Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни “ Об’єктно - орієнтоване програмування”

“Посібник лікаря”

Керівник , проф. Бондарєв В.М.

Студент гр. ПІ - 15 - 1 Маначенко М.В.

Комісія:

проф. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дудар З.В.

<посада> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_<прізвище, ініціали >

<посада> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_<прізвище, ініціали >

Харків 2016

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

Дисципліна: ***Об’єктно-орієнтоване програмування***

Спеціальність: ***Програмна інженерія***

Курс 1 .Група *ПІ-15-1* . Семестр *2* .

***ЗАВДАННЯ***

***на курсовий проект студента***

*Маначенко Маргарити Вікторівни*

1. Тема проекту: «Посібник лікаря»
2. Термін здачі студентом закінченого проекту: ***“18” - травня - 2016 р***.
3. Вихідні дані до проекту: база хвороб: назва, симптоми, процедури, перелік рекомендованих ліків із вказанням потребуємої кількості. База медикаментів на складі: назва, кількість, взаємозамінність. Формування рецепта після огляду хворого, перевірка наявності ліків, корегування запасів.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 1-03-2016 р. |
| 2 | Формулювання вимог до програми | 1-03-2016 – 6-03-2016 р. |
| 3 | Розробка підсистеми зберігання та пошуку медикаментів та хвороб. | 7-03-2016 – 20-03-2016 р. |
| 4 | Розробка підсистеми зберігання та пошуку медикаментів та хвороб. | 21-03-2016 – 30-03-2016 р. |
| 5 | Розробка функцій формування рецепту за назвою хвороби. | 1-04-2016 – 3-04-2016 р. |
| 6 | Розробка функцій зберігання та завантаження даних | 4-04-2016 – 9-04-2016 р. |
| 7 | Тестування і доопрацювання розробленої програмної системи. | 10-04-2016 – 29-04-2016 р. |
| 8 | Оформлення пояснювальної записки, додатків, графічного матеріалу | 1-05-2016 – 31-05-2016 р. |
| 9 | Захист | 18-05-2016 р. |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Маначенко М.В.

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарєв В. М.

« 1 »\_березня\_\_\_\_\_\_\_\_\_\_ 2016 р.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 56 с., 32 рис., 1 додаток, 5 джерел.

Метою роботи є розробка програми “Посібник лікаря” на засадах об'єктно-орієнтованого програмування.

Методи розробки базуються на використанні середи розробки Microsoft Visual Studio 2015, Windows Forms, платформи .NET Framework 4.5, мови програмування C#.

Профіль на github: https://github.com/margaretMan

В результаті отримана програма під назвою “Посібник лікаря”, яка дозволяє зберігати список медикаментів та хвороб, характеристики кожного медикамента та хвороби. Медикамент представляють такі пункти як: назва, опис, дозування, кількість на складі, а хворобу – назва, симптоми, процедури, список ліків. Є можливість додавати, видаляти та редагувати медикаменти та хвороби. Також, є функція формування рецепту в html документ та друк.

ПРОГРАМА, ОБ’ЕКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, КЛАС, МОВА ПРОГРАМУВАННЯ C#, ПЛАТФОРМА .NET, МЕДИКАМЕНТ, XML, ХВОРОБА, РЕЦЕПТ.

ЗМІСТ

Вступ 6

1 Специфікація програми 7

1.1 Список задач 7

1.2 Функції програми 7

1.3 Інтерфейс користувача 8

2 Проектна специфікація. 20

2.1 Архітектурний стиль 20

2.2. Об’єктна структура програми 20

2.3 Формат даних 22

3 Інструкція користувача 24

ВИСНОВКИ 34

ПЕРЕЛІК ПОСИЛАНЬ.. 35

Додаток А – Код програми.. 36

ВСТУП

Медицина – дуже непроста наука. Наявність складних термінів у медицині часто змушує звертатися до різноманітних джерел інформації. Крім цього, змінюються форми випуску медикаментів, з’являються нові торгові назви і дозування. Все це потрібно брати до уваги кожному лікарю незалежно від спеціалізації. Посібник лікарських засобів необхідний у практиці будь-якого медика кожен день.

Дослідивши переваги та недоліки паперових носіів інформаціїї, будо прийнято рішення створити прикладну програму для вищеперелічених цілей.

Електроний посібник лікаря вміщає у собі описи багатьох медикаментів і захворювань, а також функцію формування рецепту.

Із назви зрозуміло, що посібник містить інформацію для лікарів, але це не зовсім так. Медичні довідники необхідні і простим людям, не тільки медикам. Пошук необхідного медикаменту або хвороби відбувається швидко і просто – це основна перевага електронного посібника перед паперовим.

Варто зауважити, що використання даного довідника для домашнього користування призначене тільки для ознайомлення з інформацією, але ніяк не для самолікування.

1 СПЕЦИФІКАЦІЯ ПРОГРАМИ

1.1 Список задач

Метою курсового проекту є створення програми «Посібник лікаря».

Посібник має містити у собі детальну інформацію щодо лікарських засобів, а саме: назву, опис препарату, дозування (порція та кількість прийомів на день), тривалість лікування, кількість на складі. А також базу хвороб з характеристиками: назва, симптоми, процедури, список медикаментів для лікування.

У програмі повинна бути функція "Виписати рецепт", що формує та друкує список лікарських засобів,наявних на складі та вибраних користувачем, а також список рекомендованих процедур для лікування вказаного захворювання або вказаних симптомів.

Не менш важливим є інтерфейс програми. Він повинен бути інтуїтивно зрозумілим та простим у використанні. Одією із найважливіших задач є правильна реакція програми на будь-які дії користувача.

1.2 Функції програми

Програма, що буде розроблена, повинна мати наступні функції:

1. відображення бази медикаментів;
2. додавання нового медикаменту;
3. редагування медикаменту;
4. видалення медикаменту;
5. пошук медикамента за назвою або описом;
6. відображення бази хвороб;
7. додавання нової хвороби;
8. редагування хвороби;
9. видалення хвороби;
10. пошук хвороби за назвою, симптомами, процедурами або лікуванням;
11. формування рецепту за назвою хвороби або її симптомами;
12. вибір користувачем ліків, серед тих,що є на складі;
13. друк рецепту.

1.3 Інтерфейс користувача

Після запуску програми користувач побачить головну форму, вона ж форма управління базою медикаментів (рисунок 1.1).

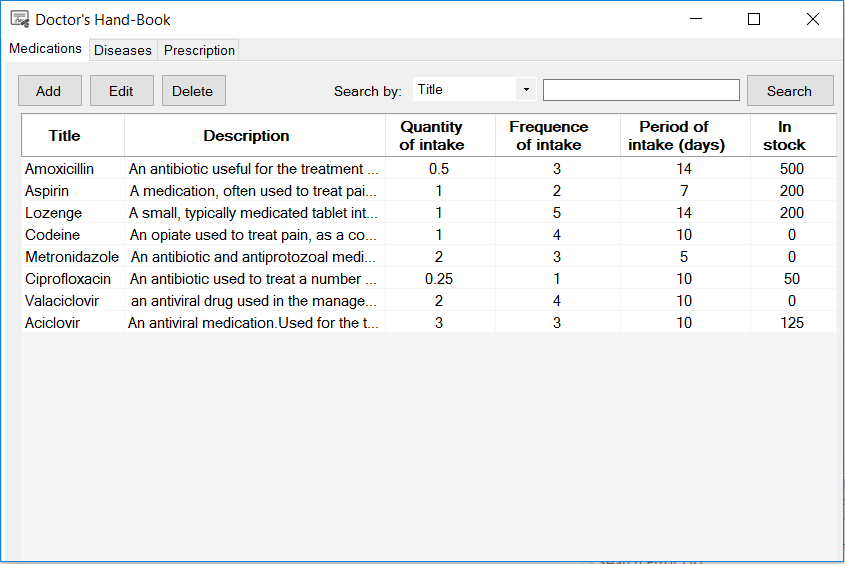


Рисунок 1.1 – Головна форма

У меню вибрана закладка «Medications». На формі зображена база медикаментів. Таким чином користувач може переглянути всі характеристики бажаного лікарського засобу, а саме: назву, опис, скільки разів на день приймати певну кількість одиниць препарату та протягом якого періоду, а також кількість одиниць на складі.

При натисканні кнопки «Add» користувач отримає можливість додати новий препарат до існуючої бази. Форма додавання елемента зображена на рисунку 1.2.

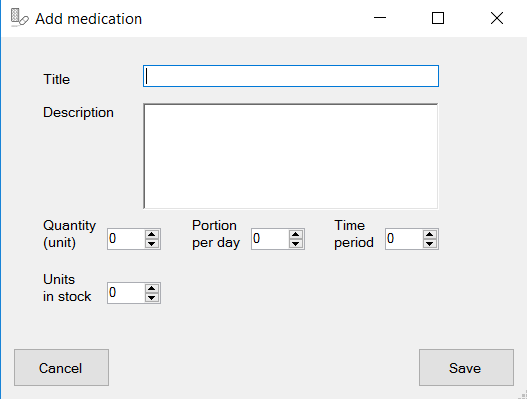


Рисунок 1.2 – Форма додавання медикаменту

Поля форми незаповнені. На формі присутні дві кнопки: «Cancel» і «Save». «Cancel» закриває форму без зберігання введених даних, а користувач повертається до головної форми. «Save» також закривє форму, але зберігає новий лікарський засіб, якщо користувач не залишив поля пустими. При цьому користувач побачить, що новий медикамент уже занесений у базу на головній формі закладки «Medications». У разі, якщо хоча б одне поле залишилось незаповненим, користувач побачить попередження (рисунок 1.3)

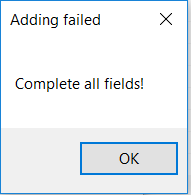


Рисунок 1.3 – Форма «Попередження»

При виділенні рядку у базі та натисканні кнопки «Edit» користувач побачить форму редагування виділеного препарату, наведену на рисунку 1.4.

