Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліні " Об’єктно-орієнтоване програмування "

«Довідник лікаря»

Керівник , професор Бондарєв В. М.

Студент гр. ПЗПІ-19-4 Червінська А. Л.

Комісія:

ст. викладач Черепанова Ю. Ю.

професор Бондарєв В. М.

доцент Побіженко І. О.

Харків 2020

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

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

Спеціальність: ***121***  ***Інженерія програмного забезпечення***

Освітня програма: ***Програмна інженерія***

Курс 1 .Група *ПЗПІ-19 - 4*. Семестр *2* .

***ЗАВДАННЯ***

***на курсовий проект студента***

*Червінська Анастасія Любомирівна*

1 Тема проекту: Довідник лікаря

2 Термін здачі студентом закінченого проекту: ***“19” - червня - 2020 р***.

3 Вихідні дані до проекту:

*Специфікація програми, методичні вказівки до виконання курсової роботи*

4 Зміст розрахунково-пояснювальної записки:

*Вступ,* *специфікація програми, проектна специфікація, інструкція користувача, висновки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5 Перелік графічного матеріалу:

\_\_ *Схема об’єктної моделі, діаграми класів, алгоритми та приклади екранних форм* \_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_КАЛЕНДАРНИЙ ПЛАН

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 21-02-2020 р. |
| 2 | Формулювання вимог до програми | 21-02-2020 – 31-03-2020 р. |
| 3 | Розробка підсистеми формування рецептів. | 31-03-2020 – 07-04-2020 р. |
| 4 | Розробка підсистеми коригування запасів. | 07-04-2020 – 14-04-2020 р. |
| 5 | Розробка функцій головних користувачів. | 14-04-2020 – 21-04-2020 р. |
| 6 | Розробка функцій зберігання та завантаження даних | 21-04-2020 – 30-04-2020 р. |
| 7 | Тестування і доопрацювання розробленої програмної системи. | 1-05-2020 – 15-05-2020 р. |
| 8 | Оформлення пояснювальної записки, додатків, графічного матеріалу | 15-05-2020 – 05-06-2020 р. |
| 9 | Захист | 01-06-2018 – 19-06-2020 р. |

Студент Червінська Анастасія Любомирівна

Керівник Бондарєв Володимир Михайлович

« 21 » лютого 2020 р.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 42 с., 15 рис., 4 додатки, 5 джерел.

ВІДОМІСТЬ, ДОКТОР, МЕДСЕСТРА, МЕДИКАМЕНТ, РЕЦЕПТ, КЛАС, МОВА ПРОГРАМУВАННЯ C#, РЕЦЕПТ, ОБ’ЕКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, ПРОГРАМА, ПЛАТФОРМА .NET, СКЛАД МЕДИКАМЕНТІВ, БАЗА ХВОРОБ.

Метою роботи є розробка програми «Довідник лікаря» на засадах об'єктно-орієнтованого програмування.

Методи розробки базуються на використанні середи розробки Microsoft Visual Studio 2019, Windows Forms, платформи .NET Framework 4.7. 2, мови програмування C#.

В результаті отримана програма під назвою «Довідник лікаря», яка дозволяє зберігати список медикаментів та хвороб, характеристики кожного медикаменту, такі як: назва, кількість, одиниця виміру, взаємозамінність; характеристики кожної хвороби, такі як: назва, порція медикаменту та симптоми. Є можливість утворювати нові рецепти, додавати, видаляти та редагувати медикамент. Також, є функція, яка дозволяє дивитися звіт використаних медикаментів.

ЗМІСТ

Перелік умовних скорочень 5

Вступ 6

1 Специфікація програми 7

1.1 Користувачі програми 7

1.2 Функції програми................................................................................7

1.3 Функції лікарів.....................................................................................8

1.3.1 Функція «Вхід у програму».............................................................8

1.3.2 Функція «Вибір симптомів»............................................................9

1.3.3 Функція «Пошук хвороби за симптомами».................................10

1.3.4 Функція «Редагування рецепту»...................................................11

1.4 Функції старшої медсестри..............................................................12

1.4.1 Функція «Вхід у програму»...........................................................12

1.4.2 Функція «Зміна медикаменту».....................................................14

