідна інформація – це вся інформація, яка надходить у програму для опрацювання. Ця інформація може вводитися користувачем, або надходити з файлів які вводиться адміністратором та лікарями, роль яких будуть передбачені в проекті.

Вхідною інформацією проекту будуть дані різних типів, що будуть вводитися користувачем під час роботи із програмою з клавіатури, а саме інформація про:

Хворих:

* прізвище (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* ім’я (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* по батькові (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* громадянство (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* група крові (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* резус фактор (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* домашня адреса (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* тип документа що засвідчує особу (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* номер документу (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні);
* ID пацієнта (тип цілочисельний; береться з файлу, або автоматично формується програмою) ;
* прізвище (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* дата народження (тип дата; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* дата госпіталізації (тип дата; береться з файлу; береться з файлу, або автоматично формується програмою) ;
* дата виписки (тип дата; береться з файлу; береться з файлу, або автоматично формується програмою) ;
* стать (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* діагноз клінічний (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* діагноз при госпіталізації (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* ким направлений (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* номер палати (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації пацієнта/редагуванні) ;
* ID лікуючого лікаря (тип цілочисельний; береться з файлу, або автоматично формується програмою) ;

Лікарів:

* прізвище (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;
* ім’я (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;
* по батькові (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;
* громадянство (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;
* група крові (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;
* резус фактор (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;
* домашня адреса (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;
* тип документа що засвідчує особу (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;
* номер документу (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;
* ID лікаря (тип цілочисельний; береться з файлу, або автоматично формується програмою) ;
* кваліфікація (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;
* посада (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;
* логін (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;
* пароль (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;
* дата народження (тип дата; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;
* стать (тип строкові дані; береться з файлу, або вводиться користувачем при реєстрації лікаря/редагуванні) ;

У разі будь-якої зміни даних в програмі, ці зміни повинні бути записані у файл збереження даних

Отже вхідна інформація, це та інформація яка надходить у програму для опрацювання. Вона може надходити з файлів даних, або вводитись користувачем вручну з клавіатури.

**1.3 Опис результуючої інформації**

Результуюча інформація – це інформація, яка отримується в результаті опрацювання вхідної інформації, а також збережена інформація.

Результуючою інформацією проекту є збереження інформації у файлах та вивід інформації про всіх пацієнтів та лікарів.

Після заповнення та редагування усієї інформації пов’язаної з хворими, користувач отримує таблицю, яка зручно показує, всіх хворих та інформацію про них.

Також є додаткова форма для редагування, яка показує нам, що ми можемо змінити. Результати форми редагування також виводяться у загальну таблицю, де ми можемо чітко побачити всі зміни.

Отже результуюча інформація, це інформація яку ми отримаємо в результаті роботи з вхідною інформацією, в даному випадку це серіалізація об’єктів класу у бінарний файл, а також у вона виводиться на екран користувачу.

**1.4 Стадії та етапи розробки задачі**

Для того, щоб реалізувати поставлену задачу, її потрібно розділити на окремі під-задачі, кожна з яких повинна логічно завершуватись та виконувати певну функцію:

* аналіз предметної області – визначити усі складові предметної області, основу їх роботи, визначити методи їх виконання структурувати опис змінних. Визначити основні проблеми системи, щоб обійти під час етапу проектування чи замінити їх певною альтернативою:
* проектування структури вхідних даних;
* вибір програмних продуктів для реалізації поставлених цілей, пошуку помилок та попереднього тестування;
* визначення методів організації обробки структур вхідних та вихідних даних;
* проектування класів, архітектури проекту, інтерфейсу проекту;
* програмування та тестування функціоналу;
* дослідна експлуатація проекту.

Визначенням мети та завдання проекту було виконано ще на початковому етапі планування курсового проекту та під час вибору теми для опрацювання.

Під час етапу аналізу предметної області, було визначено основні структурні одиниці, що потрібні для мінімального функціонування проекту. Було виконано аналіз їхньої роботи та вивчено альтернативні способи їх реалізації.

Для запобігання проблем у зв’язку з введенням некоректних даних необхідно створити систему обмежень та передбачити загальну логіку програми.

На етапі вибору програмних продуктів для реалізації поставлених завдань, стало зрозуміло, що з переліку доступних засобів, які в майбутньому будуть найкомфортнішими і зроблять виконання роботи найбільш ефективнішими. Було обрано програмне забезпечення Microsoft Visual Studio 2019.

Збереження даних буде реалізовано за допомогою серіалізації та десеріалізації файлів. Оскільки серіалізація у бінарні файли забезпечує певний рівень захисту даних, та відносну простоту в реалізації, коли не потрібно створювати ще додаткові таблиці для роботи в базах даних. Для збереження та запису даних буде використано декілька класів, а саме “Hvoryi”, “People”, “Doctor”.

Наступним етапом є архітектурне проектування. На цьому етапі вирішено представити даний проект у вигляді одного десктопного додатку. Данна програма буде містити декілька форм, які в свою чергу - багато додаткових компонентів, які реалізовують роботу з різними даними. Дані компоненти форм детальніше можна розглянути в діаграмі компонентів.

Отже, етапи роботи з програмою можна розділити на архітектурне проектування, та на роботу з вхідною та вихідною інформацією.

**1.5 Опис існуючих методів та рішень**

Для того, щоб проект був максимально зручним, а його функціонал найбільше відповідав меті, потрібно розглянути існуючі рішення на ринку.

Helsi - [медична інформаційна система](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B4%D0%B8%D1%87%D0%BD%D0%B0_%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) для закладів охорони здоров'я та медичний портал для пацієнтів в Україні. Медична інформаційна система автоматизує роботу закладу охорони здоров‘я, лікаря, лабораторії, стаціонару, ведення електронних медичних карток. Система має зручний та зрозумілий інтерфейс, допомагає формувати звіти, має вбудований конструктор бланків і форм медичних документів та функціонал для роботи за програмою медичних гарантій з [Національною службою здоров'я України](https://uk.wikipedia.org/wiki/%D0%9D%D0%B0%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0_%D0%B7%D0%B4%D0%BE%D1%80%D0%BE%D0%B2%27%D1%8F_%D0%A3%D0%BA%D1%80%D0%B0%D1%97%D0%BD%D0%B8) (НСЗУ). [10]

Недоліком даної платформи є необхідність постійного доступу в мережу і на вузьконапрямлених закладах, де є закртита мережа без доступу в глобальну павутину, цей сайт категорично не підходить.

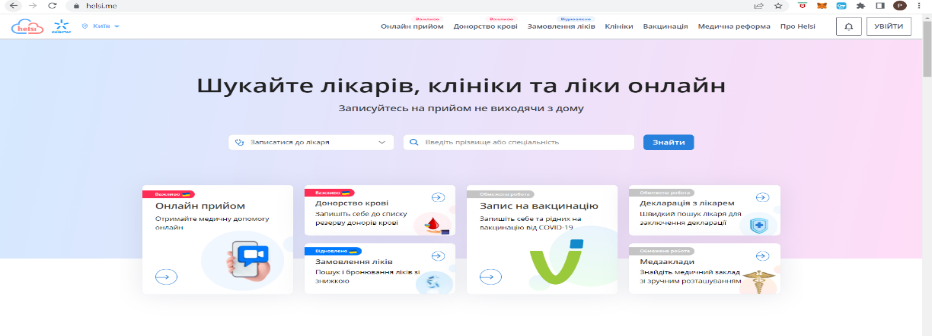


Рисунок 1.1 – Зразок існуючих методів

Doc.ua - безкоштовний інтернет-сервіс в Україні з пошуку лікарів і запису на прийом. Спеціалізується на наданні інформації щодо медичних фахівців, медичних установ. Допомагає пацієнтам у підборі лікарів, клінік і записі на прийом. Проект заробляє на комісії від клініки за пацієнта, який записався через сервіс та здійснив візит до лікаря. Для пацієнтів сервіс є безкоштовним. Запис на прийом можливий через онлайн-форму на сайті або телефоном.[9]

Негативним у даному випадкує залишається необхідність для роботи в програмі постійного доступу до мережі інтернет та зайва нагруженість проекту, яка не є функціонально потрібною у медичних закладах із закритою системою документообігу.

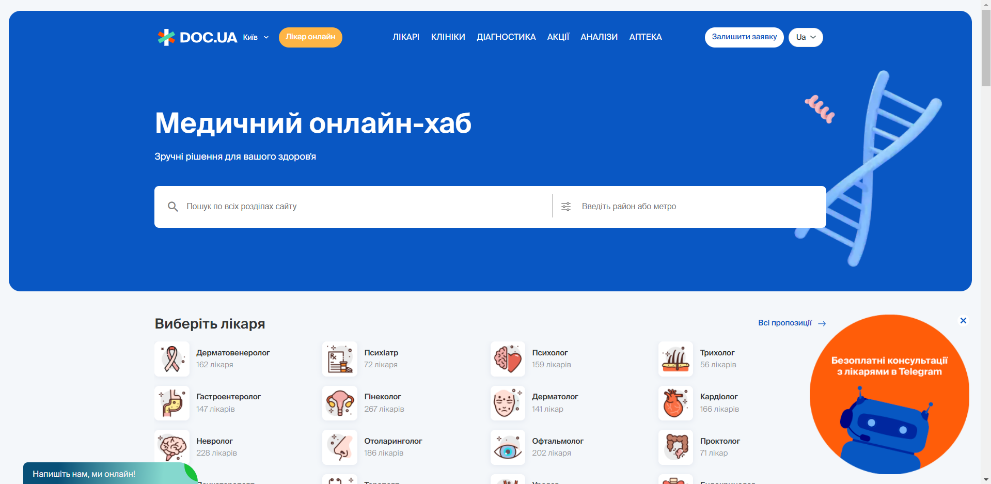


Рисунок 1.2 – Зразок існуючих методів

Оскільки мій вибір впав на мову C# яка є об’єктно-орієнтованою мовою програмування, підтримує безліч бібліотек для роботи з даними, я буду створювати програму під ОС Windows. Ця мова є більш зручнішою у використанні, та має більше функцій, ніж ті самі мови C, C++, Java, JavaScript. Також я обрав С# для роботи з Windows Forms оскільки Microsoft фактично для цього її створювала, для зручного та відносно швидкого візуального програмування під платформи сімейства Windows.

Отже, розглянувши можливі варіанти рішень та враховуючи обмеження з часом, я вирішив розробляти ПЗ на об’єктно-орієнтованій мову програмування С#.

**2. Розробка технічного та робочого проекту**

**2.1 Опис та обґрунтування вибору структури та методу організації вхідних та вихідних даних**

В програмуванні та комп’ютерних науках структура даних – це спосіб організації даних в комп’ютерах. Часто разом зі структурою даних пов’язується і специфічний перелік операцій, що можуть бути виконаними над даними, організованими в таку структуру.

Правильний вибір структур даних є надзвичайно важливим для ефективного функціонування відповідних алгоритмів, їх обробки. Добре побудовані структури даних дозволяють оптимізувати використання машинного часу та пам’яті комп’ютера для виконання операцій.

За постановкою задачі завданням є створення проекту для збереження, перегляду, видалення великих обсягів інформації, а також реалізація швидкого доступу до інформації, та зручного пошуку інформації.

У курсовому проекті для вводу і виведення інформації буде використано бінарні файли збереження, які реалізуються за допомогою серіалізації та відповідно десеріалізації розроблених у майбутньому класів ”Doctor” та ”Hvoryi”, опис інформації, що зберігається у них, детально наведена розділі 1.2 пояснювальної записки до проекту “Облік ковід-хворих лікарем інфекціоністом”.

Коли у файлі міститься великий об’єм інформації, дуже важливо, щоб дані швидко завантажувались та вивантажувались. Також збереження інформації у бінарний файл забезпечує її структурованість, що допомагає швидко та легко здійснювати пошук та виводити необхідну інформацію.

Отже, найзручнішим методом для структурування даних є збереження їх у бінарні файли.

**2.2** **Розробка структурної схеми проекту та об’єктної моделі задачі на основі UML діаграм**

UML – це уніфікована мова моделювання, яка використовується в парадигмі об’єктно-орієнтованого програмування. Це невід’ємна частина єдиного процесу розробки програмного забезпечення. UML — це мова широкого формату, відкритий стандарт, який використовує графічні нотації для створення абстрактних моделей систем, які називаються моделями UML. UML був створений для визначення, візуалізації, проектування та документування більшості програмних систем. UML не є мовою програмування, але код може бути згенерований шляхом виконання моделі UML як інтерпретованого коду.

Головними об’єктами є користувач (лікар) та адміністратор. Вони виступають в ролі актора. Саме користувач здійснює всі дії, після додавання його в систему адміністратором. Оскільки програмний продукт має реалізовувати роботу лікарів у медичному закладі та завідувачів відділень (адміністраторів), тому акторів лише два. []

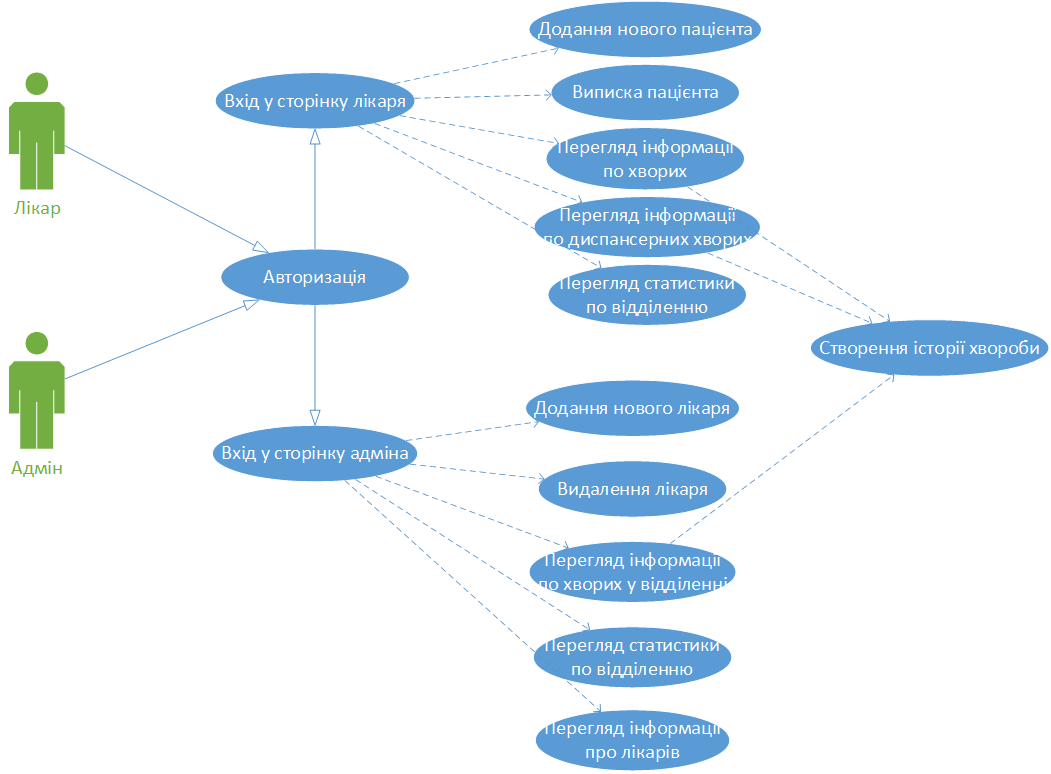


Рисунок 2.1 – Use case diagram

Class diagram основна діаграма для створення коду додатка. За допомогою діаграми класів створюється внутрішня структура системи, описується спадкування й взаємне положення класів одного щодо іншого. Клас в мові UML служить для позначення множини об’єктів, які мають однакову структуру, поведінку і відносини з об’єктами інших класів.

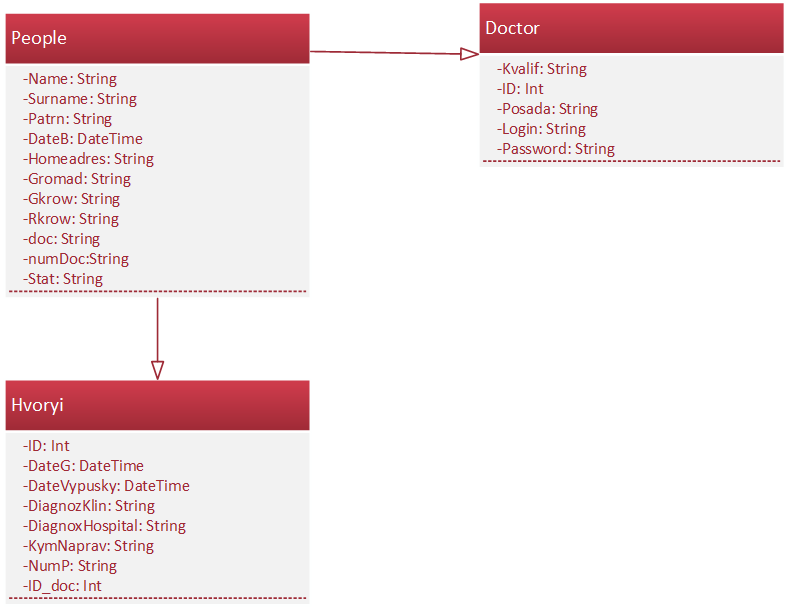


Рисунок 2.2 – Class diagram

Діаграми станів представляють скінчений автомат набором станів та переходів. Об'єкт можна описати з точки зору станів, у яких він може знаходитись, а також можливих переходів з одного стану в інший у випадку виникнення певних подій. Свій скінчений автомат може бути пов'язаний з будь-яким класом моделі.

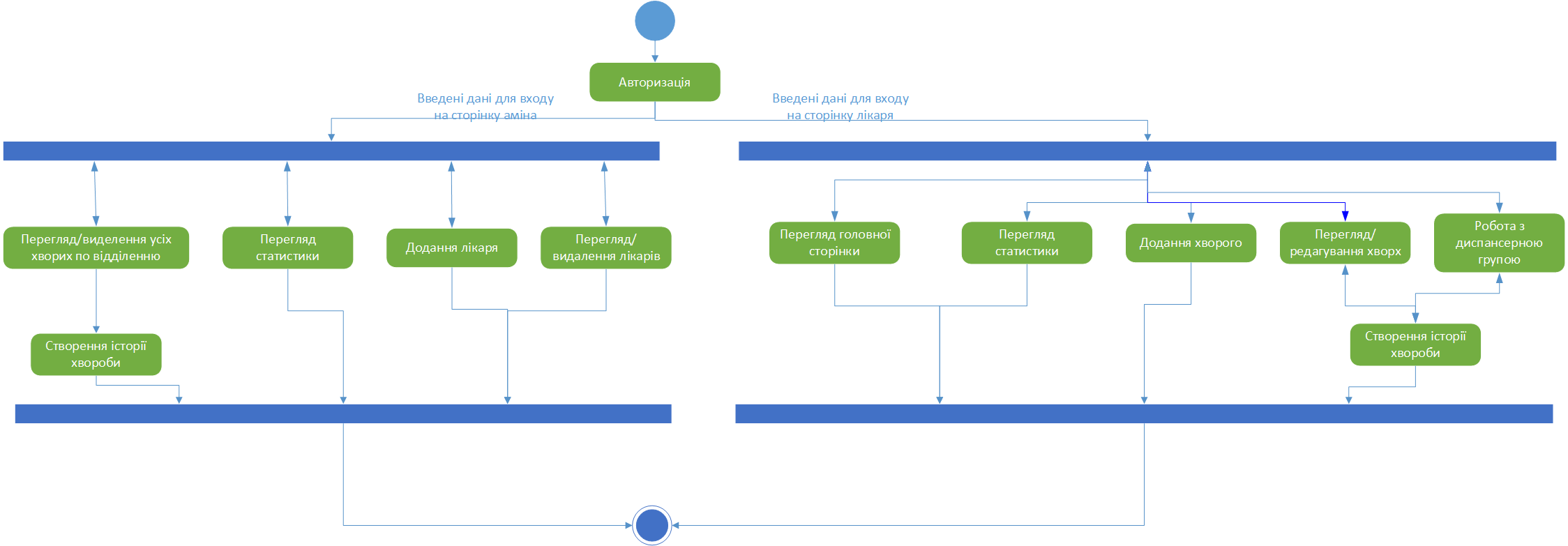


Рисунок 2.3 – State chart diagram

Діаграма діяльності (діаграма активності) дозволяє моделювати послідовності бізнес-процесів або дій, реалізованих методами класів. Зазначені послідовності можуть являти собою альтернативні галузей процесу обробки даних або галузям, які можуть виконуватися паралельно. Діаграми діяльності є аналогом блок-схеми будь-якого алгоритму. Вони, як і діаграми станів та переходів, відображаються у вигляді орієнтованого графу, вершинами якого є дії, а ребрами – переходи між діями.