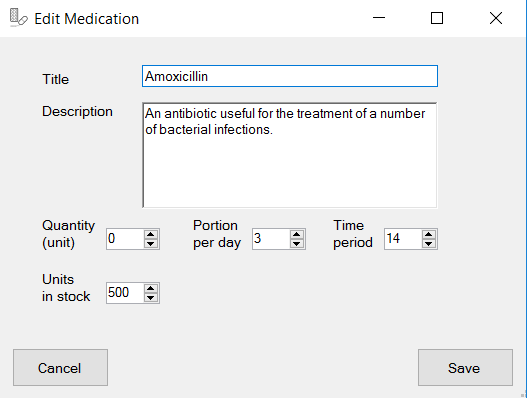


Рисунок 1.4 – Форма редагування медикаменту

Поля форми заповнені даними з бази. На формі знаходяться дві кнопки: «Cancel» і «Save». «Cancel» закриває форму, виділений препарат залишається незмінним і користувач повертається до головної форми. «Save» також закривє форму, але зберігає зміни у лікарському засібі. Перевірка на заповненість полів також присутня. На головній формі у базі користувач побачить зроблені ним зміни у даному препараті.

При натисканні «Delete» відобразиться дочірня форма (рисунок 1.5) для підтвердження видалення виділеного у базі препарату.

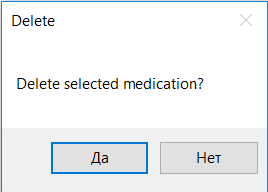


Рисунок 1.5 – Форма підтвердження видалення медикаменту

Функція пошуку дозволяє вибирати один або більше лікарських засобів у базі за двома категоріями, зображеними на рисунку 1.6. В залежності від вибору, необхідно заповнити поле пошуку та натиснути «Search». У разі наявності рядок з медикаментом стане виділеним.

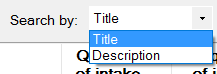


Рисунок 1.6 – Категорії пошуку медикаменту

При виборі наступної закладки «Diseases» відображається форма управління базою хвороб як на рисунку 1.7. Користувач може переглянути всі характеристики бажаної хвороби, а саме: назву, симптоми, процедури, а також список медикаментів, що допомагають у лікуванні хвороби.

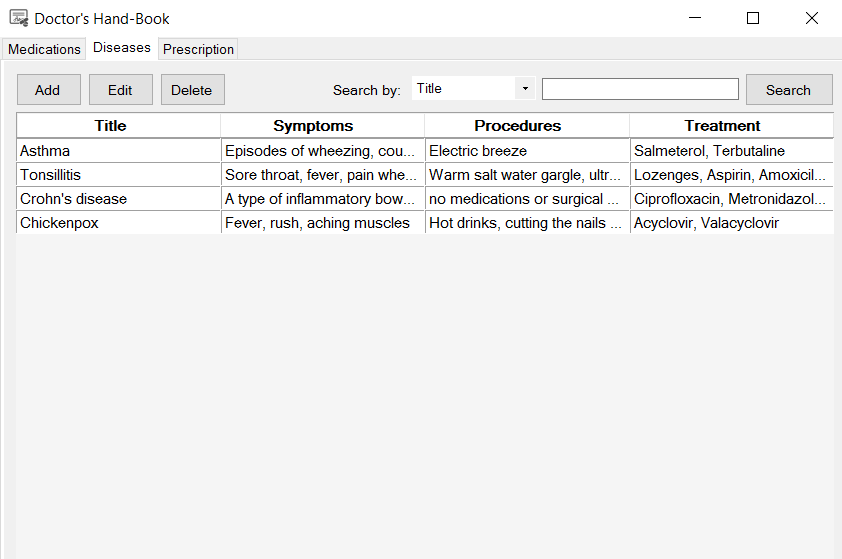


Рисунок 1.7 – Форма управління хворобами

При натисканні кнопки «Add» користувач отримає можливість додати нову хворобу до існуючої бази. Форма додавання елемента зображена на рисунку 1.8.

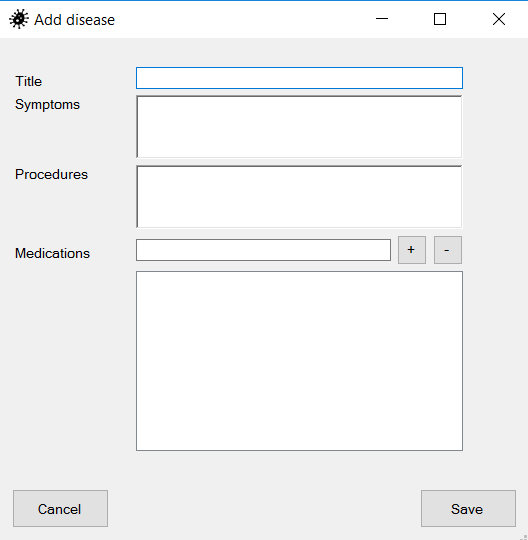


Рисунок 1.8 – Форма додавання хвороби

Поля форми незаповнені. На формі присутні дві кнопки: «Cancel» і «Save». «Cancel» закриває форму без зберігання введених даних, а користувач повернеться до форми управління хворобами. «Save» зберігає нову хворобу, якщо користувач заповнив усі поля. При цьому користувач побачить нову хворобу занесену у базу. У разі, якщо хоча б одне поле залишилось незаповненим, користувач побачить попередження (рисунок 1.3). Також на формі присутні кнопки «+» і «-», що дозволяють формувати список лікарських засобів. «+» для додавання медикаменту у список і «-» для видалення зі списку.

При виділенні рядку у базі та натисканні кнопки «Edit» користувач побачить форму редагування виділеної хвороби наведену на рисунку 1.9.

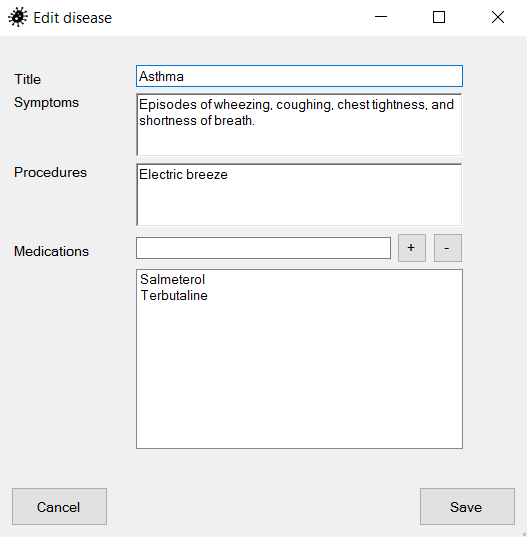


Рисунок 1.9 – Форма редагування хвороби

Поля форми заповнені даними з бази. На формі знаходяться дві кнопки: «Cancel» і «Save». «Cancel» закриває форму, виділений препарат залишається незмінним і користувач повертається до головної форми. «Save» також закривє форму, але зберігає зміни у хворобі. Перевірка на заповненість полів також присутня. На головній формі у базі користувач побачить зроблені ним зміни у даній хворобі.

При натисканні «Delete» відобразиться дочірня форма (рисунок 1.10) для підтвердження видалення виділеного у базі препарату.

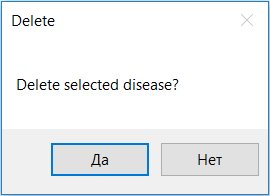


Рисунок 1.10 - Форма підтвердження видалення медикаменту

Наявна функція пошуку, яка дозволяє вибирати одну або більше хвороб у базі за чотирма категоріями, зображеними на рисунку 1.11. В залежності від вибору, необхідно заповнити поле пошуку та натиснути «Search». У разі наявності рядок з захворюванням стане виділеним.

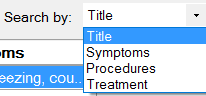


Рисунок 1.11 – Категорії пошуку хвороби

При виборі останньої закладки «Prescription» відобразиться форма формувавання рецепту (рисунок 1.12).