1.4.3 Функція «Звіт використаних медикаментів»...............................15

2 Проектна специфікація........................................................................17

2.1 Архітектура........................................................................................17

2.2 Структура проекту............................................................................17

2.3 Об'єктна модель. Опис......................................................................18

2.4 Сховище та доступ даних.................................................................19

3 Інструкція користувача........................................................................20

Висновки..................................................................................................23

Перелік джерел посилання....................................................................24

Додаток А Код бібліотеки.....................................................................25

Додаток Б Код тестів програми.............................................................30

Додаток В Код DoctorApp.......................................................................31

Додаток Г Код NurseApp.........................................................................37

ВСТУП

Важко уявити наше життя без лікарів. Кожна людина хоч раз у житті зверталась до людей з цією професією. Усім лікарям довелося пройти надзвичайно складний період навчання для того, щоб стати справжніми професіоналами та спеціалістами у своїй сфері діяльності. Вони кожен день рятують тисячі життів, лікують мільйони, і після кожного пацієнта їм доводиться заповнювати багато рецептів, бланків, довідок, карток і заяв. І в результаті 90% їх роботи - письмова. Лікарі не повинні витрачати свої дорогоцінні часи роботи на заповнення рецептів замість того, щоб лікувати пацієнтів.

Розроблена програма полегшить роботу лікарям та медсестрам, а пацієнтам покращить розуміння виписаного рецепта. У цій програмі лікар зможе виписувати рецепт, в залежності від вибраних симптомів та наявності медикаментів на складі, а старша медсестра - корегувати запаси ліків та переглядати звіти. Доступ усіх користувачів програми здійснюється через пароль.

Окрім розробки вище описаної програмної системи метою даної курсової роботи є закріплення знань, набутих при вивчені дисципліни "Об'єктно-орієнтоване програмування", освоїти мову програмування C#, розвити навички роботи в Windows Forms.

Задачі роботи:

* Стійкість програми;
* Забезпечення цілісності даних;
* Функціональна повнота;
* Розробка системи зчитування та запису даних;
* Робота з користувачем;
* Тестування готової програми;
* Розробка логіки додавання та видалення інформації;

1 СПЕЦИФІКАЦІЯ ПРОГРАМИ

1.1 Користувачі програми

У проекті є дві категорії користувачів, яким необхідні різні функції програми - це лікар і старша медсестра. Лікар відшукує симптоми, визначає хворобу, після автоматичної перевірки наявності медикаментів формує рецепт. Старша медсестра додає нові медикаменти, змінює їх кількість і отримує звіт використаних ліків. Належність користувача до лікарів або старшої медсестри визначається в момент входу в програму. Питання безпеки в програмі не вирішується.

1.2 Функції програми

Програма надає лікареві наступні можливості:

1. Вхід у програму.
2. Вибір симптомів.
3. Пошук хвороби за симптомами.
4. Редагування рецепта.
5. Вихід з програми.

Програма надає старшій медсестрі наступні можливості:

1. Вхід у програму.
2. Зміна медикаменту.
3. Додавання медикаменту.
4. Видалення медикаменту.
5. Звіт по використанню лікарських препаратів.
6. Вихід з програми.

Старшій медсестрі не доступні всі функції лікаря, так само лікарю не доступні функції медсестри.

Загальні функції:

1. Завантаження даних з постійної пам'яті в оперативну.
2. Збереження поточного стану даних в постійній пам'яті.
3. Генерація тестових даних.

1.3 Функції лікарів

1.3.1 Функція «Вхід у програму»

Лікарі входять в програму за допомогою імені та пароля. Імена і паролі лікарів зберігаються в даних програми. У лікарів повинні бути паролі, щоб медсестра, інші лікарі або пацієнти не змогли увійти в програму під іншим ім'ям. Медсестри або пацієнти не можуть виписати рецепт через невідповідність освіти.

Основний сценарій:

1. У програмі відразу відкривається вікно входу для лікаря (див. рис. 1.1).
2. Після вибору імені лікаря та введення правильного пароля, треба натиснути кнопку «Ready».
3. Введені дані перевіряються, і відкривається головне вікно лікаря (див. рис. 1.2).
4. Якщо дані не пройшли перевірку, то відкривається вікно-повідомлення (див. рис. 1.3). Після закриття вікна можна спробувати увійти знову. Число спроб не обмежена.