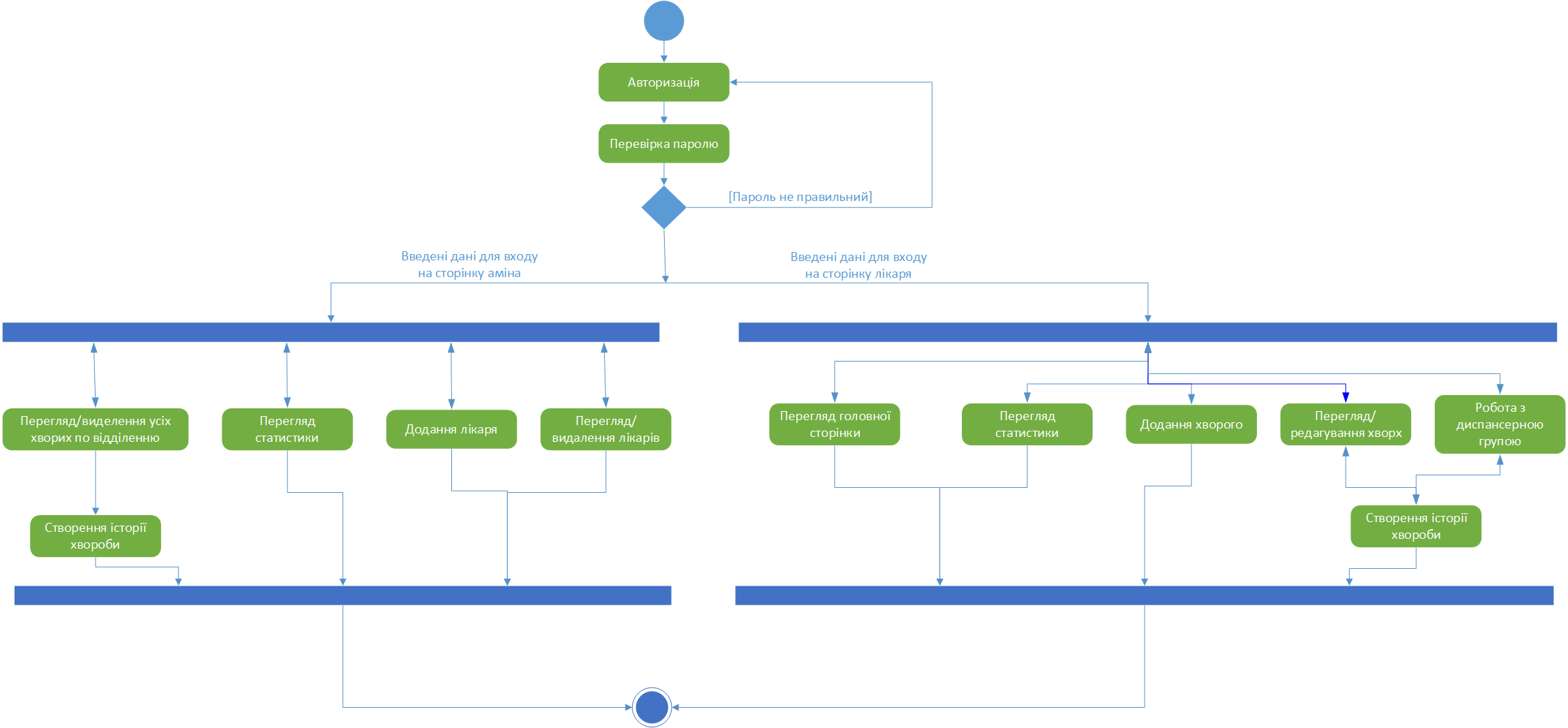
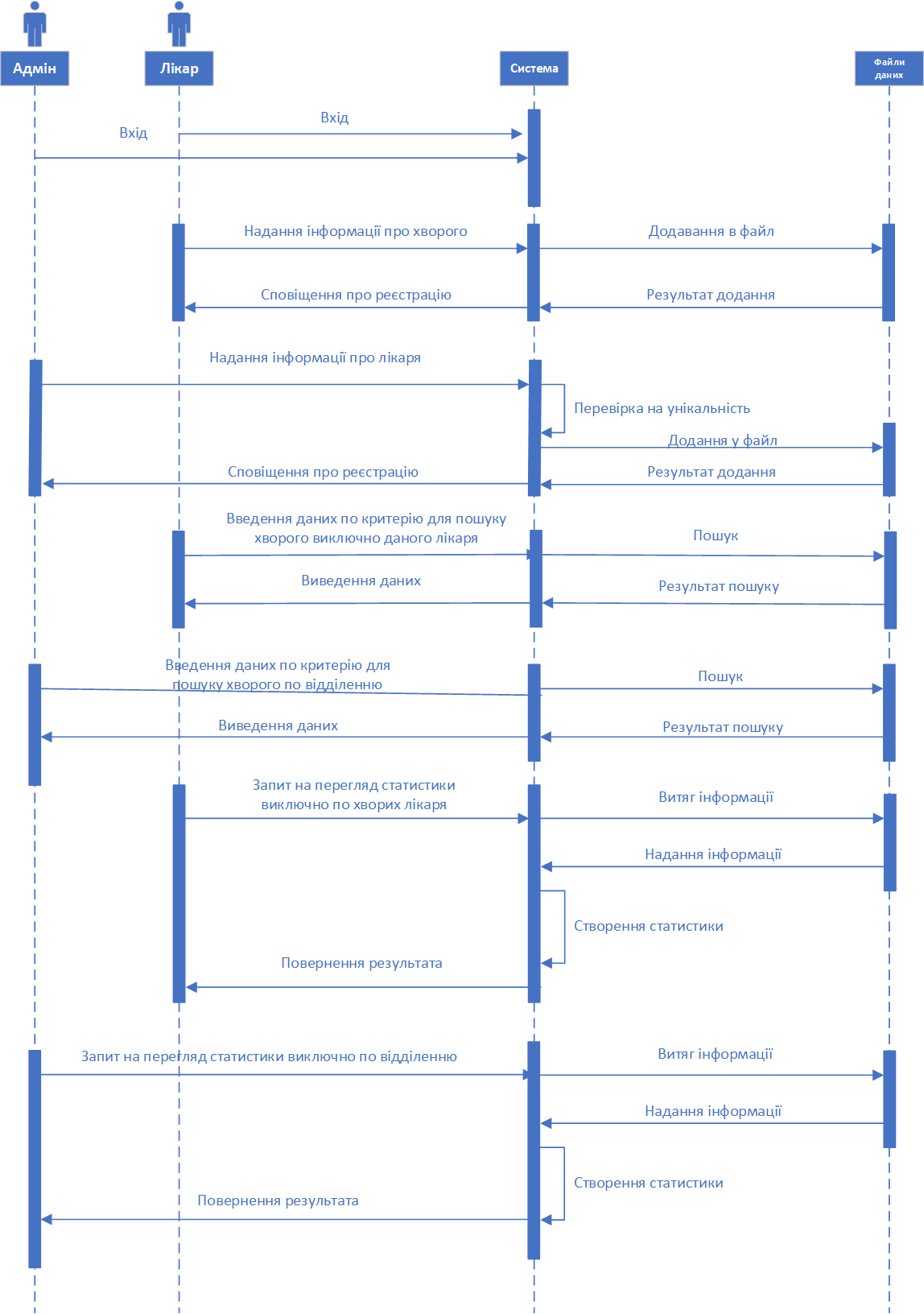


Рисунок 2.4 –Activity diagram

Для моделювання взаємодії об'єктів у мові UML використовуються відповідні діаграми взаємодії. Говорячи про ці діаграми, мають на увазі два аспекти взаємодії. По-перше, взаємодії об'єктів можна розглядати в часі, і тоді для подання тимчасових особливостей передавання і приймання повідомлень між об'єктами використовується діаграма послідовності.

Рисунок 2.5 – Sequence diagram

Поняття кооперації (collaboration) є одним з фундаментальних понять у мові UML. Воно служить для позначення безлічі взаємодіючих з певною метою об'єктів в загальному контексті модельованої системи.

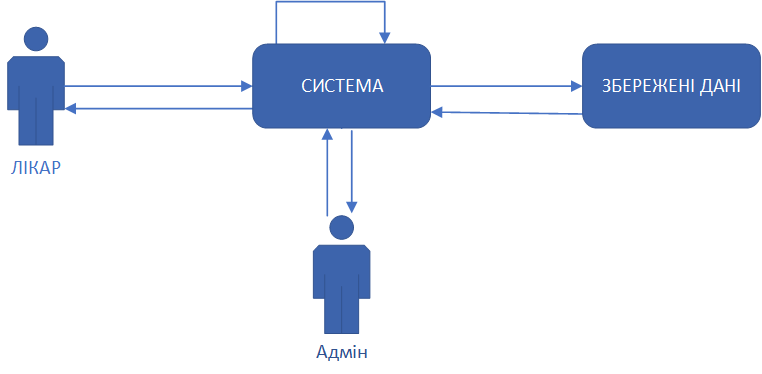


Рисунок 2.6 – Collaboration diagram

Діаграма компонентів розробляється для з ціллю візуалізації загальної структури вихідного коду програмної системи,  
забезпечення багаторазового використання окремих фрагментів програмного коду.

Компонент (Component) відповідає програмному модулеві з добре визначеним інтерфейсом. У вікні специфікації можна задати стереотип (Actіve, Applet, Applіcatіon, DLL, EXE та ін.)

Специфікація і тіло пакета (Package Specіfіcatіon and Body). Пакет у даному випадку - це реалізація класу. Специфікацією пакета є заголовний файл зі зведеннями про прототипи функцій для класу. Також в діаграмі компонентів часто відображають з чого складатиметься розроблюване програмне забезпечення, тобто з яких фреймворків, бібліотек, модулів, файлів, що будуть підключені та використані у ПЗ.

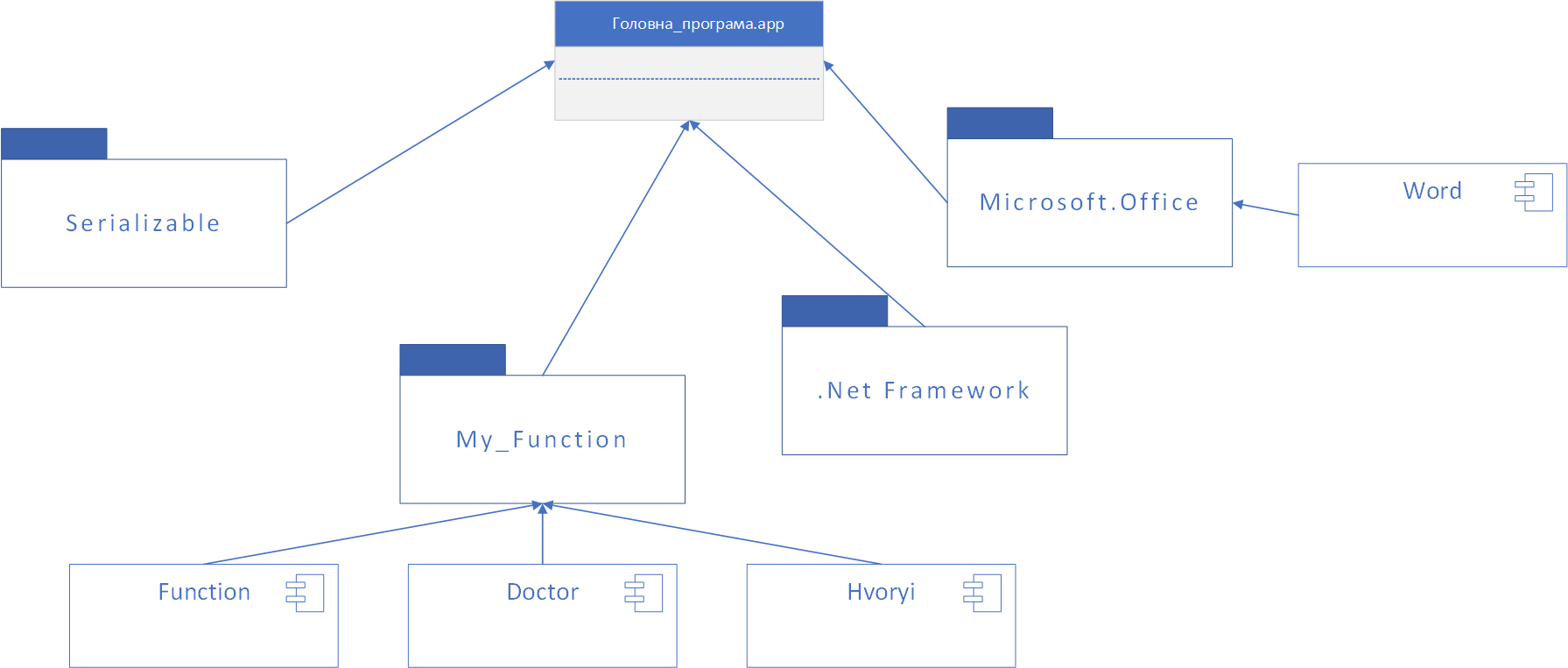


Рисунок 2.7 – Component diagram

Діаграма розгортання — діаграма на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах.



Рисунок 2.8 – Deployment diagram

**2.3** Об’єктно-орієнтоване проектування, програмування та тестування

Написання програми відбудеться на об’єктно-орієнтованій мові С# з використанням технології .NET, в середовищі програмування Visual Studio 2019. Це середовище дозволяє швидко та якісно писати код на різних мовах, воно має автоматичне правлення та часткову генерацію коду по шаблону, що суттєво спрощує процес написання програм.

У проекті застосовано 3 класи, а саме: “People”,”Doctor”,”Hvoryi”.

Саме для того, щоб користувачу було зручно користуватись програмою, було вирішено всю інформацію розбити на пару меню. Таким чином утворились меню перегляду усіх хворих, меню редагування кожного з них та меню перегляду усіх хворих переведених на диспансерний нагляд. В кожному з цих меню робота з інформацією про різні об’єкти представлені у різних інтерфейсах

Для зберігання даних використається процес перетворення структури даних в послідовність бітів - серіалізація, а для її коректної роботи заздалегідь необхідно прописати в класі, який будемо серіалізовувати, спеціальний атрибут та підключити відповідні розширення.

Для роботи з кожним класом було розроблено окремі інтерфейси, з дотриманням основних вимог для User Friendly інтерфейсу, а саме:

* Бути інтуїтивно зрозумілим
* Бути мінімалістичним
* Швидко завантажуватись
* Бути створеним на зрозумілій для користувача мові
* Мати короткі форми для заповнення

Саме для того, щоб користувачу було зручно користуватись програмою було вирішено всю інформацію розбити на декілька меню.

Для керування файлами Word підключаємо пакет NuGet “Microsoft.Office.Interop.Word” та прописуємо наступну логіку :

using Word = Microsoft.Office.Interop.Word;

private void ReplaceWordStud(string stubToReplace, string text, Word.Document wordDocument)

{

var range = wordDocument.Content;

range.Find.ClearFormatting();

range.Find.Execute(FindText: stubToReplace, ReplaceWith: text);

}

У частині кода, описаній вище, реалізовано заміну символів в документу Word за допомогою “Microsoft.Office.Interop.Word”. Дальше буде фрагмент як це виглядатиме в коді:

...

for (int i = 0; i < dataGridView1.RowCount; i++)

{

if (k == Convert.ToInt32(dataGridView1.Rows[i].Cells[0].Value))

{

MessageBox.Show("Зачекайте відриття Word, ваш запит на виконанні", "Процес", MessageBoxButtons.OK, MessageBoxIcon.Information);

var wordApp = new Word.Application();

wordApp.Visible = false;

var wordDocument = wordApp.Documents.Open(TempleteFileName);

ReplaceWordStud("{ID}", Convert.ToString(dataGridView1.Rows[i].Cells[0].Value), wordDocument);

ReplaceWordStud("{DateB}", Convert.ToString(dataGridView1.Rows[i].Cells[4].Value), wordDocument);

ReplaceWordStud("{DateG}", Convert.ToString(dataGridView1.Rows[i].Cells[5].Value), wordDocument);

ReplaceWordStud("{Stat}", Convert.ToString(dataGridView1.Rows[i].Cells[7].Value), wordDocument);

ReplaceWordStud("{OldYear}", Convert.ToString(dataGridView1.Rows[i].Cells[8].Value), wordDocument);

ReplaceWordStud("{Home}",

wordDocument.SaveAs2(@"D:\result.docx");

wordApp.Visible = true;

break;

}

}

Для створення користувача, в нашому випадку – лікаря, пропишемо наступну логіку:

private void button1\_Click(object sender, EventArgs e)

{

if (textBoxLog.Text == "" || textBoxPosada.Text == "" || textBoxPass.Text == "" || textBoxSurN.Text == "" || textBoxNam.Text == "" || textBoxPatrn.Text == "" || textBoxHome.Text == "" || textBoxNumDoc.Text == "" || textBoxKval.Text == "" || textBox2.Text == "")

{

MessageBox.Show("Заповніть усі поля!!!");

}

else

{

for(int l = 0; l<doctors.Count;l++)

{

if(doctors[l].getLogin() == textBoxLog.Text)

{

MessageBox.Show("Такий логін уже існує");

break;

}

}

doctors.Add(createDoc());

writeAll();

MessageBox.Show("Лікар доданий");

UpDat();

ClearPage1();

ID++;

label10.Text = ID.ToString();

}

}

У фрагменті коду, описаному вище, використовуються власні функції для роботи з динамічними структурами, а саме: createDoc(), UpDat(), writeAll().

createDoc() - формує екземпляр класу, для подальшого його додавання в динамічну структуру даних власного класу.

private Doctor createDoc()

{

Doctor doctor = new Doctor();

doctor.setDateB(dateTimePicker1.Value);

doctor.setDoc(comboBox2.Text);

doctor.setGkrow(textBox2.Text);

doctor.setGromad(comboBox5.Text);

doctor.setHomeadres(textBoxHome.Text);

doctor.setID(Convert.ToInt32(label10.Text));

doctor.setKvalif(textBoxKval.Text);

doctor.setLogin(textBoxLog.Text);

doctor.setName(textBoxNam.Text);

doctor.setNumDoc(textBoxNumDoc.Text);

doctor.setPasswrd(textBoxPass.Text);

doctor.setPatrn(textBoxPatrn.Text);

doctor.setPosada(textBoxPosada.Text);

doctor.setRkrow(comboBox1.Text);

if (radioButton1.Checked)

doctor.setStat(radioButton1.Text);

else doctor.setStat(radioButton2.Text);

doctor.setSurname(textBoxSurN.Text);

return doctor;

}

writeAll() – записує у файл з користувачами нового користувача.

private void writeAll()

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Лікарі.dat", FileMode.OpenOrCreate))

binaryFormatter.Serialize(fileStream, doctors);

}

UpDat() – зчитує дані з файлу запису для подальшої роботи з нами, у разі коли файлу в нас не існує передбачене виключення конструкцією “TryCatch”

private void UpDat()

{

doctors.Clear();

try

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Лікарі.dat", FileMode.OpenOrCreate))

{

doctors = (List<Doctor>)binaryFormatter.Deserialize(fileStream);

}

}

catch { }

}

Далі напишемо логіку для пошуку по критерію:

if (comboBox1.SelectedIndex == 0)

{

kk = 0;

t = textBox2.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && hvoryis[i].getSurname().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

Використовуючи відкриті методи класу, звертаємося до змінних та отримуємо можливість безпосередньо порівнювати їх. Функція “DatSetToGrit” відповідає за виведення одержаної в результаті пошуку інформації користувачу.