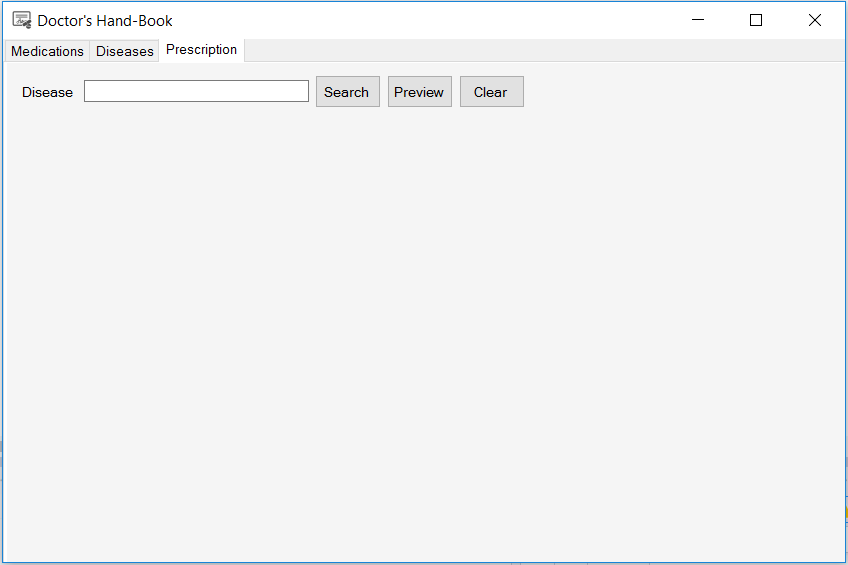


Рисунок 1.12 – Формування рецепту

На формі присутнє поле для пошуку хвороби за назвою. При його заповненні та натисканні кнопки «Search» на формі буде сформовано перелік засобів лікування, таких як медикаменти та процедури. Якщо потрібний медикамент буде наявним на складі, він відобразиться в активному стані (приклад на рисунку 1.13). Користувач зможе сам зформувати список ліків, вибравши їх зі списку наявних.

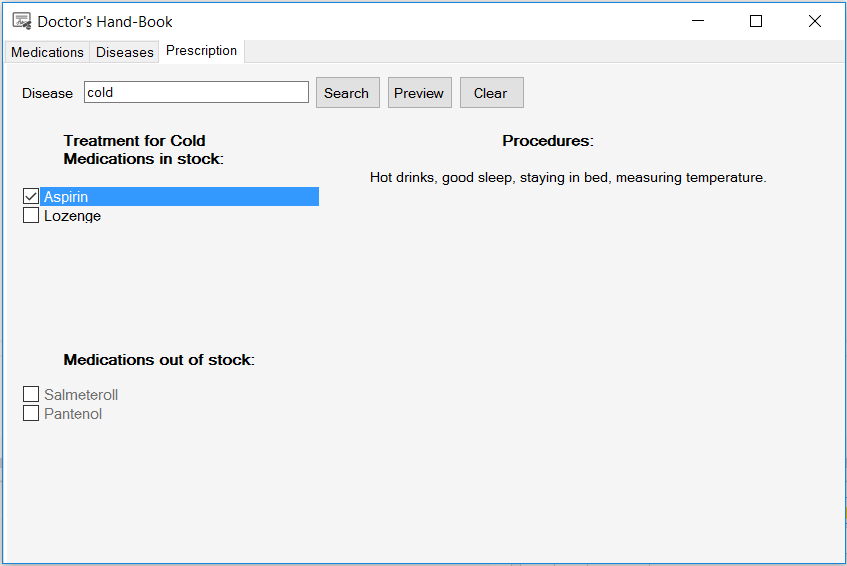


Рисунок 1.13 – Показ наявних засобів від хвороби

При натисканні «Preview», ураховуючи вибрані медикаменти та процедури, в окремому вікні буде сформовано рецепт (рисунок 1.14), який користувач зможе роздрукувати, натиснувши кнопку «Print». При цьому з’явиться діалогове вікно для вибору принтера як на рисунку 1.15.



Рисунок 1.14 – Приклад сформованого рецепту

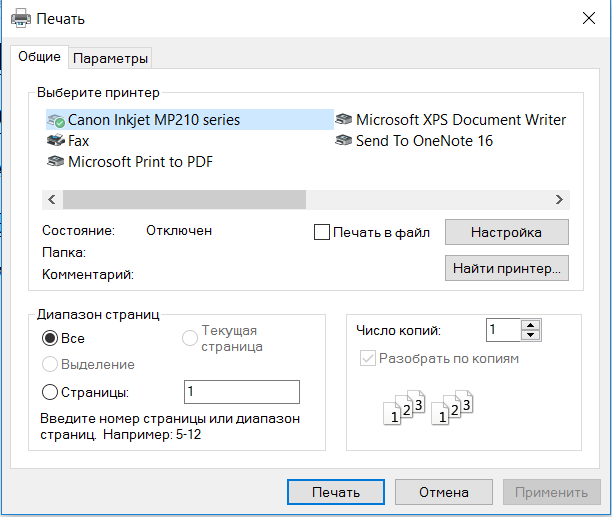


Рисунок 1.15 – Діалогове вікно друку

2 ПРОЕКТНА СПЕЦИФІКАЦІЯ

2.1 Архітектурний стиль

Для реалізації даного проекта найкраще підходить архітектура MVC. У ній вся логіка поділяється на три окремих компоненти: Model, View, Controller.

Model (модель) називають частину програми, в якій міститься функціональна логіка програми.

У компоненті View (зображення) відбувається відображення даних моделі. Зображення має вільний доступ до моделі, але тільки для зчитування.

Controller (контролер) реагує на зовнішні події та змінює модель та зображення. Варто зазначити,що контролер не є посередником між моделлю і зображенням, не передає ніякої інформації з одного компонену в інший.

У процесі розробки було вирішено об’єднати контролер і зображення. Це ніяк не порушує ідею MVC. Головна задача – зробити модель повністю незалежною від інших частин програми, що дозволить робити зміни у контролері або зображенні, не впливаючи на модель.

Зараз проект працює з файлами, але у майбутньому може виникнути необхідність підключити базу данних. Дуже часто також виникає необхідність змінити зображення, не змінюючи інші частини програми. Саме через ці причини була вибрана архітектура MVC.

2.2. Обєктна структура програми

Програма представлена набором класів, що взаємодіють між собою (рисунок 2.1).

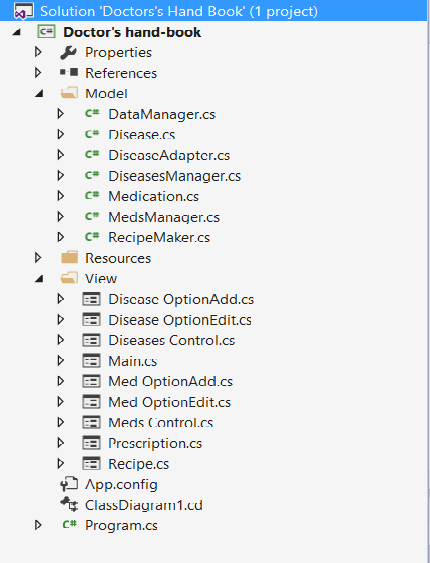


Рисунок 2.1. – Архітектура проекту

У програмі реалізовані такі класи: Medication, MedsManager, Disease, DiseaseManager, DiseaseAdapter, DataManager, RecipeMaker. Діаграма класів представлена на рисунку 2.2.

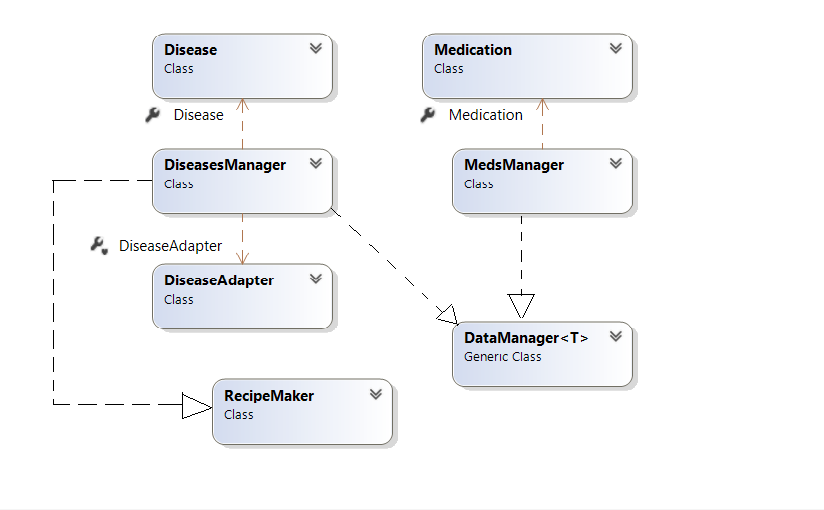


Рисунок 2.2 – Діаграма класів

2.3. Формат даних

Для зберігання інформації використовується механізм серіалізації. Серіалізація – це процес перетворення об’єкта у потік байтів. Після перетворення цей потік байтів можна записати на диск або тимчасово зберегти у пам’яті. За необхідністю можливий зворотній процес – десереалізація, тобто отримання із потоку байтів попередньо збереженого об’єкту.

У програмі всі дані зберігаються у .xml файлі, як показано на рисунку 2.3.

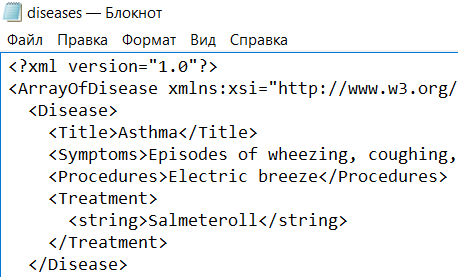


Рисунок 2.3 – Зберігання даних

3 ІНСТРУКЦІЯ КОРИСТУВАЧА

Встановлення програми:

* 1. Запускаємо «Мій комп'ютер».
  2. Заходимо на носій з програмним продуктом.
  3. Розпаковуємо архів у бажану папку на комп'ютері.
  4. Можна запускати «.exe» файл і починати роботу з програмою.

Робота з програмою

Запускаємо програму. Головне вікно зображене на рисунку 3.1. За змовчуванням вибрана закладка «Medications».

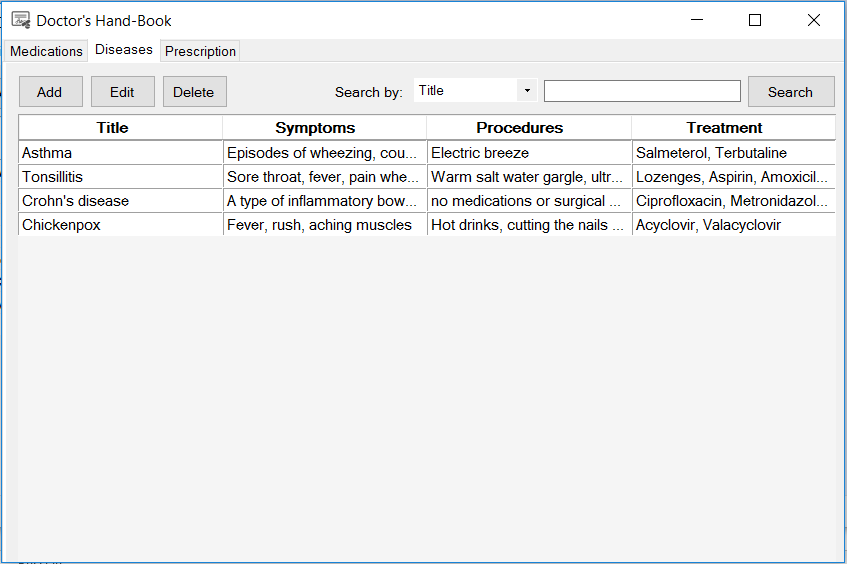


Рисунок 3.1 – Головне вікно

Додавання медикаменту:

1. натиснути «Add»;
2. відкриється діалогове вікно «Add Medication» (рисунок 3.2);

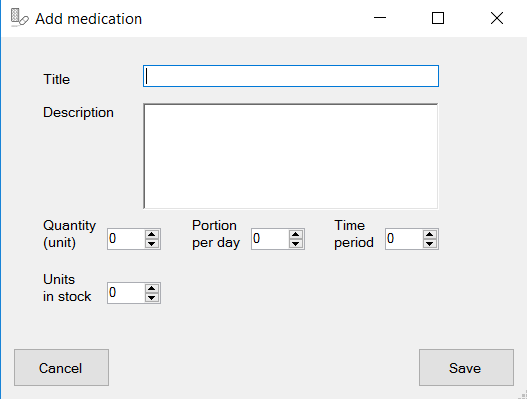


Рисунок 3.2 – Вікно додавання медикаменту

1. потрібно заповнити всі поля і натиснути «Save»;
2. при правильному виконанні всіх дії новий медикамент відобразиться у базі на головній формі.

Редагування медикаменту:

1. натиснути «Edit»;
2. відкриється діалогове вікно «Edit Medication» (рисунок 3.3);

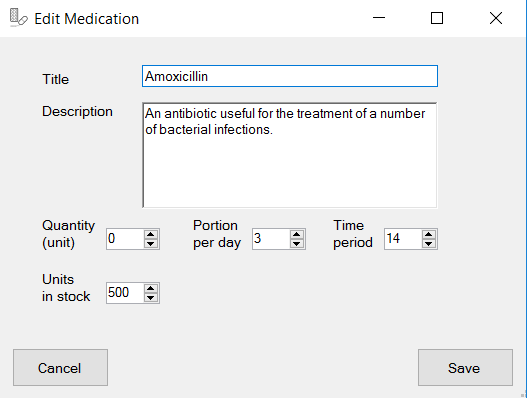


Рисунок 3.3 – Вікно редагування медикаменту

1. вносимо необхідні зміни;
2. при правильному виконанні всіх дії медикамент буде змінено, результат можна буде побачити у базі на головній формі.

Видалення медикаменту:

1. вибрати з бази бажаний медикаментж
2. натиснути «Delete»;
3. з’явится вікно для підтверждення видалення (рисунок 3.4).

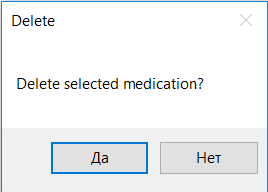


Рисунок 3.4 – Вікно підтвердження вибору

1. підтвердити свій вибір.

Пошук медикаменту за назвою або описом:

1. на головній формі вибрати критеріїї пошуку: назва або опис (рисунок 3.5);

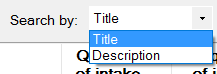


Рисунок 3.5 – Список критеріїв пошуку

1. обовязково заповнити поле пошуку;

У разі успішного пошуку потрібні рядки у базі видяляться. У противному разі, жоден рядок не буде виділеним.

Переходимо до наступної закладки «Diseases» (рисунок 3.6).

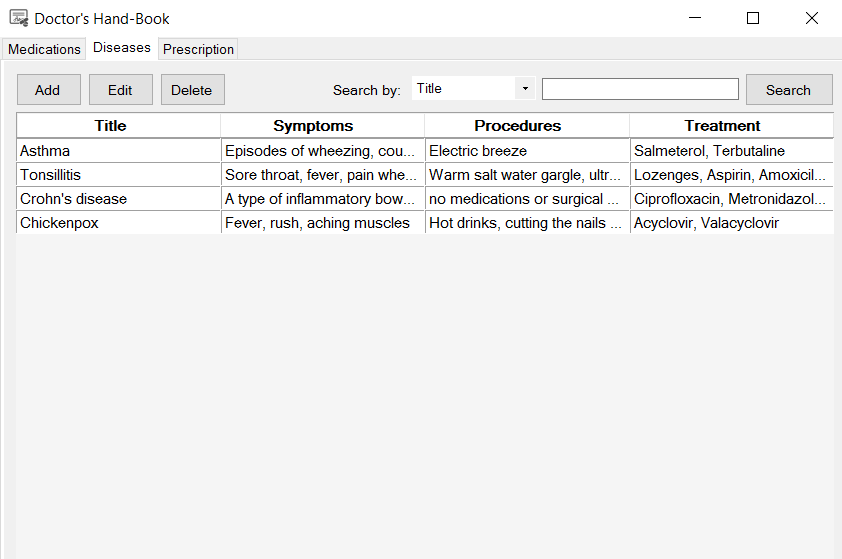


Рисунок 3.6 – Закладка «Diseases»

Додавання хвороби:

1. натиснути «Add»;
2. відкриється діалогове вікно «Add Disease» (рисунок 3.7);

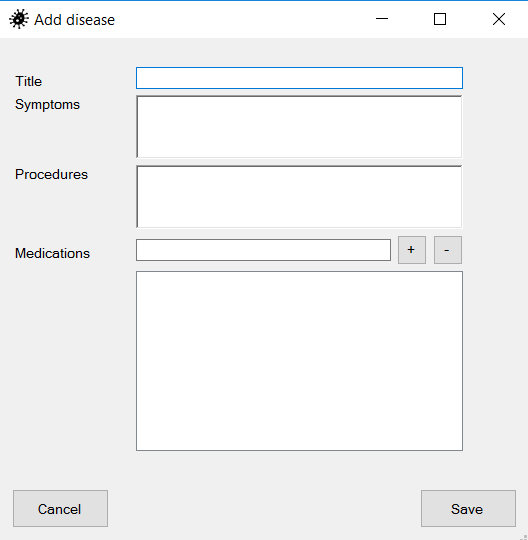


Рисунок 3.7 – Вікно додавання хвороби

1. потрібно заповнити всі поля і натиснути «Save»;
2. при правильному виконанні всіх дії нова хвороба відобразиться у базі на головній формі.

Редагування хвороби:

1. натиснути «Edit»;
2. відкриється діалогове вікно «Edit Disease» (рисунок 3.8);

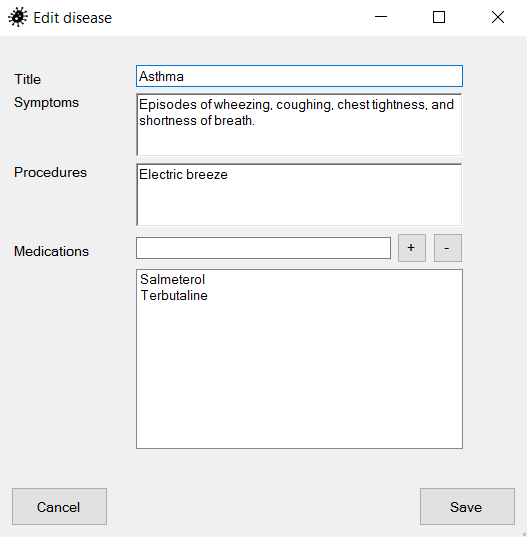


Рисунок 3.8 – Вікно редагування хвороби

1. вносимо необхідні зміни;
2. при правильному виконанні всіх дій хворобу буде змінено, результат можна буде побачити у базі хвороб.

Видалення хвороби:

1. вибрати з бази бажану хворобу;
2. натиснути «Delete»;
3. з’явится вікно для підтверждення видалення (рисунок 3.9).