Задля унеможливлення “вильоту” програмного забезпечення, у всіх програмних блоках, де це можливо в наслідо некоректних дій користувача, я прописав конструкції виключень “TryCatch”. Якщо виконані некоректні дії типу спроби виписати пацієнта коли він ще не доданий, див. рисунок 2.9, створити історію хвороби без вказано ID пацієнта, див рисунок 2.10, введення хибних даних при вході у програму, див рисунок 2.11 і тому подібні, програма виводить відповідні повідомлення на екран користувача.

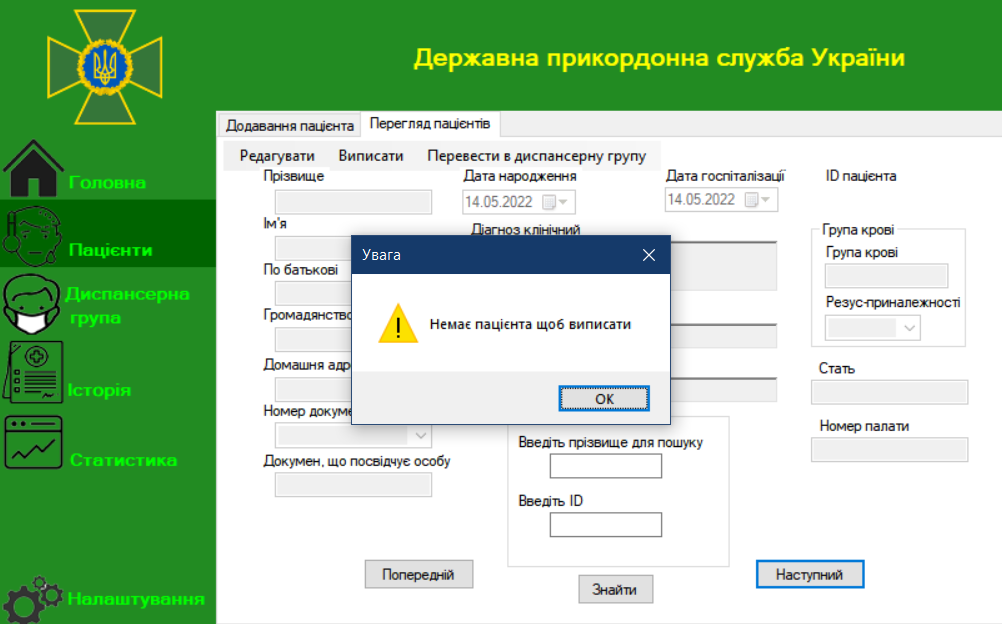


Рисунок 2.9 – Зразок тестування програми на завадостійкість

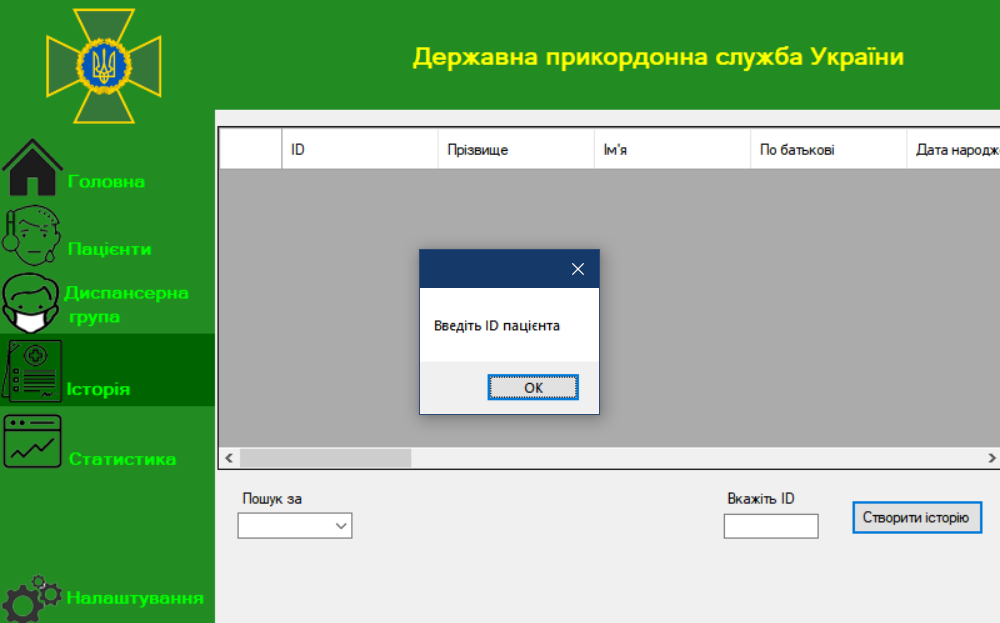


Рисунок 2.10 – Зразок тестування програми на завадостійкість

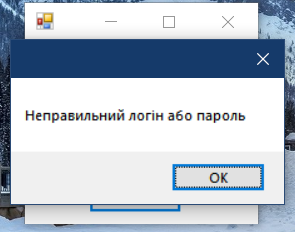


Рисунок 2.11 – Зразок тестування програми на завадостійкість

Отже, на даному етапі роботи було реалізовано: роботу з пакетами розширень NuGet; роботу з усіма необхідними даними; їх збереженням у файл даних та доступ до них; налаштовано коректну роботу авторизації та додавання облікового запису; організація роботи з документом-шаблоном Word; оптимізацію коду. Також були пропрацьовані всі проблемні місця в коді де можливі некоректні дії користувача, та створенні відповідні виключання для них, з виводом повідомлення на екран користувача. Все це допомогло написати логіку для коректної роботи програми.

**3 Спеціальний розділ**

**3.1 Інструкція з інсталяції розробленого проекту**

Для інсталяції програми – “Облік ковід-хворих лікарем інфікціоністом”, необхідні такі мінімальні системні вимоги до ПК:

* + тактова частота процесору не менше 1 GHz;
  + операційна система Windows 7 і вище;
  + оперативна пам’ять 1 Гб та вище;
  + місце на жорсткому диску від 200 Мб;
  + інстальований Microsoft Word.

Щоб розпочати роботу з нашим проектом “Облік ковід-хворих лікарем інфікціоністом” ми повинні завантажити файл “Облік ковід-хворих лікарем інфікціоністом.msi” з носія та запустити його на виконання. Дальше перед вами відкриється стандартне інсталяційне меню, в якому необхідно вказати місце розташування програми та обрати додаткові опції, див. рисунок 3.1.

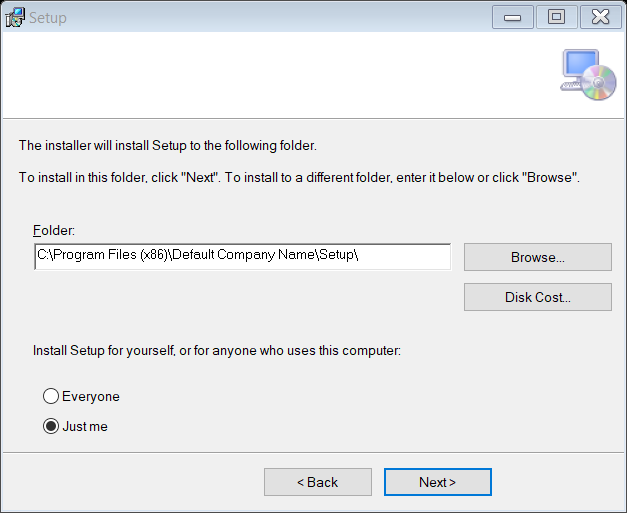
****

Рисунок 3.1 – Інсталятор проекту

Вказавжи необіхідні опції та натиснувши “Next” >> “Install”, програма буде встановлена на ваш персональний кома’ютер.

На даному етапі вказаний спосіб інсталяції програмного забезпечення на ПК та вимоги до нього.

* 1. **Інструкція з експлуатації проекту**

Щоб запупустити інстальовану програму *–* “Облік ковід-хворих лікарем інфікціоністом”*,* необхідно натиснути на якрик на робочому сторі, див. рисунок 3.2.



Рисунок 3.2 – Ярлик для відкриття програми

При запуску програми вас зустрічає форма для входу в система, див. рисунок 3.3.

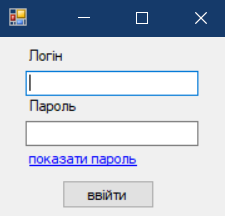


Рисунок 3.3 – Форма для входу

Оскільки це перший запуск додатку, то доступний лише кабінет адміністратор, оскільки в програмі ще не зареєстрований жодний користувач. Відповідно логін адміністратора – admin, пароль – admin1111.

Ввійшовши, ми бачимо інтерфейс завдяки якому можна створювати облікові записи лікарів в системі, див. рисунок 3.4.

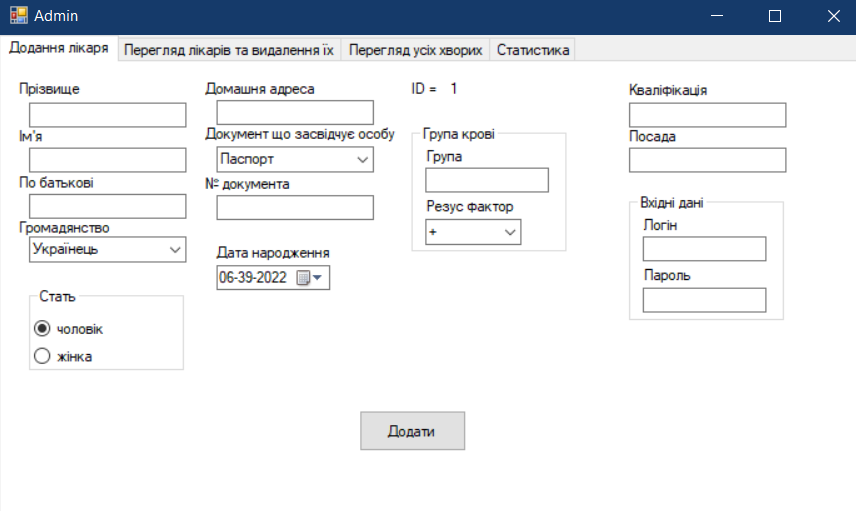


Рисунок 3.4 – Сторінка адміністратора (додання лікаря)

Також реалізовано вкладки для перегляду та видалення лікарів, перегляду статистики по відділенню та перегляд усіх хворих, див. рисунок 3.5-3.7.

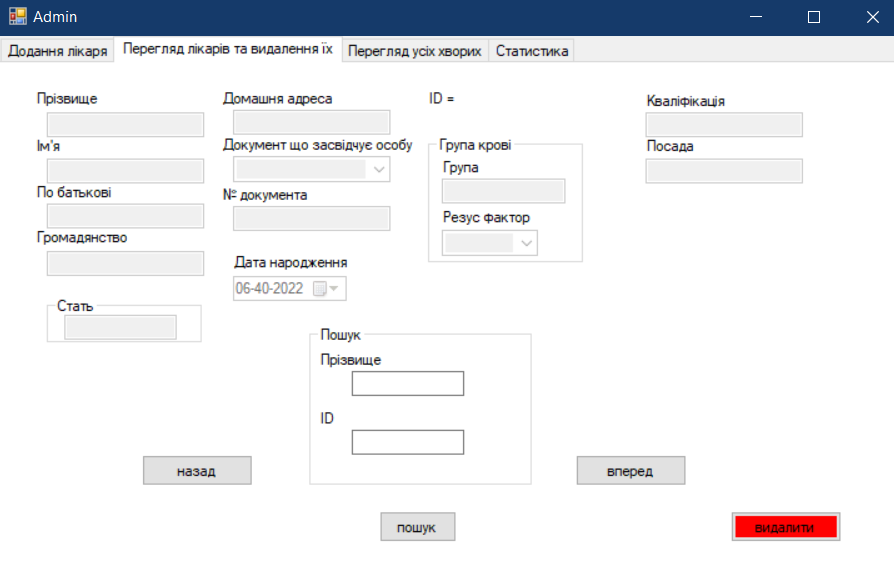


Рисунок 3.5 – Сторінка адміністратора (перегляд/видалення лікаря)

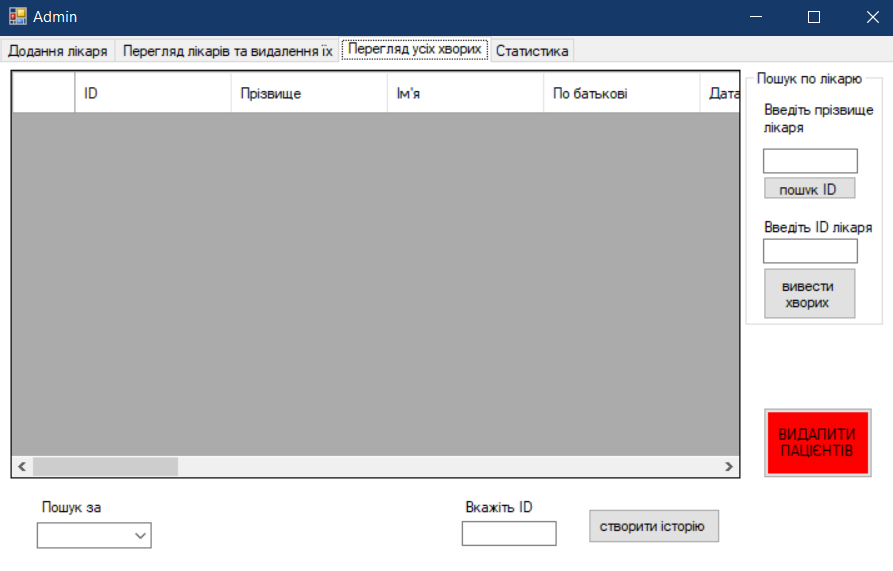


Рисунок 3.6 – Сторінка адміністратора (перегляд та пошук хворих)

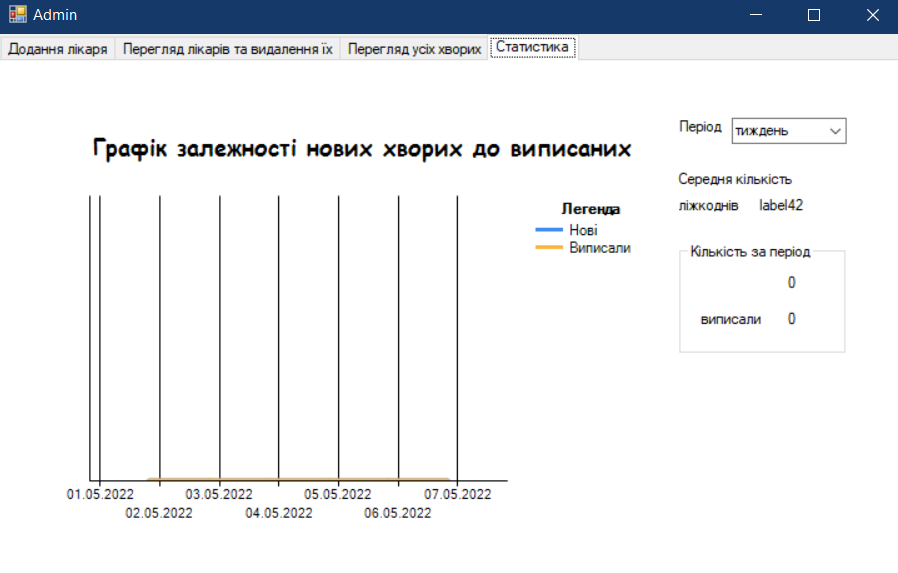


Рисунок 3.7 – Сторінка адміністратора (перегляд статистики по відділенню)

Після створення адміністратором облікового запису для лікаря, наступний може виконати вхід у системо використовуючи логін який його видав адміністратор та пароль, який згодом можна змінити в системі, див. рисунок 3.12.

Ввійшовши, лікар бачить сторінку “Головна” на яку винесено також коротка інформація про програму, див. рисунок 3.8.

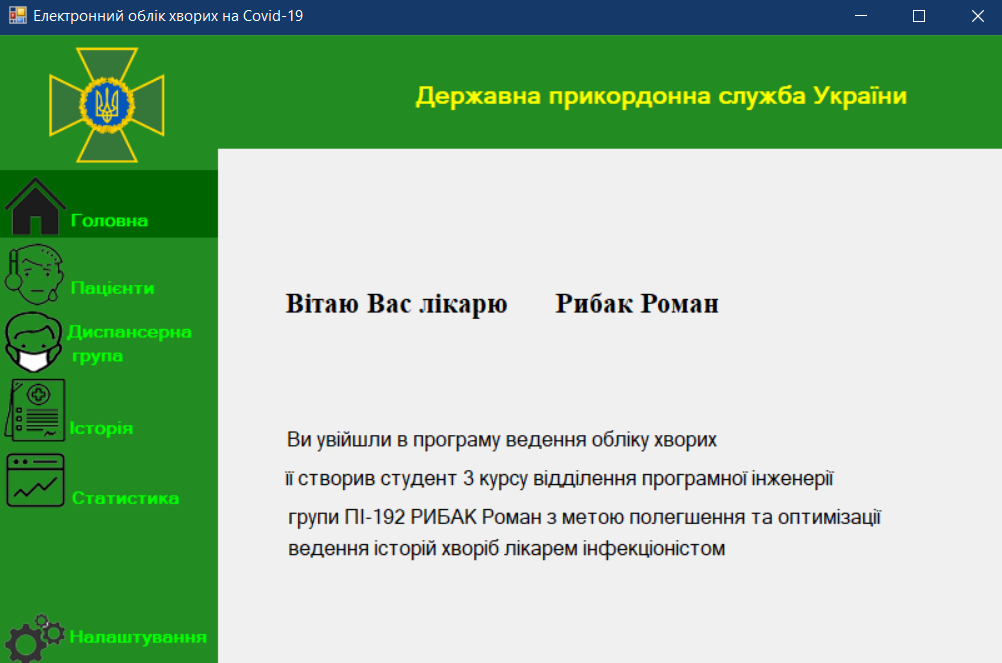
**

Рисунок 3.8 – Головна сторінка

Обравши вкладку “Пацієнти”, відкривається доступ до інтерфейсу для додавання пацієнта та редагування його даних або статусу як хворого, див. рисунок 3.9.1-2.