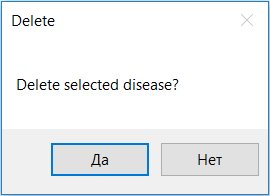


Рисунок 3.9 – Вікно підтвердження вибору

1. підтвердити свій вибір.

Пошук хвороби:

1. на головній формі вибрати критеріїї пошуку: назва, симптоми, процедури, лікування (рисунок 3.5);

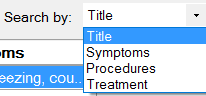


Рисунок 3.10 – Список критеріїв пошуку

1. обовязково заповнити поле пошуку;

У разі успішного пошуку потрібні рядки у базі видяляться. У противному разі, жоден рядок не буде виділеним.

Переходимо до наступної закладки «Prescription» (рисунок 3.10).

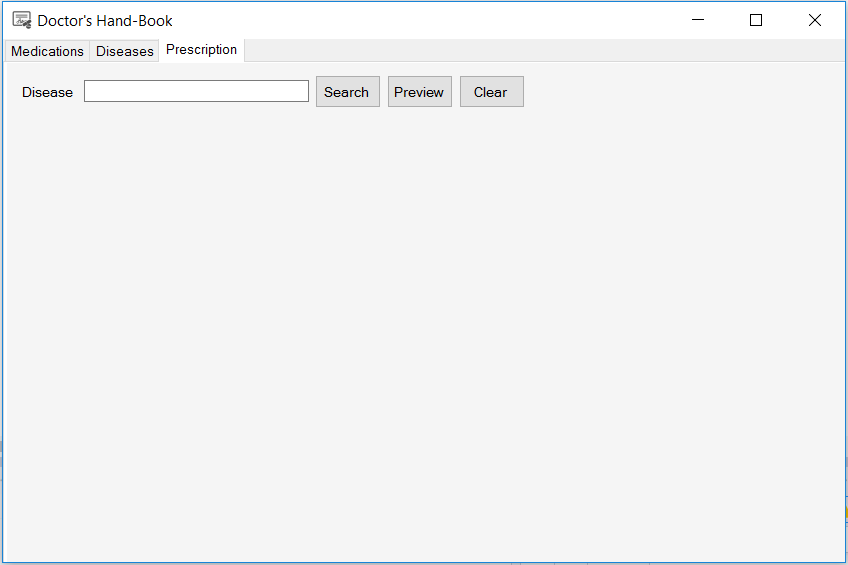


Рисунок 3.10 – закладка «Prescription»

Формування та друк рецепту:

1. заповнити поле пошуку хвороби;
2. натиснути «Search»;
3. з’являться всі процедури і препарати для лікування хвороби. Ті, що наявні на складі у момент пошуку, доступні для додавання їх у рецепт. Вибрати необхідні лікарські засоби зі списку (рисунок 3.11);

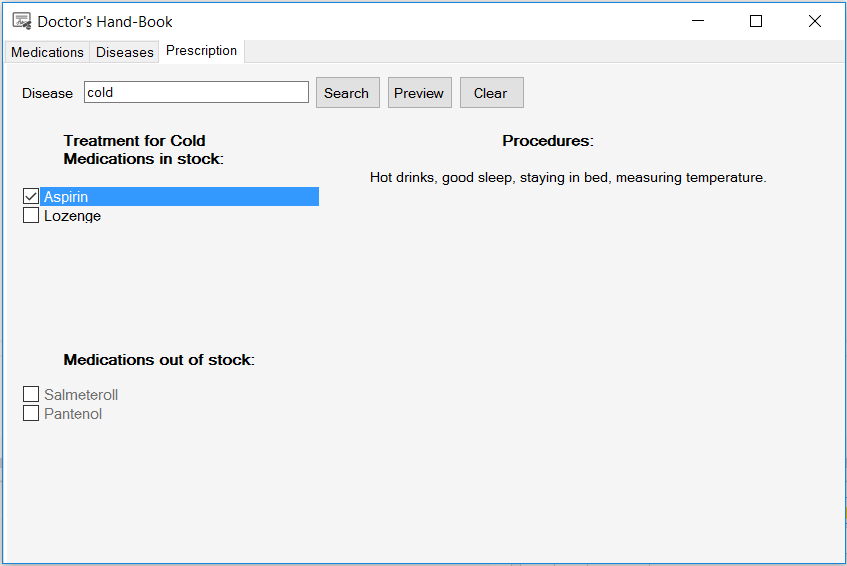


Рисунок 3.11 – Відображення наявних засобів лікування

1. натиснути «Preview». З’явиться нове вікно, де буде сформований рецепт (рисунок 3.12);



Рисунок 3.12 – Вікно зі сформованим рецептом

1. натиснути «Print». З’явиться діалогове вікно для вибору принтера (рисунок 3.13).

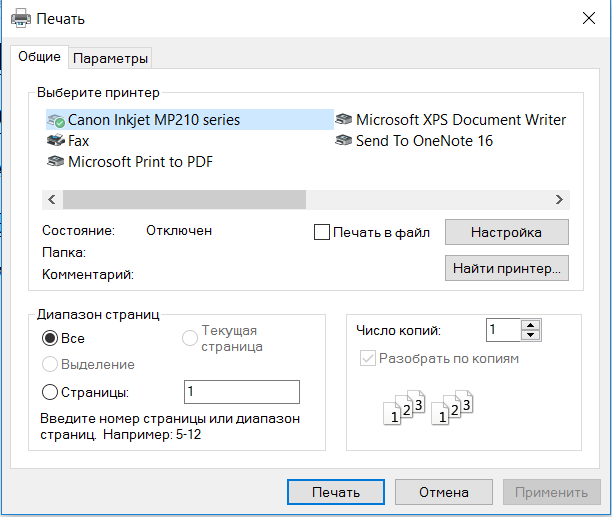


Рисунок 3.13 – Діалогове вікно друку

ВИСНОВКИ

Під час виконнання курсового проекту та супроводження програмного продукту мною були пройдені всі шляхи розробки даної програми, починаючи з постановки задачі до тестування і налагодження програми.

У результаті розробки проекту була створена прикладна програма «Посібник лікаря» на платформі .Net з використанням мови C#.

Посібник містить у собі детальну інформацію щодо лікарських засобів (назву, опис препарату, дозування, тривалість лікування, кількість на складі), а також базу хвороб з характеристиками: назва, симптоми, процедури, список медикаментів для лікування. Присутня функція "Виписати рецепт", що формує та друкує список лікарських засобів, наявних на складі та вибраних користувачем, а також список рекомендованих процедур для лікування вказаного захворювання.

Мною були докладно розглянуті такі аспекти .Net: основні принципи об’єктно-орієнтованого програмування, LINQ запити, XML серіалізація об’єктів, делегати, події, ітератори та індексатори. Був отриманий досвід у проектуванні, реалізації, документуванні повноцінної програми.

У майбутньому планується виконання деяких пунктів для покращення програми, а саме: переробка дизайну і використання бази даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд. Язык программирования C#. Классика Computers Science. 4-е издание= C# Programming Language (Covering C# 4.0), 4th Ed. – СПб.: «Питер», 2012. – 784 с.
2. Бондарев В. М. «Объектно-ориентированное программирование на С#» [Текст]: учеб. пособ / В. М. Бондарев. – Х.: Смит, 2009. – 224 с.;
3. Дейт, К. Дж. Введення у системи баз даних [Текст] / К. Дж. Дейт.- 7-е вид. – М. : Вид. дім «Вільямс», 2001. – 846 с.
4. Троелсен: «Мова програмування С# 2005 та платформа .NET 2.0» [Текст]: учеб. пособ, 3е видання. : Пер. з англ. – Москва. : ООО «И.Д. Вільямс», 2007. – 1168с.
5. Ульман, Дж. Д. Основи систем баз даних [Текст] / Дж. Д. Ульман - М. : Фінанси і статистика, 1983. – 334 с.

Додаток А – Код програми

using System;

namespace DHB.Model

{

//Класс, имеющий информацию о названии, описании, дозах, наличии на складе медикамента

[Serializable]

public class Medication

{

public string Title { get; set; }

public string Description { get; set; }

public decimal DoseUnit { get; set; }

public int DosePortion { get; set; }

public int DosePeriod { get; set; }

public int QuantityInStock { get; set; }

//конструктор без параметров необходим для сериализации

public Medication() { }

//конструктор

public Medication(string title, string description, decimal doseUnit, int dosePortion, int dosePeriod, int qinstock)

{

Title = title;

Description = description;

DoseUnit = doseUnit;

DosePortion = dosePortion;

DosePeriod = dosePeriod;

QuantityInStock = qinstock;

}

}

}

using System.Collections.Generic;

using System.Linq;

namespace DHB.Model

{

//класс, содержащий все методы работы с коллекцией медикаментов

public class MedsManager

{

private List<Medication> \_meds;

private const string MedsFileName = "med.xml";

private readonly DataManager<Medication> \_dataManager = new DataManager<Medication>(MedsFileName);

public MedsManager()

{

\_meds = new List<Medication>();

}

//доступ к полю \_meds для заполнения DataGridView

public List<Medication> GetMeds()

{

return \_meds;

}

//индексатор, дает информацию о наличии медикамента на складе

public bool this[string med]

{

get

{ return \_meds.Any(unit => med == unit.Title && unit.QuantityInStock != 0); }

}

public List<int> GetStock()