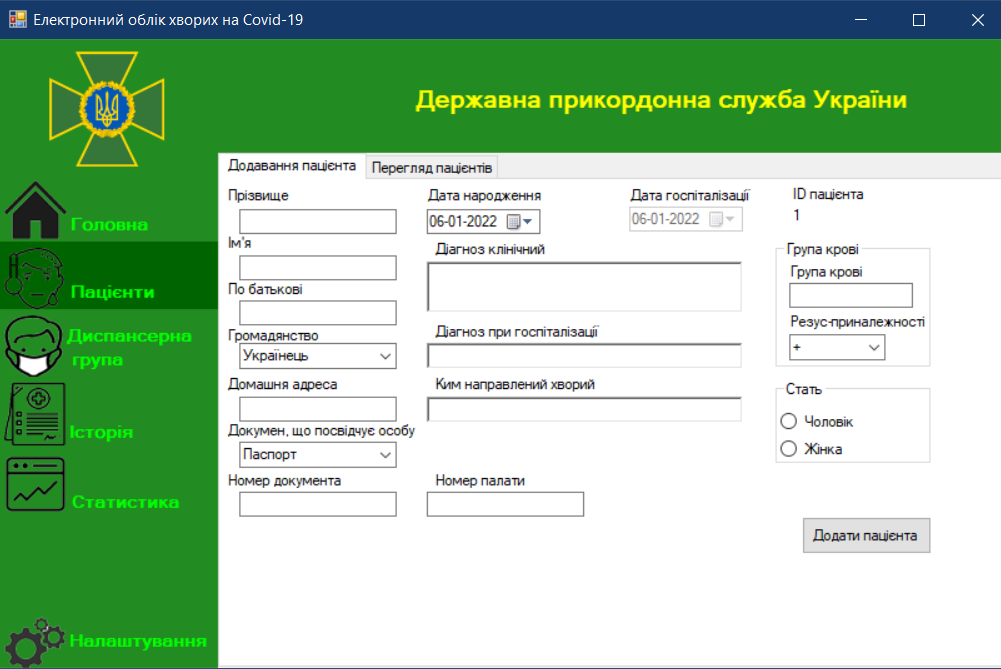
**

Рисунок 3.9.1 – Додавання пацієнта

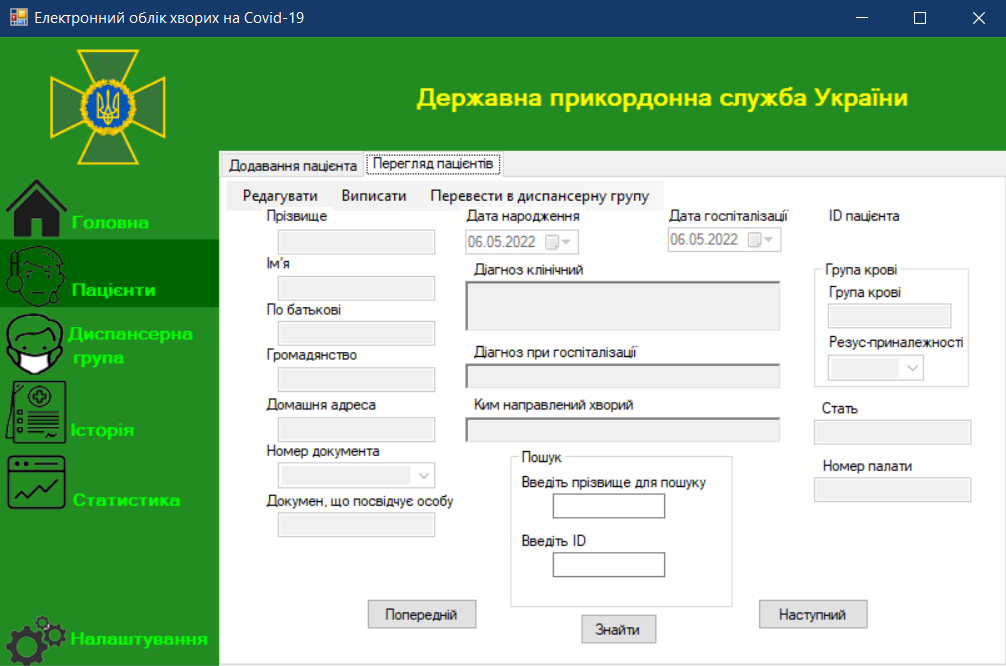
**

Рисунок 3.9.2 – Пошук та редагування його даних, статусу

Пацієнти, які були виписані та переведені на диспансерний огляд будуть відображатися на вкладці “Диспансерна група”, див. рисунок 3.10.

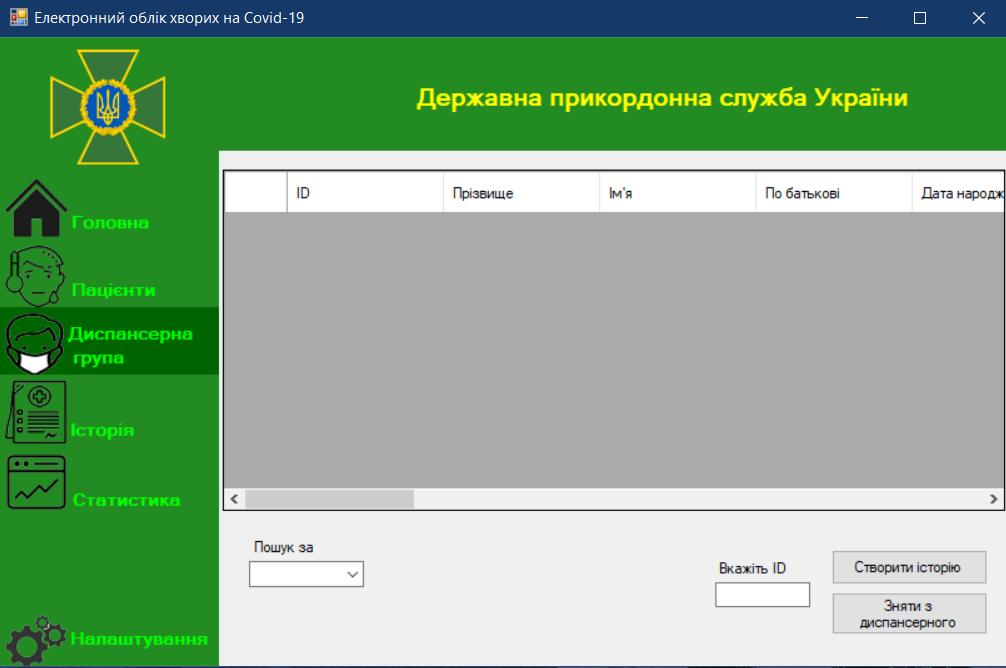
**

Рисунок 3.10 – Диспансерна група

Пацієнти, які були виписані або зняті з диспансерного огляду будуть відображатися на вкладці “Історія ”, див. рисунок 3.11.

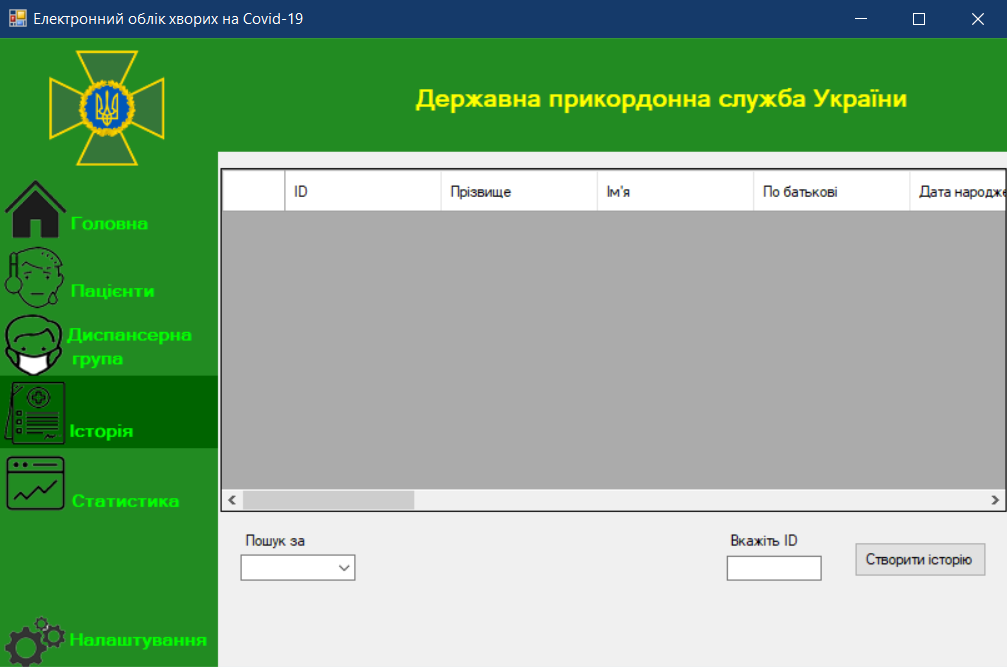
**

Рисунок 3.11 – Історія

Кожний лікар може переглянути статистику по своїх хворих вкладці “Статистика”, див. рисунок 3.12.

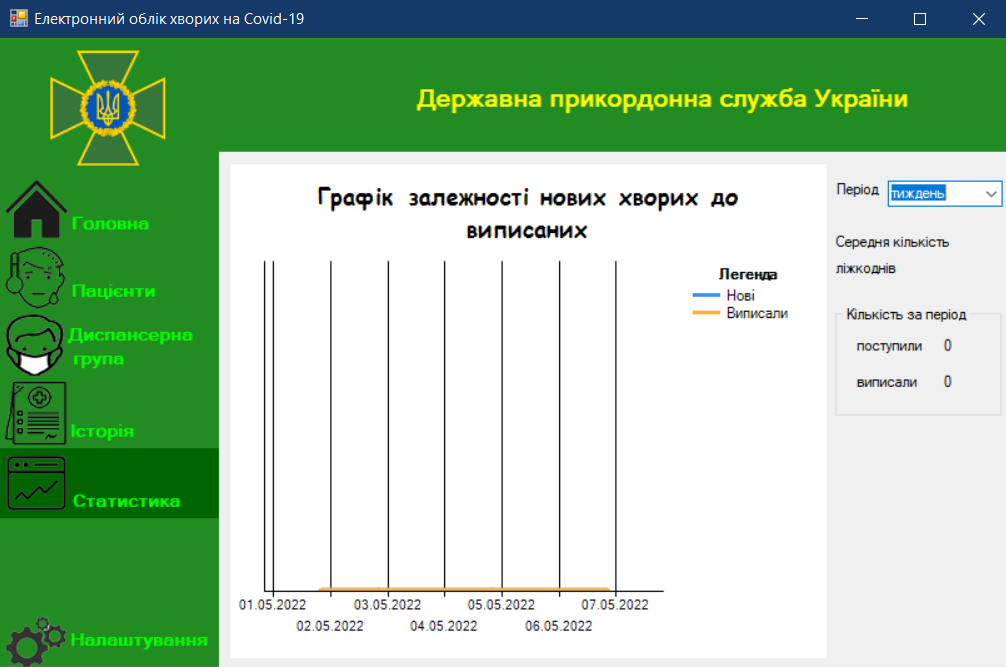
**

Рисунок 3.12 – Статистика

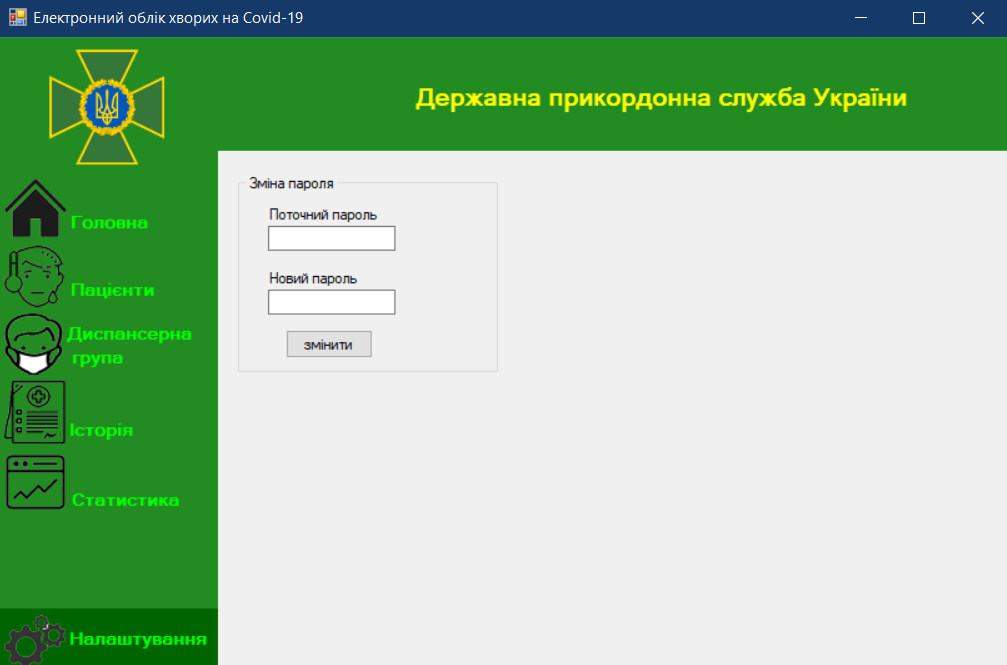
**

Рисунок 3.13 – Налаштування

Отже в даному підрозділі показано інтерфейс програмного забезбечення та інстукція з користування, завдяки розробленому інтервейсу з дотриманням вимог “ User Friendly ”, він є інтуєтивно зрозумілим для користувача.

**ВИСНОВКИ**

У ході створення курсового проекту було розроблено програму – “Облік ковід-хворих лікарем інфікціоністом”, на мові програмування C# під операційні системи сімейства Windows, з підключенням сторонніх розширень пакетів NuGet для роботи із документами Word. В програмі передбачена можливість перегляду кожного пацієнта, редагування його даних та статусу як пацієнта зі збереженням у файл запису, адміністрування доступу до даних та створено систему облікових записів.

Як і будь-який інших, даний програмний продукт, має ряд переваг та недоліків.

Переваги:

* + Інтуїтивно зрозумілий інтерфейс користувача;
  + Легкий перегляд всієї інформації;
  + Низькі вимоги до системи;
  + Зручний вивід інформації в електронну історію хвороби;
  + Швидке оновлення даних;
  + Швидкодія програмного продукту;
  + Зручне редагування;
  + ПЗ не потребує підключення персонального комп’ютера до мережі інтернет.

Недоліки:

* + ПЗ не передбачає віддаленого входу в систему, тобто воно є стаціонарним на кожному ПК;
  + Відсутня можливість синхронізації програм встановлених на різних ПК.

**Список використаних джерел**

1. Інтернет-ресурс для вивчення програмування [Електроний ресурс] – Режим доступу: <https://metanit.com/>
2. Об’єктно-орієнтоване програмування (C#) [Електроний ресурс] – Режим доступу: [https://docs.microsoft.com/ruru/dotnet/csharp/fundamentals/tutor ials/oop](https://docs.microsoft.com/ruru/dotnet/csharp/fundamentals/tutor%20ials/oop)
3. Youtube канал з уроками [Електроний ресурс] – Режим доступу: <https://www.youtube.com/watch?v=lg_D_-1KNBek>
4. Відео-урок «Поліморфізм на практиці» [Електроний ресурс] - Режим доступу: <https://www.youtube.com/watch?v=bKbVmXuh9wQ>
5. EHelsi, електронна система здоров’я [Електроний ресурс] - Режим доступу: <https://uk.wikipedia.org/wiki/Helsi>
6. Askep, електронна система здоров’я [Електроний ресурс] - Режим доступу: <https://askep.net/>
7. Інтернет-форум для пошуку вирішення запитань програмування [Електроний ресурс] – Режим доступу: <https://stackoverflow.com/>
8. Наслідування в C# и .NET [Електроний ресурс] – Режим доступу: <https://docs.microsoft.com/ru-ru/dotnet/csharp/fundamentals/tutorials/inheritance>
9. Зразок готового рішення “Doc.ua” [Електроний ресурс] – Режим доступу : <https://doc.ua/ua>
10. Зразок готового рішення “Helsi.me” [Електроний ресурс] – Режим доступу : <https://helsi.me/>
11. ДСТУ 3008-95 "Документація.  Звіти в сфері науки і техніки. Структура і правила оформлення" – Державний стандарт України.

**Додаток A**

**Setting.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using курсовий\_проект\_100.Forms;

namespace курсовий\_проект\_100.Forms

{

public partial class Setting : Form

{

Doctor doctor = new Doctor();

List<Doctor> doctors = new List<Doctor>();

public Setting()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

BinaryFormatter binaryFormatter0 = new BinaryFormatter();

using (FileStream fileStream = new FileStream("cookies.dat", FileMode.OpenOrCreate))

{

doctor = (Doctor)binaryFormatter0.Deserialize(fileStream);

}

}

catch { }

try

{

BinaryFormatter binaryFormatter1 = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Лікарі.dat", FileMode.OpenOrCreate))

{

doctors = (List<Doctor>)binaryFormatter1.Deserialize(fileStream);

}

}

catch { }

if (textBox1.Text == doctor.getPasswrd())

{

doctor.setPasswrd(textBox2.Text);

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

{

if (doctors[i].getID() == doctor.getID())

{

doctors.Remove(doctors[i]);

break;

}

}

doctors.Add(doctor);

MessageBox.Show("Пароль змінено", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Лікарі.dat", FileMode.Create))

binaryFormatter.Serialize(fileStream, doctors);

}

else MessageBox.Show("Попередній пароль не співпадає!!!", "Попередження", MessageBoxButtons.OK, MessageBoxIcon.Warning);

textBox1.Text = "";

textBox2.Text = "";

}

}

}

**Statystyka.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace курсовий\_проект\_100.Forms

{

public partial class Statystyka : Form

{

List<Hvoryi> hvoryis = new List<Hvoryi>();

List<Hvoryi> hvoryis\_D = new List<Hvoryi>();

Doctor doctor = new Doctor();

public Statystyka()

{

InitializeComponent();

UpDat\_cookies();

UpDat();

for(int i=0;i<hvoryis\_D.Count;i++)

{

if(hvoryis\_D[i].getIDdoc() == doctor.getID())

{

hvoryis.Add(hvoryis\_D[i]);

}

}

label4.Text = "";

comboBox1.SelectedIndex = 0;

}

private void UpDat()

{

hvoryis.Clear();

List<Hvoryi> hvoryis\_DD = new List<Hvoryi>();

try

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Хворі.dat", FileMode.OpenOrCreate))

{

hvoryis\_DD = (List<Hvoryi>)binaryFormatter.Deserialize(fileStream);

}

BinaryFormatter binaryFormatter1 = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Диспансерна\_група.dat", FileMode.OpenOrCreate))

{

hvoryis\_D = (List<Hvoryi>)binaryFormatter1.Deserialize(fileStream);

}

for(int i = 0; i < hvoryis\_DD.Count; i++)

{

hvoryis\_D.Add(hvoryis\_DD[i]);

}

}

catch { }

}

private void Statystyka\_Load(object sender, EventArgs e)

{

}

private void UpDat\_cookies()

{

try

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("cookies.dat", FileMode.OpenOrCreate))

{

doctor = (Doctor)binaryFormatter.Deserialize(fileStream);

}

}

catch { MessageBox.Show("Не вдалося підгрузити поточного лікаря!!!"); }

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

List<int> post = new List<int>();

List<int> vyp = new List<int>();

int p = 0, v = 0, kkk = 0, klDay = 0;

if (comboBox1.SelectedIndex == 0)

{

chart1.Series.Clear();

post.Clear();

vyp.Clear();

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

{

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 7) v++;

if ((DateTime.Now - hvoryis[i].getDateG()).Days < 7) p++;

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 7 && (DateTime.Now - hvoryis[i].getDateV()).Days < 7)

{

kkk++;

klDay += ((hvoryis[i].getDateV() - hvoryis[i].getDateG()).Days);

}

}

label7.Text = Convert.ToString(p);

label8.Text = Convert.ToString(v);

if (kkk != 0) label4.Text = Convert.ToString(klDay / kkk);

List<DateTime> dateTimes = new List<DateTime>();

for (int i = -5; i <= 0; i++)

{

dateTimes.Add(Convert.ToDateTime(DateTime.Now.AddDays(i)));

}

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

{

v = 0;

p = 0;

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

{

if (dateTimes[i].Date == hvoryis[j].getDateG().Date) p++;

if (dateTimes[i].Date == hvoryis[j].getDateV().Date) v++;

}

post.Add(p);

vyp.Add(v);

}

chart1.Series.Add("Нові");

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[0].BorderWidth = 5;

chart1.Series.Add("Виписали");

chart1.Series[1].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[1].BorderWidth = 5;

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

{

chart1.Series[0].Points.AddXY(dateTimes[i],post[i]);

chart1.Series[1].Points.AddXY(dateTimes[i],vyp[i]);

}

}

else if (comboBox1.SelectedIndex == 1)

{

chart1.Series.Clear();

post.Clear();

vyp.Clear();

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

{

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 31) v++;

if ((DateTime.Now - hvoryis[i].getDateG()).Days < 31) p++;

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 31 && (DateTime.Now - hvoryis[i].getDateV()).Days < 31)

{

kkk++;

klDay += ((hvoryis[i].getDateV() - hvoryis[i].getDateG()).Days);

}

}

label7.Text = Convert.ToString(p);

label8.Text = Convert.ToString(v);

if (kkk != 0) label4.Text = Convert.ToString(klDay / kkk);