{

return \_meds.Select(med => med.QuantityInStock).ToList();

}

//метод добавления в коллекцию

public void AddMed(Medication med)

{

\_meds.Add(med);

}

//метод удаления из коллекции

public void Delete(string selectedMed)

{

for (var i = 0; i < \_meds.Count; i++)

{

if (selectedMed != \_meds[i].Title) continue;

\_meds.Remove(\_meds[i]);

break;

}

}

//получение из xml

public void Init()

{

\_meds = \_dataManager.Init();

}

//запись в xml

public void Update()

{

\_dataManager.Update(\_meds);

}

//метод редактирования

public void Edit(Medication oldMed, Medication newMed)

{

for (var i = 0; i < \_meds.Count; i++)

{

if (\_meds[i].Title != oldMed.Title) continue;

\_meds[i] = newMed;

break;

}

}

}

}

using System;

using System.Collections.Generic;

namespace DHB.Model

{

//Класс, имеющий информацию о названии, симптомах, процедурах, лечении болезни

[Serializable]

public class Disease

{

public string Title { get; set; }

public string Symptoms { get; set; }

public string Procedures { get; set; }

public List<string> Treatment { get; set; }

//конструктор без параметров необходим для сериализации

public Disease() { }

//конструктор

public Disease(string title, string symptoms, string procedures, List<string> treatment)

{

Title = title;

Symptoms = symptoms;

Procedures = procedures;

Treatment = treatment;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

namespace DHB.Model

{

//класс, содержащий все методы работы с коллекцией болезний

internal class DiseasesManager

{

private List<Disease> \_diseases;

private const string DiseaseFileName = "diseases.xml";

private readonly DataManager<Disease> \_dataManager = new DataManager<Disease>(DiseaseFileName);

private List<RecipeMaker> \_recipeMaker;

public DiseasesManager()

{

\_diseases = new List<Disease>();

}

//индексатор,позволяющий получать объект типа Disease по индексу

public Disease this[int position] => \_diseases[position];

//метод для заполнения DataGridView

public List<DiseaseAdapter> GetDisiases()

{

List<DiseaseAdapter> collection = new List<DiseaseAdapter>();

foreach (var dis in \_diseases)

{

string treatment = "";

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

{

if (dis.Treatment.Count - 1 != i)

{

treatment += dis.Treatment[i] + "\n, ";

}

else

{

treatment += dis.Treatment[i] + "\n";

}

}

DiseaseAdapter disease = new DiseaseAdapter(dis.Title, dis.Symptoms, dis.Procedures, treatment);

collection.Add(disease);

}

return collection;

}

//метод добавления в коллекцию

public void Add(Disease disease)

{

\_diseases.Add(disease);

}

//получение из xml

public void Init()

{

\_diseases = \_dataManager.Init();

}

//запись в xml

public void Update()

{

\_dataManager.Update(\_diseases);

}

//метод удаления из коллекции

public void Delete(string selectedDis)

{

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

{

if (selectedDis != \_diseases[i].Title) continue;

\_diseases.Remove(\_diseases[i]);

break;

}

}

//метод редактирования

public void Edit(Disease oldDis, Disease newDis)

{

for (var i = 0; i < \_diseases.Count; i++)

{

if (\_diseases[i].Title != oldDis.Title) continue;

\_diseases[i] = newDis;

break;

}

}

//метод получения списка всех лекарств от данного заболевания

public RecipeMaker GetTreatment(string title)

{

return (from disease in \_diseases where string.Equals(disease.Title, title, StringComparison.CurrentCultureIgnoreCase) select new RecipeMaker(disease.Treatment, disease.Procedures, disease.Title)).FirstOrDefault();

}

}

}

namespace DHB.Model

{

//класс для формирования объекта,который хранит в себе информацию, необходимую для передачи

// коллекции из таких объектов в DataGridView на форму "Disease Control"

internal class DiseaseAdapter

{

public string Title { get; set; }

public string Symptoms { get; set; }

public string Procedures { get; set; }

public string Treatment { get; set; }

public DiseaseAdapter(string title, string symptoms, string procedures, string treatment)

{

Title = title;

Symptoms = symptoms;

Procedures = procedures;

Treatment = treatment;

}

}

}

using System.Collections.Generic;

namespace DHB.Model

{

//класс, хранящий информацию о названии болезни, процедурах и списке лекарст

//используется для формы формирования рецепта

class RecipeMaker

{

public List<string> MedsRecipe { get; set; }

public string ProceduresRecipe { get; set; }

public string DiseaseTitle { get; set; }

public RecipeMaker(List<string> medsRecipe, string proceduresRecipe, string diseaseTitle)

{

MedsRecipe = medsRecipe;

ProceduresRecipe = proceduresRecipe;

DiseaseTitle = diseaseTitle;

}

}

}

using System.Collections.Generic;

using System.IO;

using System.Xml.Serialization;

namespace DHB.Model

{

//класс для сохранения и добывания данных из xml файла при помощи сериализации

internal class DataManager<T>

{

private readonly string \_fileName;

public DataManager(string file)

{

\_fileName = file;

}

//десериализация

public List<T> Init()

{

List<T> collection = new List<T>();

if (File.Exists(\_fileName))

{

XmlSerializer formatter = new XmlSerializer(typeof(List<T>));

using (FileStream fs = new FileStream(\_fileName, FileMode.Open))

{

collection = (List<T>)formatter.Deserialize(fs);

}

}

return collection;

}

//сериализация

public void Update(List<T> collection)

{

XmlSerializer formatter = new XmlSerializer(typeof(List<T>));

using (FileStream fs = new FileStream(\_fileName, FileMode.Create))

{

formatter.Serialize(fs, collection);

}

}

}

}

using System;

using System.Windows.Forms;

namespace DHB.View

{

//Главная форма

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

//добавление в закладки формы контролов

private void Form1\_Load(object sender, EventArgs e)

{

var form = new MedsControl();

form.TopLevel = false;

form.Visible = true;

form.FormBorderStyle = FormBorderStyle.None;

tabMedications.Controls.Add(form);

var form2 = new DiseasesControl();

form2.TopLevel = false;

form2.FormBorderStyle = FormBorderStyle.None;

form2.Visible = true;

tabDiseases.Controls.Add(form2);

var form3 = new Prescription();

form3.TopLevel = false;

form3.FormBorderStyle = FormBorderStyle.None;

form3.Visible = true;

tabPrescription.Controls.Add(form3);

}

}

}

using System;

using System.Windows.Forms;

using DHB.Model;

namespace DHB.View

{

//Форма, позволяющая управлять коллекцией медикаментов, сохраненной в файле(добавлять,редактировать,удалять)

//а также искать медикамент по названию или описанию

public partial class MedsControl : Form

{

private readonly MedsManager \_medsManager;

private Medication \_oldMed;

public MedsControl()

{

InitializeComponent();

\_medsManager = new MedsManager();

}

//Обновление DataGridView

private void RefreshDataGrid()

{

try

{

\_medsManager.Update();

\_medsManager.Init();

MedsDataGrid.DataSource = \_medsManager.GetMeds();

}

catch (Exception ex)

{

MessageBox.Show(@"Error: " + ex.Message, @"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//Формирование DataGridView при загрузке формы

private void Meds\_Control\_Load(object sender, EventArgs e)

{

\_medsManager.Init();

try

{

MedsDataGrid.DataSource = \_medsManager.GetMeds();

}

catch (Exception ex)

{

MessageBox.Show(@"Error: " + ex.Message, @"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

MedsDataGrid.Columns[0].HeaderText = @"Title";

MedsDataGrid.Columns[2].HeaderText = @"Quantity of intake";

MedsDataGrid.Columns[3].HeaderText = @"Frequence of intake";

MedsDataGrid.Columns[4].HeaderText = "Period of \nintake (days)";

MedsDataGrid.Columns[5].HeaderText = @"In stock";

MedsDataGrid.Columns[0].DefaultCellStyle.Alignment = DataGridViewContentAlignment.TopLeft;

//MedsDataBase.Columns[0].SortMode = DataGridViewColumnSortMode.Automatic;

FitToContent();

cmbBoxSearch.SelectedIndex = 0;

}

//Расположение данных в таблице

private void FitToContent()

{

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

{

if (i != 1)

{

MedsDataGrid.Columns[i].AutoSizeMode = DataGridViewAutoSizeColumnMode.DisplayedCells;

continue;

}

MedsDataGrid.Columns[i].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

}

//Обновление документа с данными о медикаментах при закрытии формы

private void Meds\_Control\_FormClosing(object sender, FormClosingEventArgs e)

{

try

{

\_medsManager.Update();

}

catch (Exception ex)

{

MessageBox.Show(@"Error: " + ex.Message, @"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//Удаление выбранного медикамента из файла

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

{

if (MedsDataGrid.CurrentRow == null) return;

var result = MessageBox.Show(@"Delete selected medication?", @"Delete", MessageBoxButtons.YesNo);

if (result != DialogResult.Yes) return;

var selectedMed = MedsDataGrid.CurrentRow.Cells[0].Value.ToString();

\_medsManager.Delete(selectedMed);

\_medsManager.Update();

RefreshDataGrid();