List<DateTime> dateTimes = new List<DateTime>();

for (int i = -33; i <= 0; i++)

{

dateTimes.Add(Convert.ToDateTime(DateTime.Now.AddDays(i)));

}

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

{

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

{

if (dateTimes[i].Date == hvoryis[j].getDateG().Date) p++;

if (dateTimes[i].Date == hvoryis[j].getDateV().Date) v++;

}

if (i % 7 == 0)

{

post.Add(p);

vyp.Add(v);

v = 0;

p = 0;

}

}

post.Reverse();

vyp.Reverse();

chart1.Series.Add("Нові");

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[0].BorderWidth = 5;

chart1.Series.Add("Виписали");

chart1.Series[1].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[1].BorderWidth = 5;

for (int i = 0; i < 4; i++)

{

chart1.Series[0].Points.AddXY("Тиждень" + (i + 1), post[i]);

chart1.Series[1].Points.AddXY("Тиждень" + (i + 1), vyp[i]);

}

}

else if (comboBox1.SelectedIndex == 2)

{

chart1.Series.Clear();

post.Clear();

vyp.Clear();

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

{

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 92) v++;

if ((DateTime.Now - hvoryis[i].getDateG()).Days < 92) p++;

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 92 && (DateTime.Now - hvoryis[i].getDateV()).Days < 92)

{

kkk++;

klDay += ((hvoryis[i].getDateV() - hvoryis[i].getDateG()).Days);

}

}

label7.Text = Convert.ToString(p);

label8.Text = Convert.ToString(v);

if (kkk != 0) label4.Text = Convert.ToString(klDay / kkk);

List<DateTime> dateTimes = new List<DateTime>();

for (int i = -92; i <= 0; i++)

{

dateTimes.Add(Convert.ToDateTime(DateTime.Now.AddDays(i)));

}

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

{

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

{

if (dateTimes[i].Date == hvoryis[j].getDateG().Date) p++;

if (dateTimes[i].Date == hvoryis[j].getDateV().Date) v++;

}

if (i % 31 == 0)

{

post.Add(p);

vyp.Add(v);

v = 0;

p = 0;

}

}

post.Reverse();

vyp.Reverse();

chart1.Series.Add("Нові");

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[0].BorderWidth = 5;

chart1.Series.Add("Виписали");

chart1.Series[1].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[1].BorderWidth = 5;

for (int i = 0; i < 3; i++)

{

chart1.Series[0].Points.AddXY("Місяць " + (i + 1), post[i]);

chart1.Series[1].Points.AddXY("Місяць " + (i + 1), vyp[i]);

}

}

else if (comboBox1.SelectedIndex == 3)

{

chart1.Series.Clear();

post.Clear();

vyp.Clear();

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

{

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 182) v++;

if ((DateTime.Now - hvoryis[i].getDateG()).Days < 182) p++;

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 182 && (DateTime.Now - hvoryis[i].getDateV()).Days < 182)

{

kkk++;

klDay += ((hvoryis[i].getDateV() - hvoryis[i].getDateG()).Days);

}

}

label7.Text = Convert.ToString(p);

label8.Text = Convert.ToString(v);

if (kkk != 0) label4.Text = Convert.ToString(klDay / kkk);

List<DateTime> dateTimes = new List<DateTime>();

for (int i = -182; i <= 0; i++)

{

dateTimes.Add(Convert.ToDateTime(DateTime.Now.AddDays(i)));

}

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

{

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

{

if (dateTimes[i].Date == hvoryis[j].getDateG().Date) p++;

if (dateTimes[i].Date == hvoryis[j].getDateV().Date) v++;

}

if (i % 31 == 0)

{

post.Add(p);

vyp.Add(v);

v = 0;

p = 0;

}

}

post.Reverse();

vyp.Reverse();

chart1.Series.Add("Нові");

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[0].BorderWidth = 5;

chart1.Series.Add("Виписали");

chart1.Series[1].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[1].BorderWidth = 5;

for (int i = 0; i < 6; i++)

{

chart1.Series[0].Points.AddXY("Місяць " + (i + 1), post[i]);

chart1.Series[1].Points.AddXY("Місяць " + (i + 1), vyp[i]);

}

}

else if (comboBox1.SelectedIndex == 4)

{

chart1.Series.Clear();

post.Clear();

vyp.Clear();

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

{

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 365) v++;

if ((DateTime.Now - hvoryis[i].getDateG()).Days < 365) p++;

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 365 && (DateTime.Now - hvoryis[i].getDateV()).Days < 365)

{

kkk++;

klDay += ((hvoryis[i].getDateV() - hvoryis[i].getDateG()).Days);

}

}

label7.Text = Convert.ToString(p);

label8.Text = Convert.ToString(v);

if (kkk != 0) label4.Text = Convert.ToString(klDay / kkk);

List<DateTime> dateTimes = new List<DateTime>();

for (int i = -365; i <= 0; i++)

{

dateTimes.Add(Convert.ToDateTime(DateTime.Now.AddDays(i)));

}

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

{

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

{

if (dateTimes[i].Date == hvoryis[j].getDateG().Date) p++;

if (dateTimes[i].Date == hvoryis[j].getDateV().Date) v++;

}

if (i % 92 == 0)

{

post.Add(p);

vyp.Add(v);

v = 0;

p = 0;

}

}

post.Reverse();

vyp.Reverse();

chart1.Series.Add("Нові");

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[0].BorderWidth = 5;

chart1.Series.Add("Виписали");

chart1.Series[1].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[1].BorderWidth = 5;

for (int i = 0; i < 4; i++)

{

chart1.Series[0].Points.AddXY("Квартал " + (1 + i), post[i]);

chart1.Series[1].Points.AddXY("Квартал " + (1 + i), vyp[i]);

}

}

}

**Admin.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using курсовий\_проект\_100.Forms;

using Word = Microsoft.Office.Interop.Word;

namespace курсовий\_проект\_100

{

public partial class Admin : Form

{

private int ID;

List<Doctor> doctors = new List<Doctor>();

List<Hvoryi> hvoryis = new List<Hvoryi>();

private int curentH = 0;

public Admin()

{

this.Text = "Сторінка адміністратора (завідувача відділенням)";

InitializeComponent();

ClearPage1();

UpDat();

UpDatHV();

comboBox1.SelectedIndex = 0;

textBox3.Enabled = false;

textBox3.Visible = false;

dateTimePicker3.Enabled = false;

dateTimePicker3.Visible = false;

label41.Text = "";

comboBox7.SelectedIndex = 0;

textBox16.Text = "";

int o = doctors.Count - 1;

UpDat();

for (int i = 0, k = 0; i < hvoryis.Count; i++, k++)

{

DatSetToGrid(k,i);

}

try

{

Add\_DatInFormVisio(doctors[o]);

ID = doctors[o].getID() + 1;

}

catch

{

MessageBox.Show("Лікарі ще не додані, перший запуск програми");

ID = 1;

}

label10.Text = Convert.ToString(ID);

}

private void UpDatHV()

{

hvoryis.Clear();

List<Hvoryi> hvoryis\_D = new List<Hvoryi>();

try

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Хворі.dat", FileMode.OpenOrCreate))

{

hvoryis = (List<Hvoryi>)binaryFormatter.Deserialize(fileStream);

}

BinaryFormatter binaryFormatter1 = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Диспансерна\_група.dat", FileMode.OpenOrCreate))

{

hvoryis\_D = (List<Hvoryi>)binaryFormatter1.Deserialize(fileStream);

}

for (int i = 0; i < hvoryis\_D.Count; i++)

{

hvoryis.Add(hvoryis\_D[i]);

}

hvoryis\_D.Clear();

}

catch { }

}

private void Add\_DatInFormVisio(Doctor doc)

{

textBox10.Text = doc.getSurname();

textBox11.Text = doc.getName();

textBox9.Text = doc.getPatrn();

textBox12.Text = doc.getGromad();

textBox8.Text = doc.getHomeadres();

comboBox6.Text = doc.getDoc();

textBox7.Text = doc.getNumDoc();

dateTimePicker2.Value = doc.getDateB();

textBox6.Text = doc.getGkrow();

comboBox3.Text = doc.getRkrow();

textBox13.Text = doc.getStat();

textBox5.Text = doc.getKvalif();

textBox4.Text = doc.getPosada();

label19.Text = doc.getID().ToString();

}

private void DatSetToGrid(int k, int i)

{

dataGridView1.Rows.Add();

dataGridView1.Rows[k].Cells[0].Value = Convert.ToString(hvoryis[i].getID());

dataGridView1.Rows[k].Cells[1].Value = hvoryis[i].getSurname();

dataGridView1.Rows[k].Cells[2].Value = hvoryis[i].getName();

dataGridView1.Rows[k].Cells[3].Value = hvoryis[i].getPatrn();

dataGridView1.Rows[k].Cells[4].Value = hvoryis[i].getDateB().ToString("dd-MM-yyyy");

dataGridView1.Rows[k].Cells[5].Value = hvoryis[i].getDateG().ToString("dd-MM-yyyy");

dataGridView1.Rows[k].Cells[6].Value = hvoryis[i].getHomeadres();

dataGridView1.Rows[k].Cells[7].Value = hvoryis[i].getStat();

int kk = DateTime.Now.Year - hvoryis[i].getDateB().Year;

dataGridView1.Rows[k].Cells[8].Value = Convert.ToString(kk);

dataGridView1.Rows[k].Cells[9].Value = hvoryis[i].getGromad();

dataGridView1.Rows[k].Cells[10].Value = hvoryis[i].getDoc();

dataGridView1.Rows[k].Cells[11].Value = hvoryis[i].getNumDoc();

dataGridView1.Rows[k].Cells[12].Value = hvoryis[i].getGkrow();

dataGridView1.Rows[k].Cells[13].Value = hvoryis[i].getRkrow();

dataGridView1.Rows[k].Cells[14].Value = hvoryis[i].getDiagnozKlin();

dataGridView1.Rows[k].Cells[15].Value = hvoryis[i].getDiagnozHospital();

dataGridView1.Rows[k].Cells[16].Value = hvoryis[i].getKymNaprav();

dataGridView1.Rows[k].Cells[17].Value = hvoryis[i].getNumP();

if (hvoryis[k].getDateV() == DateTime.MinValue)

{

dataGridView1.Rows[k].Cells[18].Value = "";

dataGridView1.Rows[k].Cells[19].Value = "";

}

else

{

dataGridView1.Rows[k].Cells[18].Value = hvoryis[i].getDateV().ToString("dd-MM-yyyy");

dataGridView1.Rows[k].Cells[19].Value = Convert.ToString((hvoryis[i].getDateV() - hvoryis[i].getDateG()).Days);

}

}

private void UpDat()

{

doctors.Clear();

try

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Лікарі.dat", FileMode.OpenOrCreate))

{

doctors = (List<Doctor>)binaryFormatter.Deserialize(fileStream);

}

}

catch { }

}

private void Admin\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void tabPage1\_Click(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBoxLog.Text == "" || textBoxPosada.Text == "" || textBoxPass.Text == "" || textBoxSurN.Text == "" || textBoxNam.Text == "" || textBoxPatrn.Text == "" || textBoxHome.Text == "" || textBoxNumDoc.Text == "" || textBoxKval.Text == "" || textBox2.Text == "")

{

MessageBox.Show("Заповніть усі поля!!!");

}

else

{

for(int l = 0; l<doctors.Count;l++)

{

if(doctors[l].getLogin() == textBoxLog.Text)

{

MessageBox.Show("Такий логін уже існує");

break;

}

}

doctors.Add(createDoc());

writeAll();

MessageBox.Show("Лікар доданий");

UpDat();

ClearPage1();

ID++;

label10.Text = ID.ToString();

}

}

private void ClearPage1()

{

textBox2.Text = "";

textBoxHome.Text = "";

textBoxKval.Text = "";

textBoxLog.Text = "";

textBoxNam.Text = "";

textBoxNumDoc.Text = "";

textBoxPass.Text = "";

textBoxPatrn.Text = "";

textBoxPosada.Text = "";

textBoxSurN.Text = "";

comboBox1.SelectedIndex = 0;

comboBox2.SelectedIndex = 0;

comboBox5.SelectedIndex = 0;

radioButton1.Checked = true;

}

private void writeAll()

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Лікарі.dat", FileMode.OpenOrCreate))

binaryFormatter.Serialize(fileStream, doctors);

}

private Doctor createDoc()

{

Doctor doctor = new Doctor();

doctor.setDateB(dateTimePicker1.Value);

doctor.setDoc(comboBox2.Text);

doctor.setGkrow(textBox2.Text);

doctor.setGromad(comboBox5.Text);

doctor.setHomeadres(textBoxHome.Text);

doctor.setID(Convert.ToInt32(label10.Text));

doctor.setKvalif(textBoxKval.Text);

doctor.setLogin(textBoxLog.Text);

doctor.setName(textBoxNam.Text);

doctor.setNumDoc(textBoxNumDoc.Text);

doctor.setPasswrd(textBoxPass.Text);

doctor.setPatrn(textBoxPatrn.Text);

doctor.setPosada(textBoxPosada.Text);

doctor.setRkrow(comboBox1.Text);

if (radioButton1.Checked)

doctor.setStat(radioButton1.Text);

else doctor.setStat(radioButton2.Text);

doctor.setSurname(textBoxSurN.Text);

return doctor;

}

private void button2\_Click(object sender, EventArgs e)

{

try

{

if (curentH == doctors.Count - 1)

{

curentH = 0;

Add\_DatInFormVisio(doctors[curentH]);

}

else

{

curentH++;

Add\_DatInFormVisio(doctors[curentH]);

}

}

catch { }

}

private void button3\_Click(object sender, EventArgs e)

{

try

{

if (curentH == 0)

{

curentH = doctors.Count - 1;

Add\_DatInFormVisio(doctors[curentH]);

}

else

{

curentH--;

Add\_DatInFormVisio(doctors[curentH]);

}

}

catch { }

}

private void button4\_Click(object sender, EventArgs e)

{

string x1;

bool posh = false;

int k = 0;

string r = "\n";

if (textBox14.Text == "" && textBox1.Text == "") MessageBox.Show("Введіть дані для пошуку");

else if (textBox14.Text != "" && textBox1.Text == "")

{

try

{

int pp = Convert.ToInt32(textBox14.Text) - 1;

Add\_DatInFormVisio(doctors[pp]);

textBox1.Text = "";

textBox14.Text = "";

}

catch

{

MessageBox.Show("Мабуть такого пацієнта не існує :((");

textBox1.Text = "";

textBox14.Text = "";

}

}

else

{

try

{

x1 = Convert.ToString(textBox1.Text);

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

{

if (doctors[i].getSurname() == x1)

{

r += (i + 1).ToString() + "-" + doctors[i].getSurname() + " " + doctors[i].getName() + "\n";

k++;

posh = true;

}

}

if (!posh)

{

MessageBox.Show("Такого запису не знайдено", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

MessageBox.Show("Кількість записів = " + k + r, "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

textBox1.Text = "";

textBox14.Text = "";

}

catch

{

MessageBox.Show("Схоже виникла помилка спробуйте пізніше");

textBox1.Text = "";

textBox14.Text = "";

}

}

}

private void button5\_Click(object sender, EventArgs e)

{

if (doctors.Count == 0) MessageBox.Show("Лікарі ще не додані","Попередження",MessageBoxButtons.OK,MessageBoxIcon.Warning);

else if (MessageBox.Show("Ви справді бажаєте видалити лікаря цього лікаря?", "Видалення",MessageBoxButtons.YesNo,MessageBoxIcon.Question) == DialogResult.Yes)

{

doctors.Remove(doctors[curentH]);

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Лікарі.dat", FileMode.Create))

binaryFormatter.Serialize(fileStream, doctors);

}

}

private void comboBox4\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox4.SelectedIndex == 0 || comboBox4.SelectedIndex == 1 || comboBox4.SelectedIndex == 2 || comboBox4.SelectedIndex == 3 || comboBox4.SelectedIndex == 4 || comboBox4.SelectedIndex == 5)

{

textBox3.Enabled = true;

textBox3.Visible = true;

dateTimePicker3.Enabled = false;

dateTimePicker3.Visible = false;

}

else

{

textBox3.Enabled = false;

textBox3.Visible = false;

dateTimePicker3.Enabled = true;

dateTimePicker3.Visible = true;

}

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

string t;

int kk;

if (comboBox4.SelectedIndex == 0)

{

kk = 0;

t = textBox3.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getSurname().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk,i);

kk++;

}

}

}

else if (comboBox4.SelectedIndex == 1)

{

kk = 0;

t = textBox3.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getName().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox4.SelectedIndex == 2)

{

kk = 0;

t = textBox3.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getHomeadres().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox4.SelectedIndex == 3)

{

kk = 0;

t = textBox3.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getStat().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox4.SelectedIndex == 4)

{

kk = 0;

t = textBox3.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getGromad().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox4.SelectedIndex == 5)

{

kk = 0;

t = textBox3.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getNumDoc().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

}

private void dateTimePicker3\_ValueChanged(object sender, EventArgs e)

{

int kk;

if (comboBox4.SelectedIndex == 6)

{

kk = 0;

dataGridView1.Rows.Clear();

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

{

if (dateTimePicker3.Value.Date == hvoryis[i].getDateG().Date)

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox4.SelectedIndex == 7)

{

kk = 0;

dataGridView1.Rows.Clear();

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

{

if (dateTimePicker3.Value.Date == hvoryis[i].getDateV().Date)

{

DatSetToGrid(kk, i);

kk++;

}

}

}

}

private readonly string TempleteFileName = "D:\\Visual Studio2019\\3 курс\\курсовий\_проект\_100\\bin\\Debug\\Історіяя.docx";

private void ReplaceWordStud(string stubToReplace, string text, Word.Document wordDocument)

{

var range = wordDocument.Content;

range.Find.ClearFormatting();

range.Find.Execute(FindText: stubToReplace, ReplaceWith: text);

}

private void button6\_Click(object sender, EventArgs e)

{

if (textBox15.Text == "")

{

MessageBox.Show("Введіть ID пацієнта");

}

else

{

int k = Convert.ToInt32(textBox15.Text);

for (int i = 0; i < dataGridView1.RowCount; i++)

{

if (k == Convert.ToInt32(dataGridView1.Rows[i].Cells[0].Value))

{

MessageBox.Show("Зачекайте відриття Word, ваш запит на виконанні", "Процес", MessageBoxButtons.OK, MessageBoxIcon.Information);

var wordApp = new Word.Application();

wordApp.Visible = false;

var wordDocument = wordApp.Documents.Open(TempleteFileName);

try

{

ReplaceWordStud("{ID}", Convert.ToString(dataGridView1.Rows[i].Cells[0].Value), wordDocument);

ReplaceWordStud("{DateB}", Convert.ToString(dataGridView1.Rows[i].Cells[4].Value), wordDocument);

ReplaceWordStud("{DateG}", Convert.ToString(dataGridView1.Rows[i].Cells[5].Value), wordDocument);

ReplaceWordStud("{Stat}", Convert.ToString(dataGridView1.Rows[i].Cells[7].Value), wordDocument);

ReplaceWordStud("{OldYear}", Convert.ToString(dataGridView1.Rows[i].Cells[8].Value), wordDocument);

ReplaceWordStud("{Home}", Convert.ToString(dataGridView1.Rows[i].Cells[6].Value), wordDocument);

ReplaceWordStud("{Gromadyanstvo}", Convert.ToString(dataGridView1.Rows[i].Cells[9].Value), wordDocument);

ReplaceWordStud("{SurnameP}", Convert.ToString(dataGridView1.Rows[i].Cells[1].Value), wordDocument);

ReplaceWordStud("{NameP}", Convert.ToString(dataGridView1.Rows[i].Cells[2].Value), wordDocument);

ReplaceWordStud("{Patron}", Convert.ToString(dataGridView1.Rows[i].Cells[3].Value), wordDocument);

ReplaceWordStud("{TypeDoc}", Convert.ToString(dataGridView1.Rows[i].Cells[10].Value), wordDocument);

ReplaceWordStud("{NumDoc}", Convert.ToString(dataGridView1.Rows[i].Cells[11].Value), wordDocument);

ReplaceWordStud("{KymNaprav}", Convert.ToString(dataGridView1.Rows[i].Cells[16].Value), wordDocument);

ReplaceWordStud("{Dgoz}", Convert.ToString(dataGridView1.Rows[i].Cells[15].Value), wordDocument);

ReplaceWordStud("{Dklin}", Convert.ToString(dataGridView1.Rows[i].Cells[14].Value), wordDocument);

ReplaceWordStud("{Gk}", Convert.ToString(dataGridView1.Rows[i].Cells[12].Value), wordDocument);

ReplaceWordStud("{Rk}", Convert.ToString(dataGridView1.Rows[i].Cells[13].Value), wordDocument);

ReplaceWordStud("{Dv}", Convert.ToString(dataGridView1.Rows[i].Cells[18].Value), wordDocument);

ReplaceWordStud("{KlDay}", Convert.ToString(dataGridView1.Rows[i].Cells[19].Value), wordDocument);

wordDocument.SaveAs2(@"D:\result.docx");

wordApp.Visible = true;

break;

}

catch

{

MessageBox.Show("Виникла помилка!");

}

}

}

}

}

private void label36\_Click(object sender, EventArgs e)

{

}

private void button7\_Click(object sender, EventArgs e)

{

bool posh = false;

int k = 0;

string r = "\n";

if (textBox16.Text == "") MessageBox.Show("Введіть дані для пошуку");

else

{

try

{

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

{

if (doctors[i].getSurname().ToLower().Contains(textBox16.Text.ToLower()))

{

r += (i + 1).ToString() + "-" + doctors[i].getSurname() + " " + doctors[i].getName() + "\n";

k++;

posh = true;

}

}

if (!posh)

{

MessageBox.Show("Такого лікаря не знайдено", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

MessageBox.Show("Кількість лікарів що підходять = " + k + r, "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

catch

{

MessageBox.Show("Схоже виникла помилка спробуйте пізніше");

}

}

}

private void tabPage3\_Click(object sender, EventArgs e)

{

}

private void button8\_Click(object sender, EventArgs e)

{

int x = Convert.ToInt32(textBox17.Text), kk=0;

dataGridView1.Rows.Clear();

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

{

if(hvoryis[i].getIDdoc() == x)

{

DatSetToGrid(kk, i);

kk++;

}

}

if (dataGridView1.Rows.Count < 1)

{

MessageBox.Show("Схоже що в цього лікаря ще ніхто не лікується", "Зауваження", MessageBoxButtons.OK, MessageBoxIcon.Warning);

for (int i = 0; i < hvoryis.Count; i++, kk++)

{

DatSetToGrid(kk, i);

}

}

}

private void comboBox7\_SelectedIndexChanged(object sender, EventArgs e)

{

List<int> post = new List<int>();

List<int> vyp = new List<int>();

int p = 0, v = 0, kkk = 0, klDay = 0;

if (comboBox7.SelectedIndex == 0)

{

chart1.Series.Clear();

post.Clear();

vyp.Clear();

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

{

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 7) v++;

if ((DateTime.Now - hvoryis[i].getDateG()).Days < 7) p++;

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 7 && (DateTime.Now - hvoryis[i].getDateV()).Days < 7)

{

kkk++;

klDay += ((hvoryis[i].getDateV() - hvoryis[i].getDateG()).Days);

}

}

label39.Text = Convert.ToString(p);

label38.Text = Convert.ToString(v);

if(kkk!=0) label42.Text = Convert.ToString(klDay / kkk);

List<DateTime> dateTimes = new List<DateTime>();

for (int i = -5; i <= 0; i++)

{

dateTimes.Add(Convert.ToDateTime(DateTime.Now.AddDays(i)));

}

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

{

v = 0;

p = 0;

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

{

if (dateTimes[i].Date == hvoryis[j].getDateG().Date) p++;

if (dateTimes[i].Date == hvoryis[j].getDateV().Date) v++;

}

post.Add(p);

vyp.Add(v);

}

post.Reverse();

vyp.Reverse();

chart1.Series.Add("Нові");

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[0].BorderWidth = 5;

chart1.Series.Add("Виписали");

chart1.Series[1].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[1].BorderWidth = 5;

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

{

chart1.Series[0].Points.AddXY(dateTimes[i], post[i]);

chart1.Series[1].Points.AddXY(dateTimes[i], vyp[i]);

}

}

else if (comboBox7.SelectedIndex == 1)

{

chart1.Series.Clear();

post.Clear();

vyp.Clear();

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

{

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 31) v++;

if ((DateTime.Now - hvoryis[i].getDateG()).Days < 31) p++;

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 31 && (DateTime.Now - hvoryis[i].getDateV()).Days < 31)

{

kkk++;

klDay += ((hvoryis[i].getDateV() - hvoryis[i].getDateG()).Days);

}

}

label39.Text = Convert.ToString(p);

label38.Text = Convert.ToString(v);

if (kkk != 0) label42.Text = Convert.ToString(klDay / kkk);

List<DateTime> dateTimes = new List<DateTime>();

for (int i = -33; i <= 0; i++)

{

dateTimes.Add(Convert.ToDateTime(DateTime.Now.AddDays(i)));

}

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

{

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

{

if (dateTimes[i].Date == hvoryis[j].getDateG().Date) p++;

if (dateTimes[i].Date == hvoryis[j].getDateV().Date) v++;

}

if (i % 7 == 0)

{

post.Add(p);

vyp.Add(v);

v = 0;

p = 0;

}

}

post.Reverse();

vyp.Reverse();

chart1.Series.Add("Нові");

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[0].BorderWidth = 5;

chart1.Series.Add("Виписали");

chart1.Series[1].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[1].BorderWidth = 5;

for (int i = 0; i < 4; i++)

{

chart1.Series[0].Points.AddXY("Тиждень" + (i + 1), post[i]);

chart1.Series[1].Points.AddXY("Тиждень" + (i + 1), vyp[i]);

}

}

else if (comboBox7.SelectedIndex == 2)

{

chart1.Series.Clear();

post.Clear();

vyp.Clear();

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

{

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 92) v++;

if ((DateTime.Now - hvoryis[i].getDateG()).Days < 92) p++;

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 92 && (DateTime.Now - hvoryis[i].getDateV()).Days < 92)

{

kkk++;

klDay += ((hvoryis[i].getDateV() - hvoryis[i].getDateG()).Days);

}

}

label39.Text = Convert.ToString(p);

label38.Text = Convert.ToString(v);

if (kkk != 0) label42.Text = Convert.ToString(klDay / kkk);

List<DateTime> dateTimes = new List<DateTime>();

for (int i = -92; i <= 0; i++)

{

dateTimes.Add(Convert.ToDateTime(DateTime.Now.AddDays(i)));

}

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

{

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

{

if (dateTimes[i].Date == hvoryis[j].getDateG().Date) p++;

if (dateTimes[i].Date == hvoryis[j].getDateV().Date) v++;

}

if (i % 31 == 0)

{

post.Add(p);

vyp.Add(v);

v = 0;

p = 0;

}

}

post.Reverse();

vyp.Reverse();

chart1.Series.Add("Нові");

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[0].BorderWidth = 5;

chart1.Series.Add("Виписали");

chart1.Series[1].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[1].BorderWidth = 5;

for (int i = 0; i < 3; i++)

{

chart1.Series[0].Points.AddXY("Місяць " + (i + 1), post[i]);

chart1.Series[1].Points.AddXY("Місяць " + (i + 1), vyp[i]);

}

}

else if (comboBox7.SelectedIndex == 3)

{

chart1.Series.Clear();

post.Clear();

vyp.Clear();

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

{

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 182) v++;

if ((DateTime.Now - hvoryis[i].getDateG()).Days < 182) p++;

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 182 && (DateTime.Now - hvoryis[i].getDateV()).Days < 182)

{

kkk++;

klDay += ((hvoryis[i].getDateV() - hvoryis[i].getDateG()).Days);

}

}

label39.Text = Convert.ToString(p);

label38.Text = Convert.ToString(v);

if (kkk != 0) label42.Text = Convert.ToString(klDay / kkk);

List<DateTime> dateTimes = new List<DateTime>();

for (int i = -182; i <= 0; i++)

{

dateTimes.Add(Convert.ToDateTime(DateTime.Now.AddDays(i)));

}

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

{

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

{

if (dateTimes[i].Date == hvoryis[j].getDateG().Date) p++;

if (dateTimes[i].Date == hvoryis[j].getDateV().Date) v++;

}

if (i % 31 == 0)

{

post.Add(p);

vyp.Add(v);

v = 0;

p = 0;

}

}

post.Reverse();

vyp.Reverse();

chart1.Series.Add("Нові");

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[0].BorderWidth = 5;

chart1.Series.Add("Виписали");

chart1.Series[1].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[1].BorderWidth = 5;

for (int i = 0; i < 6; i++)

{

chart1.Series[0].Points.AddXY("Місяць " + (i + 1), post[i]);

chart1.Series[1].Points.AddXY("Місяць " + (i + 1), vyp[i]);

}

}

else if (comboBox7.SelectedIndex == 4)

{

chart1.Series.Clear();

post.Clear();

vyp.Clear();

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

{

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 365) v++;

if ((DateTime.Now - hvoryis[i].getDateG()).Days < 365) p++;

if ((DateTime.Now - hvoryis[i].getDateV()).Days < 365 && (DateTime.Now - hvoryis[i].getDateV()).Days < 365)

{

kkk++;

klDay += ((hvoryis[i].getDateV() - hvoryis[i].getDateG()).Days);

}

}

label39.Text = Convert.ToString(p);

label38.Text = Convert.ToString(v);

if (kkk != 0) label42.Text = Convert.ToString(klDay / kkk);

List<DateTime> dateTimes = new List<DateTime>();

for (int i = -365; i <= 0; i++)

{

dateTimes.Add(Convert.ToDateTime(DateTime.Now.AddDays(i)));

}

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

{

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

{

if (dateTimes[i].Date == hvoryis[j].getDateG().Date) p++;

if (dateTimes[i].Date == hvoryis[j].getDateV().Date) v++;

}

if (i % 92 == 0)

{

post.Add(p);

vyp.Add(v);

v = 0;

p = 0;

}

}

post.Reverse();

vyp.Reverse();

chart1.Series.Add("Нові");

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[0].BorderWidth = 5;

chart1.Series.Add("Виписали");

chart1.Series[1].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

chart1.Series[1].BorderWidth = 5;

for (int i = 0; i < 4; i++)

{

chart1.Series[0].Points.AddXY("Квартал " + (4 - i), post[i]);

chart1.Series[1].Points.AddXY("Квартал " + (4 - i), vyp[i]);

}

}

}

private void textBox15\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[0-9]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox16\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я іІ їЇ ' ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox17\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[0-9]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void button9\_Click(object sender, EventArgs e)

{

if(MessageBox.Show("Ви справді бажаєте НЕЗВОРОТНЬО видалити базу з пацієнтами???","КРАЙНЄ ЗАПИТАННЯ",MessageBoxButtons.YesNoCancel,MessageBoxIcon.Warning)== DialogResult.Yes)

{

hvoryis.Clear();

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Хворі.dat", FileMode.Create))

binaryFormatter.Serialize(fileStream, hvoryis);

BinaryFormatter binaryFormatter1 = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Диспансерна\_група.dat", FileMode.Create))

binaryFormatter1.Serialize(fileStream, hvoryis);

dataGridView1.Rows.Clear();

}

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я іІ їЇ ' ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox14\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[0-9]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBoxSurN\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я іІ їЇ ' ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBoxNam\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я іІ їЇ ' ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBoxPatrn\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я іІ їЇ ' ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void comboBox5\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я іІ їЇ ' ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBoxHome\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я іІ їЇ ' , 0-9 . ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBoxKval\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я іІ їЇ ' 0-9 ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBoxPosada\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я іІ їЇ ' ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[0-9]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void comboBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[+-]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void comboBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я іІ їЇ ' ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBoxNumDoc\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[A-Z 0-9]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

}

}

**Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using курсовий\_проект\_100.Forms;

namespace курсовий\_проект\_100

{

public partial class Form1 : Form

{

List<Doctor> doctors = new List<Doctor>();

public Form1()

{

InitializeComponent();

textBox1.Text = "";

this.Text = "";

textBox2.Text = "";

textBox2.PasswordChar = '\*';

}

private void UpDat()

{

try

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Лікарі.dat", FileMode.OpenOrCreate))

{

doctors = (List<Doctor>)binaryFormatter.Deserialize(fileStream);

}

}

catch { }

}

private void button1\_Click(object sender, EventArgs e)

{

UpDat();

bool tt = true;

if (textBox1.Text == "admin" && textBox2.Text == "admin1111")

{

Admin admin = new Admin();

admin.Show();

this.Hide();

}

else

{

for (int l = 0; l < doctors.Count; l++)

{

if (textBox1.Text == doctors[l].getLogin() && textBox2.Text == doctors[l].getPasswrd())

{

tt = false;

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("cookies.dat", FileMode.Create))

binaryFormatter.Serialize(fileStream, doctors[l]);

Main a = new Main();

a.Show();

this.Hide();

break;

}

}

if(tt)

MessageBox.Show("Неправильний логін або пароль");

}

}

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

if (textBox2.PasswordChar == '\*')

{

textBox2.BringToFront();

textBox2.PasswordChar = '\0';

linkLabel1.Text = "сховати";

}

else

{

textBox2.PasswordChar = '\*';

linkLabel1.Text = "показати пароль";

}

}

}

}

**Main.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 курсовий\_проект\_100.Forms;

namespace курсовий\_проект\_100

{

public partial class Main : Form

{

private Form activeForm;

public Main()

{

InitializeComponent();

button1.Text = "Головна";

button2.Text = "Пацієнти";

button3.Text = "Диспансерна" + "\n" + " група";

button4.Text = "Історія";

button5.Text = "Статистика";

button6.Text = "Налаштування";

this.Text = "Електронний облік хворих на Covid-19";

label1.Text = "Державна прикордонна служба України";

button1.BackColor = Color.FromArgb(0, 100, 0);

button2.BackColor = Color.FromArgb(34, 139, 34);

button3.BackColor = Color.FromArgb(34, 139, 34);

button4.BackColor = Color.FromArgb(34, 139, 34);

button5.BackColor = Color.FromArgb(34, 139, 34);

button6.BackColor = Color.FromArgb(34, 139, 34);

OpenChildForm(new Forms.Home());

}

private void OpenChildForm(Form childForm)

{

if (activeForm != null)

activeForm.Close();

activeForm = childForm;

childForm.TopLevel = false;

childForm.FormBorderStyle = FormBorderStyle.None;

childForm.Dock = DockStyle.Fill;

this.panel4.Controls.Add(childForm);

this.panel4.Tag = childForm;

childForm.BringToFront();

childForm.Show();

}

private void button1\_Click(object sender, EventArgs e)

{

button1.BackColor = Color.FromArgb(0, 100, 0);

button2.BackColor = Color.FromArgb(34, 139, 34);

button3.BackColor = Color.FromArgb(34, 139, 34);

button4.BackColor = Color.FromArgb(34, 139, 34);

button5.BackColor = Color.FromArgb(34, 139, 34);

button6.BackColor = Color.FromArgb(34, 139, 34);

OpenChildForm(new Forms.Home());

}

private void button2\_Click(object sender, EventArgs e)

{

button1.BackColor = Color.FromArgb(34, 139, 34);

button2.BackColor = Color.FromArgb(0, 100, 0);

button3.BackColor = Color.FromArgb(34, 139, 34);

button4.BackColor = Color.FromArgb(34, 139, 34);

button5.BackColor = Color.FromArgb(34, 139, 34);

button6.BackColor = Color.FromArgb(34, 139, 34);

OpenChildForm(new Forms.Add());

}

private void button3\_Click(object sender, EventArgs e)

{

button1.BackColor = Color.FromArgb(34, 139, 34);

button2.BackColor = Color.FromArgb(34, 139, 34);

button3.BackColor = Color.FromArgb(0, 100, 0);

button4.BackColor = Color.FromArgb(34, 139, 34);

button5.BackColor = Color.FromArgb(34, 139, 34);

button6.BackColor = Color.FromArgb(34, 139, 34);

OpenChildForm(new Forms.Despancer());

}

private void button4\_Click(object sender, EventArgs e)

{

button1.BackColor = Color.FromArgb(34, 139, 34);

button2.BackColor = Color.FromArgb(34, 139, 34);

button3.BackColor = Color.FromArgb(34, 139, 34);

button4.BackColor = Color.FromArgb(0, 100, 0);

button5.BackColor = Color.FromArgb(34, 139, 34);

button6.BackColor = Color.FromArgb(34, 139, 34);

OpenChildForm(new Forms.History());

}

private void button5\_Click(object sender, EventArgs e)

{

button1.BackColor = Color.FromArgb(34, 139, 34);

button2.BackColor = Color.FromArgb(34, 139, 34);

button3.BackColor = Color.FromArgb(34, 139, 34);

button4.BackColor = Color.FromArgb(34, 139, 34);

button5.BackColor = Color.FromArgb(0, 100, 0);

button6.BackColor = Color.FromArgb(34, 139, 34);

OpenChildForm(new Forms.Statystyka());

}

private void button6\_Click(object sender, EventArgs e)

{

button1.BackColor = Color.FromArgb(34, 139, 34);

button2.BackColor = Color.FromArgb(34, 139, 34);

button3.BackColor = Color.FromArgb(34, 139, 34);

button4.BackColor = Color.FromArgb(34, 139, 34);

button5.BackColor = Color.FromArgb(34, 139, 34);

button6.BackColor = Color.FromArgb(0, 100, 0);

OpenChildForm(new Forms.Setting());

}

private void Main\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

}

}

**People.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace курсовий\_проект\_100.Forms

{

[Serializable]

public class People

{

private string name;

private string surname;

private string patrn;

private DateTime dateB;

private string homeadres;

private string gromad;

private string Gkrow;

private string Rkrow;

private string doc;

private string numDoc;

public People()

{

}

public void setDateB(DateTime a)

{

dateB = a;

}

public DateTime getDateB()

{

return dateB;

}

public void setNumDoc(string a)

{

numDoc = a;

}

public string getNumDoc()

{

return numDoc;

}

public void setDoc(string a)

{

doc = a;

}

public string getDoc()

{

return doc;

}

public void setRkrow(string a)

{

Rkrow = a;

}

public string getRkrow()

{

return Rkrow;

}

public void setGkrow(string a)

{

Gkrow = a;

}

public string getGkrow()

{

return Gkrow;

}

public void setName(string a)

{

name = a;

}

public string getName()

{

return name;

}

public void setSurname(string a)

{

surname = a;

}

public string getSurname()

{

return surname;

}

public void setPatrn(string a)

{

patrn = a;

}

public string getPatrn()

{

return patrn;

}

public void setHomeadres(string a)

{

homeadres = a;

}

public string getHomeadres()

{

return homeadres;

}

public void setGromad(string a)

{

gromad = a;

}

public string getGromad()

{

return gromad;

}

}

}

**Hvoryi.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace курсовий\_проект\_100.Forms

{

[Serializable]

public class Hvoryi : People

{

private int ID;

private DateTime dateG;

private DateTime dateVypusky;

private string stat;

private string diagnozKlin;

private string diagnozHospital;

private string kymNaprav;

private string NumP;

private int ID\_doc;

public Hvoryi()

{

}

public void setDateV(DateTime a)

{

dateVypusky = a;

}

public DateTime getDateV()

{

return dateVypusky;

}

public void setIDdoc(int a)

{

ID\_doc = a;

}

public int getIDdoc()

{

return ID\_doc;

}

public void setID(int a)

{

ID = a;

}

public int getID()

{

return ID;

}

public void setDateG(DateTime a)

{

dateG = a;

}

public DateTime getDateG()

{

return dateG;

}

public void setNumP(string a)

{

NumP = a;

}

public string getNumP()