}

//Редактирование выбранного медикамента

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

{

if (MedsDataGrid.CurrentRow == null) return;

var title = MedsDataGrid.CurrentRow.Cells[0].Value.ToString();

var description = MedsDataGrid.CurrentRow.Cells[1].Value.ToString();

var doseUnit = Convert.ToInt32(MedsDataGrid.CurrentRow.Cells[2].Value);

var dosePortion = Convert.ToInt32(MedsDataGrid.CurrentRow.Cells[3].Value);

var dosePeriod = Convert.ToInt32(MedsDataGrid.CurrentRow.Cells[4].Value);

var quantity = Convert.ToInt32(MedsDataGrid.CurrentRow.Cells[5].Value);

\_oldMed = new Medication(title, description, doseUnit, dosePortion, dosePeriod, quantity);

var fOptionEdit = new MedOptionEdit(\_oldMed);

fOptionEdit.FormClosed += FOptionsEdit\_FormClosed;

fOptionEdit.ShowDialog();

}

//Получение нового медикамента из формы редактирования. Замена выбранного медикамента на новый. Обновление DataGridView

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

{

var fOptionsEdit = (MedOptionEdit)sender;

if (fOptionsEdit.NewMed != null)

{

\_medsManager.Edit(\_oldMed, fOptionsEdit.NewMed);

}

RefreshDataGrid();

}

//Получение нового медикамента из формы добавления. Добавлениенового медикамента в документ. Обновление DataGridView

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

{

var fOptionsAdd = (MedOptionAdd)sender;

if (fOptionsAdd.Med != null)

{

\_medsManager.AddMed(fOptionsAdd.Med);

}

RefreshDataGrid();

}

//Создание формы добавления медикамента

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

{

MedOptionAdd fOptionsAdd = new MedOptionAdd();

fOptionsAdd.FormClosed += FOptionsAdd\_FormClosed;

fOptionsAdd.ShowDialog();

}

//Поиск медикамента в MedsDataBase, выделение найденой строки

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

{

var limit = cmbBoxSearch.SelectedIndex;

if (tBoxMedSearch.Text.Trim() == "") return;

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

{

MedsDataGrid.Rows[i].Selected = false;

if (MedsDataGrid.Rows[i].Cells[limit].Value == null) continue;

if (MedsDataGrid.Rows[i].Cells[limit].Value.ToString().Contains(tBoxMedSearch.Text.Trim()))

{

MedsDataGrid.Rows[i].Selected = true;

}

}

}

//Очищение строки поиска при изменении критерия поиска медикамента в комбобоксе

private void CmbBoxSelectedIndexChanged(object sender, EventArgs e)

{

tBoxMedSearch.Clear();

MedsDataGrid.ClearSelection();

}

}

}

using System;

using System.Windows.Forms;

using DHB.Model;

namespace DHB.View

{

//форма добавления болезни

public partial class MedOptionAdd : Form

{

public Medication Med { get; set; }

public MedOptionAdd()

{

InitializeComponent();

}

//сохранение новой болезни

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

{

if (string.IsNullOrEmpty(tBoxMedTitle.Text) || string.IsNullOrEmpty(rtBoxMedDescription.Text))

{

MessageBox.Show(@"Complete all fields!", @"Adding failed");

}

else

{

var title = tBoxMedTitle.Text.Trim();

var description = rtBoxMedDescription.Text.Trim();

var doseUnit = Convert.ToInt32(numUDUnit.Value);

var dosePortion = Convert.ToInt32(numUDPortion.Value);

var dosePeriod = Convert.ToInt32(numUDPeriod.Value);

var quantity = Convert.ToInt32(numUDinStock.Value);

Med = new Medication(title, description, doseUnit, dosePortion, dosePeriod, quantity);

Close();

}

}

//закрыте формы по кнопке "отмена"

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

{

Close();

}

//закрыте формы по кнопке "esc"

private void MedOptionAdd\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

Close();

}

}

}

}

using System;

using System.Windows.Forms;

using DHB.Model;

namespace DHB.View

{

//форма добавления медикамента

public partial class MedOptionAdd : Form

{

public Medication Med { get; set; }

public MedOptionAdd()

{

InitializeComponent();

}

//сохранение нового медикамента

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

{

if (string.IsNullOrEmpty(tBoxMedTitle.Text) || string.IsNullOrEmpty(rtBoxMedDescription.Text))

{

MessageBox.Show(@"Complete all fields!", @"Adding failed");

}

else

{

var title = tBoxMedTitle.Text.Trim();

var description = rtBoxMedDescription.Text.Trim();

var doseUnit = Convert.ToInt32(numUDUnit.Value);

var dosePortion = Convert.ToInt32(numUDPortion.Value);

var dosePeriod = Convert.ToInt32(numUDPeriod.Value);

var quantity = Convert.ToInt32(numUDinStock.Value);

Med = new Medication(title, description, doseUnit, dosePortion, dosePeriod, quantity);

Close();

}

}

//закрыте формы по кнопке "отмена"

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

{

Close();

}

//закрыте формы по кнопке "esc"

private void MedOptionAdd\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

Close();

}

}

}

}

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using DHB.Model;

namespace DHB.View

{

//Форма, позволяющая управлять коллекцией болезней, сохраненной в файле(добавлять,редактировать,удалять)

//а также искать болезнь по названию, симптомам и процедурам

public partial class DiseasesControl : Form

{

private readonly DiseasesManager \_diseaseManager;

private Disease \_oldDisease;

public DiseasesControl()

{

InitializeComponent();

\_diseaseManager = new DiseasesManager();

}

//Обновление DataGridView

private void Diseases\_Control\_Load(object sender, EventArgs e)

{

\_diseaseManager.Init();

try

{

DiseasesDataGrid.DataSource = \_diseaseManager.GetDisiases();

}

catch (Exception ex)

{

MessageBox.Show(@"Error: " + ex.Message, @"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

FitToContent();

cmbBoxSearch.SelectedIndex = 0;

}

//изменение расположения колонок DataGridView

private void FitToContent()

{

DiseasesDataGrid.Columns[0].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

//Удаление болезни

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

{

if (DiseasesDataGrid.CurrentRow == null) return;

var dialogResult = MessageBox.Show(@"Delete selected disease?", @"Delete", MessageBoxButtons.YesNo);

if (dialogResult != DialogResult.Yes) return;

var selectedMed = DiseasesDataGrid.CurrentRow.Cells[0].Value.ToString();

\_diseaseManager.Delete(selectedMed);

\_diseaseManager.Update();

RefreshDatabase();

}

//Обновление DataGridView

private void RefreshDatabase()

{

\_diseaseManager.Update();

\_diseaseManager.Init();

try

{

DiseasesDataGrid.DataSource = \_diseaseManager.GetDisiases();

}

catch (Exception ex)

{

MessageBox.Show(@"Error: " + ex.Message, @"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//получение списка лекарств в нужном нам виде, парсинг строки

private static List<string> GetTreatment(string current)

{

var treatm = new List<string>();

var med = "";

for (int i = 0; i < current.Length; i++)

{

if (current[i] != ',')

{

med += current[i];

}

if (current[i] == ',' || i == current.Length - 1)

{

treatm.Add(med.Trim());

med = "";

}

}

return treatm;

}

//добавление болезни

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

{

var formAdd = new DiseaseOptionAdd();

formAdd.FormClosed += FormAdd\_FormClosed;

formAdd.ShowDialog();

}

//получение объекта типа Disease из формы DiseaseOptionAdd

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

{

var formAdd = (DiseaseOptionAdd)sender;

if (formAdd.Disease != null)

{

\_diseaseManager.Add(formAdd.Disease);

}

RefreshDatabase();

}

//редактирование болезни

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

{

if (DiseasesDataGrid.CurrentRow != null)

{

var title = DiseasesDataGrid.CurrentRow.Cells[0].Value.ToString();

var sympt = DiseasesDataGrid.CurrentRow.Cells[1].Value.ToString();

var proced = DiseasesDataGrid.CurrentRow.Cells[2].Value.ToString();

var cur = DiseasesDataGrid.CurrentRow.Cells[3].Value.ToString();

var treatment = GetTreatment(cur);

\_oldDisease = new Disease(title, sympt, proced, treatment);

}

DiseaseOptionEdit formEdit = new DiseaseOptionEdit(\_oldDisease);

formEdit.FormClosed += FormEdit\_FormClosed;

formEdit.ShowDialog();

}

//получение объекта типа Disease из формы DiseaseOptionEdit

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

{

var formEdit = (DiseaseOptionEdit)sender;

if (formEdit.NewDisease != null)

{

\_diseaseManager.Edit(\_oldDisease, formEdit.NewDisease);

}

RefreshDatabase();

}

//очистка контролов при смене критерия поиска

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

{

tBoxDisSearch.Clear();

DiseasesDataGrid.ClearSelection();

}

//поиск по критериям

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

{

var limit = cmbBoxSearch.SelectedIndex;

if (string.IsNullOrEmpty(tBoxDisSearch.Text)) return;

for (var i = 0; i < DiseasesDataGrid.RowCount; i++)

{

DiseasesDataGrid.Rows[i].Selected = false;

if (DiseasesDataGrid.Rows[i].Cells[limit].Value == null) continue;

if (DiseasesDataGrid.Rows[i].Cells[limit].Value.ToString().Contains(tBoxDisSearch.Text.Trim()))

{

DiseasesDataGrid.Rows[i].Selected = true;

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using DHB.Model;

namespace DHB.View

{

//Форма добавления болезни

public partial class DiseaseOptionAdd : Form

{

//Экземпляр класса Disease

public Disease Disease { get; set; }

//Коллекция медикаментов, в которую добавляется новый медикамент

private readonly List<string> \_meds = new List<string>();

//Перемення, хранящая название старого медикамента для удаления

private string \_oldMed;

public DiseaseOptionAdd()

{

InitializeComponent();

}

//Сохранение медикамента

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

{

if (string.IsNullOrEmpty(tBoxDiseaseTitle.Text) || string.IsNullOrEmpty(rTBoxDiseaseSympt.Text) ||

string.IsNullOrEmpty(rTBoxDiseaseProced.Text))

{

MessageBox.Show(@"Complete all fields!", @"Adding failed");

}

else

{

var title = tBoxDiseaseTitle.Text.Trim();

var sympt = rTBoxDiseaseSympt.Text.Trim();

var proced = rTBoxDiseaseProced.Text.Trim();

Disease = new Disease(title, sympt, proced, \_meds);

Close();

}

}

//Кнопка "+" для добавления новых медикаментов в список

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

{

if (!string.IsNullOrEmpty(tBoxDiseaseMeds.Text))

{

if (lBoxDiseaseMed.Items.Contains(tBoxDiseaseMeds.Text.Trim()))

{

tBoxDiseaseMeds.Clear();

return;

}

lBoxDiseaseMed.Items.Remove(\_oldMed);

\_oldMed = "";

\_meds.Add(tBoxDiseaseMeds.Text.Trim());

lBoxDiseaseMed.Items.Add(tBoxDiseaseMeds.Text.Trim());

lBoxDiseaseMed.Refresh();

}

}

//Кнопка "-" для удаления медикамента из списка

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

{

if (lBoxDiseaseMed.SelectedIndex != -1)

{

tBoxDiseaseMeds.Clear();

\_meds.RemoveAt(lBoxDiseaseMed.SelectedIndex);

lBoxDiseaseMed.Items.RemoveAt(lBoxDiseaseMed.SelectedIndex);

}

}

//Метод динамического добавления медикамента из списка в текстбокс для возможности редактирования

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

{

if (lBoxDiseaseMed.SelectedItem != null)

{

tBoxDiseaseMeds.Text = lBoxDiseaseMed.SelectedItem.ToString();

\_oldMed = lBoxDiseaseMed.SelectedItem.ToString();

}

}

//Закрытие формы по нажатиею Esc

private void DiseaseOptionAdd\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

Close();

}

}

//Закрытие формы нажатием на кнопку отмены

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

{

Close();

}

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

{

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

using DHB.Model;

namespace DHB.View

{

//форма редактированния болезни

public partial class DiseaseOptionEdit : Form

{

public Disease NewDisease { get; set; }

private string \_oldMed;

//заполениния поле формы информацией о старой болезни

public DiseaseOptionEdit(Disease dis)

{

InitializeComponent();

tBoxDiseaseTitle.Text = dis.Title;

rTBoxDiseaseSympt.Text = dis.Symptoms;

rTBoxDiseaseProced.Text = dis.Procedures;

foreach (var x in dis.Treatment)

{

lBoxDiseaseMed.Items.Add(x);

}

}

//закрытие по нажатию "отмена"

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

{

Close();

}

//сохранение измененной болезни

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

{

var title = tBoxDiseaseTitle.Text.Trim();

var sympt = rTBoxDiseaseSympt.Text.Trim();

var proced = rTBoxDiseaseProced.Text.Trim();

var meds = new List<string>();

meds.AddRange(lBoxDiseaseMed.Items.Cast<string>());

NewDisease = new Disease(title, sympt, proced, meds);

Close();

}

//добавление медикаментов в список лечения

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

{

if (!string.IsNullOrEmpty(tBoxDiseaseMeds.Text))

{

if (lBoxDiseaseMed.Items.Contains(tBoxDiseaseMeds.Text.Trim()))

{

tBoxDiseaseMeds.Clear();

return;

}

lBoxDiseaseMed.Items.Remove(\_oldMed);

\_oldMed = "";

lBoxDiseaseMed.Items.Add(tBoxDiseaseMeds.Text.Trim());

lBoxDiseaseMed.Refresh();

tBoxDiseaseMeds.Clear();

}

}

//удаления медикаментов из списка лечения

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

{

if (lBoxDiseaseMed.SelectedIndex != -1)

{

lBoxDiseaseMed.Items.RemoveAt(lBoxDiseaseMed.SelectedIndex);

tBoxDiseaseMeds.Clear();

}

}

//Закрытие формы по нажатиею Esc

private void DiseaseOptionEdit\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Escape)

{

Close();

}

}

//Метод динамического добавления медикамента из списка в текстбокс для возможности редактирования

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

{

if (lBoxDiseaseMed.SelectedItem != null)

{

tBoxDiseaseMeds.Text = lBoxDiseaseMed.SelectedItem.ToString();

\_oldMed = lBoxDiseaseMed.SelectedItem.ToString();

}

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Web.UI.WebControls;

using System.Windows.Forms;

using DHB.Model;

namespace DHB.View

{

//форма формирования рецепта

public partial class Prescription : Form

{

private readonly MedsManager \_medsManager;

private readonly DiseasesManager \_diseaseManager;

//создание коллекции строк для хранения списка лекарсв

private List<string> \_recipe;

private string \_name = "";

//создание объектов классов MedsManager, DiseasesManager

public Prescription()

{

InitializeComponent();

\_diseaseManager = new DiseasesManager();

\_medsManager = new MedsManager();

}

//дисериализация классов MedsManager, DiseasesManager

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

{

try

{

\_diseaseManager.Init();

\_medsManager.Init();

}

catch (Exception ex)

{

MessageBox.Show(@"Error: " + ex.Message, @"Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//очистка контролов

private void ClearConrols()

{

chLBMeds.Items.Clear();

chLBOut.Items.Clear();

pnlPresc.Controls.Clear();

lblHeaderD.Visible = false;

lblHeaderO.Visible = false;

lblHeaderP.Visible = false;

lblProced.Visible = false;

}

//поиск лекрств

//вывод лекарств в чекбокслисты, в зависимоти от того, есть ли они на складе

//вывод процедур

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

{

var outOfStock = true;

var search = tBoxDiseaseSearch.Text.Trim();

if (string.IsNullOrEmpty(search)) return;

var recipe = \_diseaseManager.GetTreatment(search);

lblHeaderD.Visible = true;

chLBMeds.Visible = true;

lblHeaderD.Text = "Treatment: ";

if (recipe != null)

{

foreach (var med in recipe.MedsRecipe)

{

if (!\_medsManager[med])

{

chLBOut.Visible = true;

lblHeaderO.Visible = true;

lblHeaderO.Text = @"Medications out of stock:";

var l = new ListItem { Value = med };

chLBOut.Items.Add(l);

}

else

{

outOfStock = false;

\_name = recipe.DiseaseTitle;

if (string.IsNullOrEmpty(med)) return;

lblHeaderD.Text = @"Treatment for " + recipe.DiseaseTitle + "\nMedications in stock:";

lblHeaderD.Visible = true;

chLBMeds.Visible = true;

var l = new ListItem { Value = med };

chLBMeds.Items.Add(l);

}

}

lblHeaderP.Text = @"Procedures: ";

lblHeaderP.Visible = true;

lblProced.Text = recipe.ProceduresRecipe;

lblProced.Visible = true;

if (outOfStock)

{

lblHeaderD.Text = @"No medications in stock";

lblHeaderD.Visible = true;

}

}

}

//формирование html файла для загрузки в WebBrowser

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

{

if (string.IsNullOrEmpty(tBoxDiseaseSearch.Text.Trim())) return;

\_recipe = new List<string>();

foreach (ListItem item in chLBMeds.CheckedItems)

{

\_recipe.Add(item.Text);

}

var currentPath = Directory.GetCurrentDirectory();

var pathWithName = currentPath + "\\recipe.html";

var headlineRecipe = "<h1><u> Recepie for " + \_name + "</u></h1>";

const string headlineMeds = "<h2><u>Medications:</u></h2>";

const string headProcedures = "<h2><u>Procedures:</u></h2>";

var listOfMeds = "";

if (\_recipe.Count != 0)

{

listOfMeds += "<ul>";

listOfMeds = \_recipe.Aggregate(listOfMeds, (current, t) => current + ("<li>" + t + "</li>"));

listOfMeds += "</ul>";

File.WriteAllText(pathWithName, headlineRecipe + headlineMeds + listOfMeds + headProcedures + lblProced.Text);

}

else

{

File.WriteAllText(pathWithName, headlineRecipe + headProcedures + lblProced.Text);

}

var preview = new Recipe(pathWithName);

preview.ShowDialog();

}

//очистка контролов

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

{

tBoxDiseaseSearch.Clear();

ClearConrols();

}

}

}

using System;

using System.Windows.Forms;

namespace DHB.View

{

//форма, выводящая сформированый рецепт в webbrowser

public partial class Recipe : Form

{

private readonly string \_adress;

public Recipe(string adress)

{

\_adress = adress;

InitializeComponent();

}

//загрузка адресса

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

{

webBrRecepie.Navigate(\_adress);

}

//закрытие

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

{

Close();

}

//печать

private void btnPrint\_Click\_1(object sender, EventArgs e)

{

webBrRecepie.Print();

}

}

}