{

return NumP;

}

public void setStat(string a)

{

stat = a;

}

public string getStat()

{

return stat;

}

public void setDiagnozKlin(string a)

{

diagnozKlin = a;

}

public string getDiagnozKlin()

{

return diagnozKlin;

}

public void setDiagnozHospital(string a)

{

diagnozHospital = a;

}

public string getDiagnozHospital()

{

return diagnozHospital;

}

public void setKymNaprav(string a)

{

kymNaprav = a;

}

public string getKymNaprav()

{

return kymNaprav;

}

}

}

**Home.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace курсовий\_проект\_100.Forms

{

public partial class Home : Form

{

Doctor doctor = new Doctor();

public Home()

{

InitializeComponent();

UpDat\_cookies();

label2.Text = doctor.getSurname() + " " + doctor.getName();

}

private void UpDat\_cookies()

{

try

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("cookies.dat", FileMode.OpenOrCreate))

{

doctor = (Doctor)binaryFormatter.Deserialize(fileStream);

}

}

catch { MessageBox.Show("Не вдалося підгрузити поточного лікаря!!!"); }

}

}

}

**History.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using Word = Microsoft.Office.Interop.Word;

namespace курсовий\_проект\_100.Forms

{

public partial class History : Form

{

List<Hvoryi> hvoryis = new List<Hvoryi>();

Doctor doctor = new Doctor();

public History()

{

InitializeComponent();

UpDat\_cookies();

textBox2.Enabled = false;

textBox2.Visible = false;

dateTimePicker1.Enabled = false;

dateTimePicker1.Visible = false;

UpDat();

int kk = 0;

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

{

if (hvoryis[i].getIDdoc() == doctor.getID())

{

DatSetToGrid(kk, i);

kk++;

}

}

}

private readonly string TempleteFileName = "D:\\Visual Studio2019\\3 курс\\курсовий\_проект\_100\\bin\\Debug\\Історіяя.docx";

private void UpDat\_cookies()

{

try

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("cookies.dat", FileMode.OpenOrCreate))

{

doctor = (Doctor)binaryFormatter.Deserialize(fileStream);

}

}

catch { MessageBox.Show("Не вдалося підгрузити поточного лікаря!!!"); }

}

private void UpDat()

{

hvoryis.Clear();

try

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Хворі.dat", FileMode.OpenOrCreate))

{

hvoryis = (List<Hvoryi>)binaryFormatter.Deserialize(fileStream);

}

}

catch { }

}

private void ReplaceWordStud(string stubToReplace, string text, Word.Document wordDocument)

{

var range = wordDocument.Content;

range.Find.ClearFormatting();

range.Find.Execute(FindText: stubToReplace, ReplaceWith: text);

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "")

{

MessageBox.Show("Введіть ID пацієнта");

}

else

{

int k = Convert.ToInt32(textBox1.Text);

for (int i = 0; i < dataGridView1.RowCount; i++)

{

if (k == Convert.ToInt32(dataGridView1.Rows[i].Cells[0].Value))

{

MessageBox.Show("Зачекайте відриття Word, ваш запит на виконанні", "Процес", MessageBoxButtons.OK, MessageBoxIcon.Information);

var wordApp = new Word.Application();

wordApp.Visible = false;

var wordDocument = wordApp.Documents.Open(TempleteFileName);

try

{

ReplaceWordStud("{ID}", Convert.ToString(dataGridView1.Rows[i].Cells[0].Value), wordDocument);

ReplaceWordStud("{DateB}", Convert.ToString(dataGridView1.Rows[i].Cells[4].Value), wordDocument);

ReplaceWordStud("{DateG}", Convert.ToString(dataGridView1.Rows[i].Cells[5].Value), wordDocument);

ReplaceWordStud("{Stat}", Convert.ToString(dataGridView1.Rows[i].Cells[7].Value), wordDocument);

ReplaceWordStud("{OldYear}", Convert.ToString(dataGridView1.Rows[i].Cells[8].Value), wordDocument);

ReplaceWordStud("{Home}", Convert.ToString(dataGridView1.Rows[i].Cells[6].Value), wordDocument);

ReplaceWordStud("{Gromadyanstvo}", Convert.ToString(dataGridView1.Rows[i].Cells[9].Value), wordDocument);

ReplaceWordStud("{SurnameP}", Convert.ToString(dataGridView1.Rows[i].Cells[1].Value), wordDocument);

ReplaceWordStud("{NameP}", Convert.ToString(dataGridView1.Rows[i].Cells[2].Value), wordDocument);

ReplaceWordStud("{Patron}", Convert.ToString(dataGridView1.Rows[i].Cells[3].Value), wordDocument);

ReplaceWordStud("{TypeDoc}", Convert.ToString(dataGridView1.Rows[i].Cells[10].Value), wordDocument);

ReplaceWordStud("{NumDoc}", Convert.ToString(dataGridView1.Rows[i].Cells[11].Value), wordDocument);

ReplaceWordStud("{KymNaprav}", Convert.ToString(dataGridView1.Rows[i].Cells[16].Value), wordDocument);

ReplaceWordStud("{Dgoz}", Convert.ToString(dataGridView1.Rows[i].Cells[15].Value), wordDocument);

ReplaceWordStud("{Dklin}", Convert.ToString(dataGridView1.Rows[i].Cells[14].Value), wordDocument);

ReplaceWordStud("{Gk}", Convert.ToString(dataGridView1.Rows[i].Cells[12].Value), wordDocument);

ReplaceWordStud("{Rk}", Convert.ToString(dataGridView1.Rows[i].Cells[13].Value), wordDocument);

ReplaceWordStud("{Dv}", Convert.ToString(dataGridView1.Rows[i].Cells[18].Value), wordDocument);

ReplaceWordStud("{KlDay}", Convert.ToString(dataGridView1.Rows[i].Cells[19].Value), wordDocument);

wordDocument.SaveAs2(@"D:\result.docx");

wordApp.Visible = true;

break;

}

catch

{

MessageBox.Show("Виникла помилка!");

}

}

}

}

}

private void History\_Load(object sender, EventArgs e)

{

}

private void DatSetToGrid(int a, int i)

{

dataGridView1.Rows.Add();

dataGridView1.Rows[a].Cells[0].Value = Convert.ToString(hvoryis[i].getID());

dataGridView1.Rows[a].Cells[1].Value = hvoryis[i].getSurname();

dataGridView1.Rows[a].Cells[2].Value = hvoryis[i].getName();

dataGridView1.Rows[a].Cells[3].Value = hvoryis[i].getPatrn();

dataGridView1.Rows[a].Cells[4].Value = hvoryis[i].getDateB().ToString("dd-MM-yyyy");

dataGridView1.Rows[a].Cells[5].Value = hvoryis[i].getDateG().ToString("dd-MM-yyyy");

dataGridView1.Rows[a].Cells[6].Value = hvoryis[i].getHomeadres();

dataGridView1.Rows[a].Cells[7].Value = hvoryis[i].getStat();

int k = DateTime.Now.Year - hvoryis[i].getDateB().Year;

dataGridView1.Rows[a].Cells[8].Value = Convert.ToString(k);

dataGridView1.Rows[a].Cells[9].Value = hvoryis[i].getGromad();

dataGridView1.Rows[a].Cells[10].Value = hvoryis[i].getDoc();

dataGridView1.Rows[a].Cells[11].Value = hvoryis[i].getNumDoc();

dataGridView1.Rows[a].Cells[12].Value = hvoryis[i].getGkrow();

dataGridView1.Rows[a].Cells[13].Value = hvoryis[i].getRkrow();

dataGridView1.Rows[a].Cells[14].Value = hvoryis[i].getDiagnozKlin();

dataGridView1.Rows[a].Cells[15].Value = hvoryis[i].getDiagnozHospital();

dataGridView1.Rows[a].Cells[16].Value = hvoryis[i].getKymNaprav();

dataGridView1.Rows[a].Cells[17].Value = hvoryis[i].getNumP();

if (hvoryis[i].getDateV() == DateTime.MinValue)

{

dataGridView1.Rows[a].Cells[18].Value = "";

dataGridView1.Rows[a].Cells[19].Value = "";

}

else

{

dataGridView1.Rows[a].Cells[18].Value = hvoryis[i].getDateV().ToString("dd-MM-yyyy");

dataGridView1.Rows[a].Cells[19].Value = Convert.ToString((hvoryis[i].getDateV() - hvoryis[i].getDateG()).Days);

}

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox1.SelectedIndex == 0 || comboBox1.SelectedIndex == 1 || comboBox1.SelectedIndex == 2 || comboBox1.SelectedIndex == 3 || comboBox1.SelectedIndex == 4 || comboBox1.SelectedIndex == 5)

{

textBox2.Enabled = true;

textBox2.Visible = true;

dateTimePicker1.Enabled = false;

dateTimePicker1.Visible = false;

}

else

{

textBox2.Enabled = false;

textBox2.Visible = false;

dateTimePicker1.Enabled = true;

dateTimePicker1.Visible = true;

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

string t;

int kk;

if (comboBox1.SelectedIndex == 0)

{

kk = 0;

t = textBox2.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && hvoryis[i].getSurname().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox1.SelectedIndex == 1)

{

kk = 0;

t = textBox2.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && hvoryis[i].getName().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox1.SelectedIndex == 2)

{

kk = 0;

t = textBox2.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && hvoryis[i].getHomeadres().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox1.SelectedIndex == 3)

{

kk = 0;

t = textBox2.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && hvoryis[i].getStat().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox1.SelectedIndex == 4)

{

kk = 0;

t = textBox2.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && hvoryis[i].getGromad().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox1.SelectedIndex == 5)

{

kk = 0;

t = textBox2.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && hvoryis[i].getNumDoc().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

}

private void dateTimePicker1\_ValueChanged(object sender, EventArgs e)

{

int kk;

if (comboBox1.SelectedIndex == 6)

{

kk = 0;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && dateTimePicker1.Value.Date == hvoryis[i].getDateG().Date)

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox1.SelectedIndex == 7)

{

kk = 0;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && dateTimePicker1.Value.Date == hvoryis[i].getDateV().Date)

{

DatSetToGrid(kk, i);

kk++;

}

}

}

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[0-9]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

}

}

**Doctor.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace курсовий\_проект\_100.Forms

{

[Serializable]

public class Doctor : People

{

private string kvalif;

private int ID;

private string posada;

private string login;

private string passwrd;

private string stat;

public Doctor()

{

}

public void setStat(string a)

{

stat = a;

}

public string getStat()

{

return stat;

}

public void setPasswrd(string a)

{

passwrd = a;

}

public string getPasswrd()

{

return passwrd;

}

public void setLogin(string a)

{

login = a;

}

public string getLogin()

{

return login;

}

public void setKvalif(string a)

{

kvalif = a;

}

public string getKvalif()

{

return kvalif;

}

public void setPosada(string a)

{

posada = a;

}

public string getPosada()

{

return posada;

}

public void setID(int a)

{

ID = a;

}

public int getID()

{

return ID;

}

}

}

**Despancer.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using Word = Microsoft.Office.Interop.Word;

namespace курсовий\_проект\_100.Forms

{

public partial class Despancer : Form

{

List<Hvoryi> hvoryis = new List<Hvoryi>();

Doctor doctor = new Doctor();

public Despancer()

{

InitializeComponent();

UpDat\_cookies();

textBox2.Enabled = false;

textBox2.Visible = false;

dateTimePicker1.Enabled = false;

dateTimePicker1.Visible = false;

UpDat();

int kk = 0;

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

{

if (hvoryis[i].getIDdoc() == doctor.getID())

{

dataGridView1.Rows.Add();

dataGridView1.Rows[kk].Cells[0].Value = Convert.ToString(hvoryis[i].getID());

dataGridView1.Rows[kk].Cells[1].Value = hvoryis[i].getSurname();

dataGridView1.Rows[kk].Cells[2].Value = hvoryis[i].getName();

dataGridView1.Rows[kk].Cells[3].Value = hvoryis[i].getPatrn();

dataGridView1.Rows[kk].Cells[4].Value = hvoryis[i].getDateB().ToString("dd-MM-yyyy");

dataGridView1.Rows[kk].Cells[5].Value = hvoryis[i].getDateG().ToString("dd-MM-yyyy");

dataGridView1.Rows[kk].Cells[6].Value = hvoryis[i].getHomeadres();

dataGridView1.Rows[kk].Cells[7].Value = hvoryis[i].getStat();

int k = DateTime.Now.Year - hvoryis[i].getDateB().Year;

dataGridView1.Rows[kk].Cells[8].Value = Convert.ToString(k);

dataGridView1.Rows[kk].Cells[9].Value = hvoryis[i].getGromad();

dataGridView1.Rows[kk].Cells[10].Value = hvoryis[i].getDoc();

dataGridView1.Rows[kk].Cells[11].Value = hvoryis[i].getNumDoc();

dataGridView1.Rows[kk].Cells[12].Value = hvoryis[i].getGkrow();

dataGridView1.Rows[kk].Cells[13].Value = hvoryis[i].getRkrow();

dataGridView1.Rows[kk].Cells[14].Value = hvoryis[i].getDiagnozKlin();

dataGridView1.Rows[kk].Cells[15].Value = hvoryis[i].getDiagnozHospital();

dataGridView1.Rows[kk].Cells[16].Value = hvoryis[i].getKymNaprav();

dataGridView1.Rows[kk].Cells[17].Value = hvoryis[i].getNumP();

if (hvoryis[i].getDateV() == DateTime.MinValue)

{

dataGridView1.Rows[kk].Cells[18].Value = "";

dataGridView1.Rows[kk].Cells[19].Value = "";

}

else

{

dataGridView1.Rows[kk].Cells[18].Value = hvoryis[i].getDateV().ToString("dd-MM-yyyy");

dataGridView1.Rows[kk].Cells[19].Value = Convert.ToString((hvoryis[i].getDateV() - hvoryis[i].getDateG()).Days);

}

kk++;

}

}

}

private readonly string TempleteFileName = "D:\\Visual Studio2019\\3 курс\\курсовий\_проект\_100\\bin\\Debug\\Історіяя.docx";

private void UpDat\_cookies()

{

try

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("cookies.dat", FileMode.OpenOrCreate))

{

doctor = (Doctor)binaryFormatter.Deserialize(fileStream);

}

}

catch { MessageBox.Show("Не вдалося підгрузити поточного лікаря!!!"); }

}

private void UpDat()

{

hvoryis.Clear();

try

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Диспансерна\_група.dat", FileMode.OpenOrCreate))

{

hvoryis = (List<Hvoryi>)binaryFormatter.Deserialize(fileStream);

}

}

catch { }

}

private void ReplaceWordStud(string stubToReplace, string text, Word.Document wordDocument)

{

var range = wordDocument.Content;

range.Find.ClearFormatting();

range.Find.Execute(FindText: stubToReplace, ReplaceWith: text);

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "")

{

MessageBox.Show("Введіть ID пацієнта");

}

else

{

int k = Convert.ToInt32(textBox1.Text);

for (int i = 0; i < dataGridView1.RowCount; i++)

{

if (k == Convert.ToInt32(dataGridView1.Rows[i].Cells[0].Value))

{

MessageBox.Show("Зачекайте відриття Word, ваш запит на виконанні", "Процес", MessageBoxButtons.OK, MessageBoxIcon.Information);

var wordApp = new Word.Application();

wordApp.Visible = false;

var wordDocument = wordApp.Documents.Open(TempleteFileName);

try

{

ReplaceWordStud("{ID}", Convert.ToString(dataGridView1.Rows[i].Cells[0].Value), wordDocument);

ReplaceWordStud("{DateB}", Convert.ToString(dataGridView1.Rows[i].Cells[4].Value), wordDocument);

ReplaceWordStud("{DateG}", Convert.ToString(dataGridView1.Rows[i].Cells[5].Value), wordDocument);

ReplaceWordStud("{Stat}", Convert.ToString(dataGridView1.Rows[i].Cells[7].Value), wordDocument);

ReplaceWordStud("{OldYear}", Convert.ToString(dataGridView1.Rows[i].Cells[8].Value), wordDocument);

ReplaceWordStud("{Home}", Convert.ToString(dataGridView1.Rows[i].Cells[6].Value), wordDocument);

ReplaceWordStud("{Gromadyanstvo}", Convert.ToString(dataGridView1.Rows[i].Cells[9].Value), wordDocument);

ReplaceWordStud("{SurnameP}", Convert.ToString(dataGridView1.Rows[i].Cells[1].Value), wordDocument);

ReplaceWordStud("{NameP}", Convert.ToString(dataGridView1.Rows[i].Cells[2].Value), wordDocument);

ReplaceWordStud("{Patron}", Convert.ToString(dataGridView1.Rows[i].Cells[3].Value), wordDocument);

ReplaceWordStud("{TypeDoc}", Convert.ToString(dataGridView1.Rows[i].Cells[10].Value), wordDocument);

ReplaceWordStud("{NumDoc}", Convert.ToString(dataGridView1.Rows[i].Cells[11].Value), wordDocument);

ReplaceWordStud("{KymNaprav}", Convert.ToString(dataGridView1.Rows[i].Cells[16].Value), wordDocument);

ReplaceWordStud("{Dgoz}", Convert.ToString(dataGridView1.Rows[i].Cells[15].Value), wordDocument);

ReplaceWordStud("{Dklin}", Convert.ToString(dataGridView1.Rows[i].Cells[14].Value), wordDocument);

ReplaceWordStud("{Gk}", Convert.ToString(dataGridView1.Rows[i].Cells[12].Value), wordDocument);

ReplaceWordStud("{Rk}", Convert.ToString(dataGridView1.Rows[i].Cells[13].Value), wordDocument);

ReplaceWordStud("{Dv}", Convert.ToString(dataGridView1.Rows[i].Cells[18].Value), wordDocument);

ReplaceWordStud("{KlDay}", Convert.ToString(dataGridView1.Rows[i].Cells[19].Value), wordDocument);

wordDocument.SaveAs2(@"D:\result.docx");

wordApp.Visible = true;

break;

}

catch

{ MessageBox.Show("Мабуть сталася непередбачувана помилка, спробуйте пізніше", "Процес", MessageBoxButtons.OK, MessageBoxIcon.Information); }

}

}

}

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox1.SelectedIndex == 0 || comboBox1.SelectedIndex == 1 || comboBox1.SelectedIndex == 2 || comboBox1.SelectedIndex == 3 || comboBox1.SelectedIndex == 4 || comboBox1.SelectedIndex == 5)

{

textBox2.Enabled = true;

textBox2.Visible = true;

dateTimePicker1.Enabled = false;

dateTimePicker1.Visible = false;

}

else

{

textBox2.Enabled = false;

textBox2.Visible = false;

dateTimePicker1.Enabled = true;

dateTimePicker1.Visible = true;

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

string t;

int kk;

if (comboBox1.SelectedIndex == 0)

{

kk = 0;

t = textBox2.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && hvoryis[i].getSurname().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox1.SelectedIndex == 1)

{

kk = 0;

t = textBox2.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && hvoryis[i].getName().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox1.SelectedIndex == 2)

{

kk = 0;

t = textBox2.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && hvoryis[i].getHomeadres().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox1.SelectedIndex == 3)

{

kk = 0;

t = textBox2.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && hvoryis[i].getStat().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox1.SelectedIndex == 4)

{

kk = 0;

t = textBox2.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && hvoryis[i].getGromad().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox1.SelectedIndex == 5)

{

kk = 0;

t = textBox2.Text;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && hvoryis[i].getNumDoc().ToLower().Contains(t.ToLower()))

{

DatSetToGrid(kk, i);

kk++;

}

}

}

}

private void DatSetToGrid(int a, int i)

{

dataGridView1.Rows.Add();

dataGridView1.Rows[a].Cells[0].Value = Convert.ToString(hvoryis[i].getID());

dataGridView1.Rows[a].Cells[1].Value = hvoryis[i].getSurname();

dataGridView1.Rows[a].Cells[2].Value = hvoryis[i].getName();

dataGridView1.Rows[a].Cells[3].Value = hvoryis[i].getPatrn();

dataGridView1.Rows[a].Cells[4].Value = hvoryis[i].getDateB().ToString("dd-MM-yyyy");

dataGridView1.Rows[a].Cells[5].Value = hvoryis[i].getDateG().ToString("dd-MM-yyyy");

dataGridView1.Rows[a].Cells[6].Value = hvoryis[i].getHomeadres();

dataGridView1.Rows[a].Cells[7].Value = hvoryis[i].getStat();

int k = DateTime.Now.Year - hvoryis[i].getDateB().Year;

dataGridView1.Rows[a].Cells[8].Value = Convert.ToString(k);

dataGridView1.Rows[a].Cells[9].Value = hvoryis[i].getGromad();

dataGridView1.Rows[a].Cells[10].Value = hvoryis[i].getDoc();

dataGridView1.Rows[a].Cells[11].Value = hvoryis[i].getNumDoc();

dataGridView1.Rows[a].Cells[12].Value = hvoryis[i].getGkrow();

dataGridView1.Rows[a].Cells[13].Value = hvoryis[i].getRkrow();

dataGridView1.Rows[a].Cells[14].Value = hvoryis[i].getDiagnozKlin();

dataGridView1.Rows[a].Cells[15].Value = hvoryis[i].getDiagnozHospital();

dataGridView1.Rows[a].Cells[16].Value = hvoryis[i].getKymNaprav();

dataGridView1.Rows[a].Cells[17].Value = hvoryis[i].getNumP();

if (hvoryis[i].getDateV() == DateTime.MinValue)

{

dataGridView1.Rows[a].Cells[18].Value = "";

dataGridView1.Rows[a].Cells[19].Value = "";

}

else

{

dataGridView1.Rows[a].Cells[18].Value = hvoryis[i].getDateV().ToString("dd-MM-yyyy");

dataGridView1.Rows[a].Cells[19].Value = Convert.ToString((hvoryis[i].getDateV() - hvoryis[i].getDateG()).Days);

}

}

private void dateTimePicker1\_ValueChanged(object sender, EventArgs e)

{

int kk;

if (comboBox1.SelectedIndex == 6)

{

kk = 0;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && dateTimePicker1.Value.Date == hvoryis[i].getDateG().Date)

{

DatSetToGrid(kk, i);

kk++;

}

}

}

else if (comboBox1.SelectedIndex == 7)

{

kk = 0;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID() && dateTimePicker1.Value.Date == hvoryis[i].getDateV().Date)

{

DatSetToGrid(kk, i);

kk++;

}

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (textBox1.Text != "")

{

try

{

List<Hvoryi> temp = new List<Hvoryi>();

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Хворі.dat", FileMode.OpenOrCreate))

{

temp = (List<Hvoryi>)binaryFormatter.Deserialize(fileStream);

}

int k = Convert.ToInt32(textBox1.Text);

for (int i = 0; i < dataGridView1.RowCount; i++)

{

if (k == Convert.ToInt32(dataGridView1.Rows[i].Cells[0].Value))

{

temp.Add(hvoryis[i]);

BinaryFormatter binaryFormatter1 = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Хворі.dat", FileMode.Create))

binaryFormatter1.Serialize(fileStream, temp);

hvoryis.Remove(hvoryis[i]);

BinaryFormatter binaryFormatter2 = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Диспансерна\_група.dat", FileMode.Create))

binaryFormatter2.Serialize(fileStream, hvoryis);

}

}

int kk = 0;

dataGridView1.Rows.Clear();

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

{

if (hvoryis[i].getIDdoc() == doctor.getID())

{

DatSetToGrid(kk, i);

kk++;

}

}

}

catch { MessageBox.Show("Пацієнта з таким ID не існує", "Зауваження", MessageBoxButtons.OK, MessageBoxIcon.Warning); }

}

else MessageBox.Show("Введіть ІD пацієнта", "Зауваження", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[0-9]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

}

}

**Add.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace курсовий\_проект\_100.Forms

{

public partial class Add : Form

{

List<Hvoryi> hvoryis = new List<Hvoryi>();

List<Hvoryi> hvoryis\_D = new List<Hvoryi>();

List<Hvoryi> curr\_hvoryis = new List<Hvoryi>();

private int ID;

Doctor doctor = new Doctor();

bool x = true;

bool x1 = false;

public Add()

{

InitializeComponent();

comboBox5.SelectedIndex = 0;

button1.Text = "Додати пацієнта";

label1.Text = "Прізвище";

label2.Text = "Ім'я";

label3.Text = "По батькові";

label4.Text = "Громадянство";

label5.Text = "Домашня адреса";

label6.Text = "Дата народження";

label7.Text = "Дата госпіталізації";

label8.Text = "Діагноз клінічний";

label9.Text = "Діагноз при госпіталізації";

label10.Text = "Ким направлений хворий";

label12.Text = "Група крові";

label11.Text = "Резус-приналежності";

label13.Text = "Номер документа";

label14.Text = "Докумен, що посвідчує особу";

label15.Text = "Номер палати";

label31.Text = "Введіть прізвище для пошуку";

label30.Text = "Прізвище";

label29.Text = "Ім'я";

label28.Text = "По батькові";

label27.Text = "Громадянство";

label26.Text = "Домашня адреса";

label25.Text = "Дата народження";

label24.Text = "Дата госпіталізації";

label23.Text = "Діагноз клінічний";

label22.Text = "Діагноз при госпіталізації";

label21.Text = "Ким направлений хворий";

label20.Text = "Група крові";

label19.Text = "Резус-приналежності";

label18.Text = "Номер документа";

label17.Text = "Докумен, що посвідчує особу";

label16.Text = "Номер палати";

label33.Text = "ID пацієнта";

label36.Text = "ID пацієнта";

label35.Text = "";

textBox14.Text = "";

label37.Text = "Введіть ID";

groupBox1.Text = "Стать";

groupBox2.Text = "Група крові";

groupBox4.Text = "Пошук";

button5.Visible = false;

button5.Enabled = false;

UpDat\_cookies();

UpDat();

UpdatDD(); int maxID = 0;

if (hvoryis.Count != 0) maxID = hvoryis[0].getID();

int o = 0;

for (int k = 0; k < hvoryis.Count; k++)

{

if (maxID < hvoryis[k].getID())

{

o = k;

}

if (hvoryis[k].getIDdoc() == doctor.getID() && hvoryis[k].getDateV() == DateTime.MinValue)

{

curr\_hvoryis.Add(hvoryis[k]);

}

}

if (curr\_hvoryis.Count != 0) Add\_DatInFormVisio(curr\_hvoryis[0]);

else MessageBox.Show("Перший запуск програми або всі пацієнти виписані");

try

{

ID = hvoryis[o].getID() + 1;

}

catch

{

ID = 1;

}

label34.Text = ID.ToString();

groupBox3.Text = "Група крові";

radioButton1.Text = "Чоловік";

radioButton2.Text = "Жінка";

label32.Text = "Стать";

dateTimePicker2.Value = DateTime.Now;

comboBox5.Text = "";

textBoxHome.Text = "";

textBoxNam.Text = "";

textBoxPatrn.Text = "";

textBoxSurN.Text = "";

richTextBox1.Text = "";

textBox12.Text = "";

comboBox1.SelectedIndex = 0;

comboBox2.SelectedIndex = 0;

button2.Text = "Наступний";

button3.Text = "Попередній";

button4.Text = "Знайти";

}

private void Add\_DatInFormVisio(Hvoryi hvoryi)

{

textBox10.Text = hvoryi.getSurname();

textBox11.Text = hvoryi.getName();

textBox9.Text = hvoryi.getPatrn();

textBox8.Text = hvoryi.getGromad();

textBox7.Text = hvoryi.getHomeadres();

comboBox3.Text = hvoryi.getDoc();

textBox5.Text = hvoryi.getNumDoc();

dateTimePicker4.Value = hvoryi.getDateB();

dateTimePicker3.Value = hvoryi.getDateG();

textBox6.Text = hvoryi.getGkrow();

comboBox4.Text = hvoryi.getRkrow();

textBox13.Text = hvoryi.getStat();

richTextBox5.Text = hvoryi.getDiagnozHospital();

richTextBox6.Text = hvoryi.getDiagnozKlin();

richTextBox4.Text = hvoryi.getKymNaprav();

textBox4.Text = hvoryi.getNumP();

label35.Text = hvoryi.getID().ToString();

}

private void Clear\_Add\_Page2()

{

textBox10.Text = "";

textBox11.Text = "";

textBox9.Text = "";

textBox8.Text = "";

textBox7.Text = "";

comboBox3.Text = "";

textBox5.Text = "";

textBox6.Text = "";

comboBox4.Text = "";

textBox13.Text = "";

richTextBox6.Text = "";

richTextBox5.Text = "";

richTextBox4.Text = "";

}

private Hvoryi createHV()

{

Hvoryi hvoryi = new Hvoryi();

hvoryi.setNumP(textBox1.Text);

hvoryi.setDateB(dateTimePicker1.Value);

hvoryi.setDateG(dateTimePicker2.Value);

hvoryi.setDiagnozHospital(richTextBox2.Text);

hvoryi.setDiagnozKlin(richTextBox1.Text);

hvoryi.setDoc(comboBox2.Text);

hvoryi.setGkrow(textBox2.Text);

hvoryi.setGromad(comboBox5.Text);

hvoryi.setHomeadres(textBoxHome.Text);

hvoryi.setKymNaprav(richTextBox3.Text);

hvoryi.setName(textBoxNam.Text);

hvoryi.setNumDoc(textBox3.Text);

hvoryi.setPatrn(textBoxPatrn.Text);

hvoryi.setRkrow(comboBox1.Text);

if (radioButton1.Checked)

hvoryi.setStat(radioButton1.Text);

else hvoryi.setStat(radioButton2.Text);

hvoryi.setSurname(textBoxSurN.Text);

hvoryi.setNumP(textBox1.Text);

hvoryi.setID(ID);

hvoryi.setIDdoc(doctor.getID());

return hvoryi;

}

private void ClearPage1()

{

comboBox5.SelectedIndex = 0;

textBoxHome.Text = "";

textBoxNam.Text = "";

textBoxPatrn.Text = "";

textBoxSurN.Text = "";

textBox3.Text = "";

richTextBox1.Text = "";

richTextBox2.Text = "";

richTextBox3.Text = "";

textBox1.Text = "";

textBox2.Text = "";

}

private void writeAll()

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Хворі.dat", FileMode.OpenOrCreate))

binaryFormatter.Serialize(fileStream, hvoryis);

}

private void UpDat\_cookies()

{

try

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("cookies.dat", FileMode.OpenOrCreate))

{

doctor = (Doctor)binaryFormatter.Deserialize(fileStream);

}

}

catch { MessageBox.Show("Не вдалося підгрузити поточного лікаря!!!"); }

}

private void UpDat()

{

hvoryis.Clear();

try

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Хворі.dat", FileMode.OpenOrCreate))

{

hvoryis = (List<Hvoryi>)binaryFormatter.Deserialize(fileStream);

}

}

catch { }

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBoxSurN.Text == "" || textBoxNam.Text == "" || textBoxPatrn.Text == "" || textBoxHome.Text == "" || textBox3.Text == "" || textBox1.Text == "" || textBox2.Text == "")

{

MessageBox.Show("Заповніть усі поля!!!");

}

else

{

hvoryis.Add(createHV());

writeAll();

curr\_hvoryis.Add(createHV());

MessageBox.Show("Пацієнт доданий");

UpDat();

ClearPage1();

ID++;

label34.Text = ID.ToString();

}

}

private int curentH = 0;

private void button3\_Click(object sender, EventArgs e)

{

try

{

if (curentH == 0)

{

curentH = curr\_hvoryis.Count - 1;

Add\_DatInFormVisio(curr\_hvoryis[curentH]);

}

else

{

curentH--;

Add\_DatInFormVisio(curr\_hvoryis[curentH]);

}

}

catch { }

}

private void button4\_Click(object sender, EventArgs e)

{

string x1;

bool posh = false;

int k = 0;

string r = "\n";

if (textBox14.Text == "" && textBox12.Text == "") MessageBox.Show("Введіть дані для пошуку");

else if (textBox14.Text != "" && textBox12.Text == "")

{

try

{

int pp = Convert.ToInt32(textBox14.Text) - 1;

Add\_DatInFormVisio(curr\_hvoryis[pp]);

textBox12.Text = "";

textBox14.Text = "";

}

catch

{

MessageBox.Show("Мабуть такого пацієнта не існує :((");

textBox12.Text = "";

textBox14.Text = "";

}

}

else

{

try

{

x1 = Convert.ToString(textBox12.Text);

for (int i = 0; i < curr\_hvoryis.Count; i++)

{

if (curr\_hvoryis[i].getSurname() == x1)

{

r += (i + 1).ToString() + "-" + curr\_hvoryis[i].getSurname() + " " + curr\_hvoryis[i].getName() + "\n";

k++;

posh = true;

}

}

if (!posh)

{

MessageBox.Show("Такого запису не знайдено", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

Clear\_Add\_Page2();

}

else

{

MessageBox.Show("Кількість записів = " + k + r, "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

textBox12.Text = "";

textBox14.Text = "";

}

catch

{

MessageBox.Show("Схоже виникла помилка спробуйте пізніше");

textBox12.Text = "";

textBox14.Text = "";

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

try

{

if (curentH == curr\_hvoryis.Count - 1)

{

curentH = 0;

Add\_DatInFormVisio(curr\_hvoryis[curentH]);

}

else

{

curentH++;

Add\_DatInFormVisio(curr\_hvoryis[curentH]);

}

}

catch { }

}

private void редагуватиToolStripMenuItem\_Click(object sender, EventArgs e)

{

textBox4.Enabled = true;

textBox5.Enabled = true;

textBox6.Enabled = true;

textBox7.Enabled = true;

textBox8.Enabled = true;

textBox9.Enabled = true;

textBox10.Enabled = true;

textBox11.Enabled = true;

textBox13.Enabled = true;

comboBox3.Enabled = true;

comboBox4.Enabled = true;

richTextBox4.Enabled = true;

richTextBox5.Enabled = true;

richTextBox6.Enabled = true;

button5.Visible = true;

button5.Enabled = true;

dateTimePicker4.Enabled = true;

}

private void button5\_Click(object sender, EventArgs e)

{

if (textBox4.Text == "" || textBox5.Text == "" || textBox6.Text == "" || textBox7.Text == "" || textBox8.Text == "" || textBox9.Text == "" || textBox10.Text == "" || textBox11.Text == "" || textBox13.Text == "")

{

MessageBox.Show("Заповніть усі поля!!!");

}

else

{

Hvoryi hvoryi = new Hvoryi();

hvoryi.setDateB(dateTimePicker4.Value);

hvoryi.setDateG(dateTimePicker3.Value);

hvoryi.setDiagnozHospital(richTextBox6.Text);

hvoryi.setDiagnozKlin(richTextBox5.Text);

hvoryi.setDoc(comboBox3.Text);

hvoryi.setGkrow(textBox6.Text);

hvoryi.setGromad(textBox8.Text);

hvoryi.setHomeadres(textBox7.Text);

hvoryi.setKymNaprav(richTextBox4.Text);

hvoryi.setName(textBox11.Text);

hvoryi.setNumDoc(textBox5.Text);

hvoryi.setPatrn(textBox9.Text);

hvoryi.setRkrow(comboBox4.Text);

hvoryi.setStat(textBox13.Text);

hvoryi.setSurname(textBox10.Text);

hvoryi.setNumP(textBox4.Text);

hvoryi.setID(curr\_hvoryis[curentH].getID());

hvoryi.setIDdoc(doctor.getID());

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

{

if (hvoryis[i].getID() == curr\_hvoryis[curentH].getID())

{

hvoryis.Remove(hvoryis[i]);

}

}

hvoryis.Add(hvoryi);

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Хворі.dat", FileMode.Create))

binaryFormatter.Serialize(fileStream, hvoryis);

curr\_hvoryis[curentH] = hvoryi;

MessageBox.Show("Дані пацієнта оновлено", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

textBox4.Enabled = false;

textBox5.Enabled = false;

textBox6.Enabled = false;

textBox7.Enabled = false;

textBox8.Enabled = false;

textBox9.Enabled = false;

textBox10.Enabled = false;

textBox11.Enabled = false;

textBox13.Enabled = false;

comboBox3.Enabled = false;

comboBox4.Enabled = false;

richTextBox4.Enabled = false;

richTextBox5.Enabled = false;

richTextBox6.Enabled = false;

button5.Visible = false;

button5.Enabled = false;

dateTimePicker4.Enabled = false;

button5.Visible = false;

button5.Enabled = false;

}

private void виписатиToolStripMenuItem\_Click(object sender, EventArgs e)

{

try

{

curr\_hvoryis[curentH].setDateV(DateTime.Now);

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

{

if (hvoryis[i].getID() == curr\_hvoryis[curentH].getID())

{

hvoryis.Remove(hvoryis[i]);

}

}

hvoryis.Add(curr\_hvoryis[curentH]);

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Хворі.dat", FileMode.Create))

binaryFormatter.Serialize(fileStream, hvoryis);

MessageBox.Show("Пацієнта виписано", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch { MessageBox.Show("Немає пацієнта щоб виписати", "Увага", MessageBoxButtons.OK, MessageBoxIcon.Warning); }

}

private void UpdatDD()

{

try

{

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Диспансерна\_група.dat", FileMode.OpenOrCreate))

{

hvoryis\_D = (List<Hvoryi>)binaryFormatter.Deserialize(fileStream);

}

}

catch { }

}

private void перевестиВДиспансернуГрупуToolStripMenuItem\_Click(object sender, EventArgs e)

{

try

{

curr\_hvoryis[curentH].setDateV(DateTime.Now.Date);

hvoryis\_D.Add(curr\_hvoryis[curentH]);

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

{

if (hvoryis[i].getID() == curr\_hvoryis[curentH].getID())

{

hvoryis.Remove(hvoryis[i]);

}

}

curr\_hvoryis.Remove(curr\_hvoryis[curentH]);

BinaryFormatter binaryFormatter = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Хворі.dat", FileMode.Create))

binaryFormatter.Serialize(fileStream, hvoryis);

BinaryFormatter binaryFormatter4 = new BinaryFormatter();

using (FileStream fileStream = new FileStream("Диспансерна\_група.dat", FileMode.OpenOrCreate))

binaryFormatter4.Serialize(fileStream, hvoryis\_D);

}

catch { MessageBox.Show("Немає пацієнта щоб додати в диспансерну група", "Увага", MessageBoxButtons.OK, MessageBoxIcon.Warning); }

}

private void textBox6\_TextChanged(object sender, EventArgs e)

{

}

private void textBox6\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[1-4]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox6\_KeyDown(object sender, KeyEventArgs e)

{

}

private void textBox10\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-ЯІіЇї']").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox11\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-ЯіІїЇ']").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox9\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-ЯіІїЇ']").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBoxSurN\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-ЯіІїЇ']").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBoxNam\_TextChanged(object sender, EventArgs e)

{

}

private void textBoxNam\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-ЯіІїЇ']").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBoxPatrn\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-ЯіІїЇ']").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[0-9]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void richTextBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я їЇІі' , 0-9 . ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void richTextBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я їЇІі' , 0-9 . ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void richTextBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-ЯіІїЇ .' ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBoxHome\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-ЯіІїЇ , 0-9 .' ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[A-Z 0-9]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[1-4]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox8\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-ЯіІїЇ']").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox7\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я іІїЇ , 0-9 .' ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox5\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[A-Z 0-9]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox12\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-ЯіІїЇ']").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox14\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[0-9]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void richTextBox6\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я іІїЇ , 0-9 .' ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void richTextBox5\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я і ІїЇ , 0-9 .' ]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void richTextBox4\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я і ІїЇ . ']").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox13\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-ЯІіїЇ']").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

private void textBox4\_KeyPress(object sender, KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[0-9]").Success) e.Handled = true;

if (e.KeyChar == (char)Keys.Back) e.Handled = false;

}

}

}