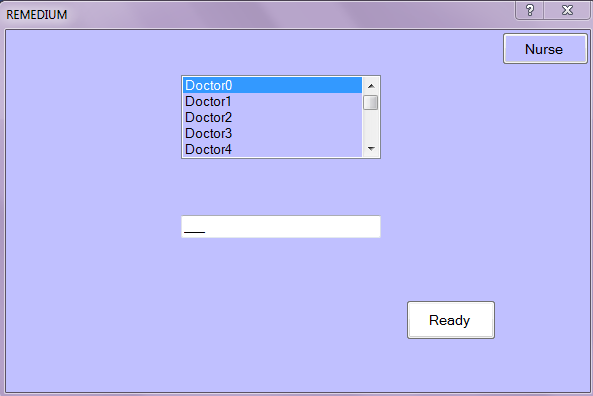


Рисунок 1.1 - Вікно входу

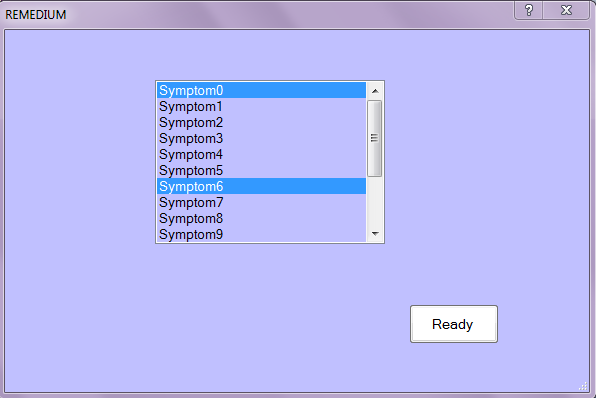


Рисунок 1.2 - Вікно, де лікар вибирає симптоми

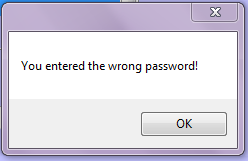


Рисунок 1.3 - Повідомлення при введені некоректних даних

1.3.2 Функція «Вибір симптомів»

На рис. 1.2 лікар зазначає симптоми. У списку є всі симптоми хвороб. Вони впорядковані.

Сценарій:

1. Лікар натискає на ті симптоми, які йому потрібні.
2. Натискає кнопку «Ready».
3. Відкривається вікно з переліком хвороб (див. рис. 1.4). Список залежить від вибраних симптомів.
4. Якщо лікар не вибрав жодного симптоми, то відкривається вікно повідомлення про це (див. рис. 1.5).

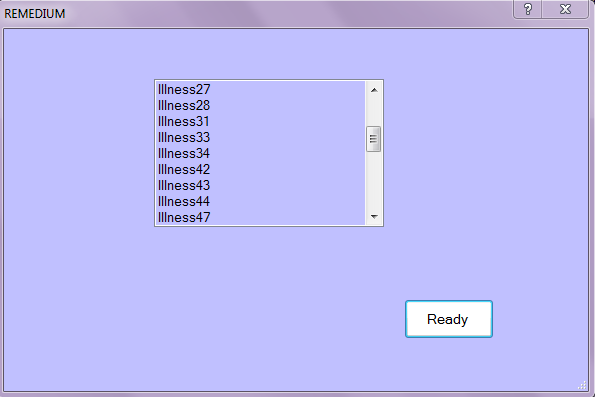


Рисунок 1.4 - Вікно, де лікар вибирає хворобу

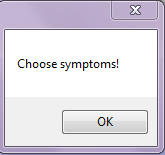


Рисунок 1.5 - Повідомлення про некоректні данні

1.3.3 Функція «Пошук хвороби за симптомами»

Так як деякі хвороби мають однакові або схожі симптоми, лікарю буде бачити список хвороб, які можуть бути з цими симптомами. Хвороби впорядковані. Лікар спираючись на свою освіту і досвід вибирає хворобу.

Сценарій «Пошук хвороби»:

1. Зі списку хвороб лікар вибирає одну хворобу та натискає на неї.
2. Далі лікар натискає кнопку «Ready».
3. Після автоматичної перевірки ліків, відкривається вікно рецепту (див. рис. 1.6).

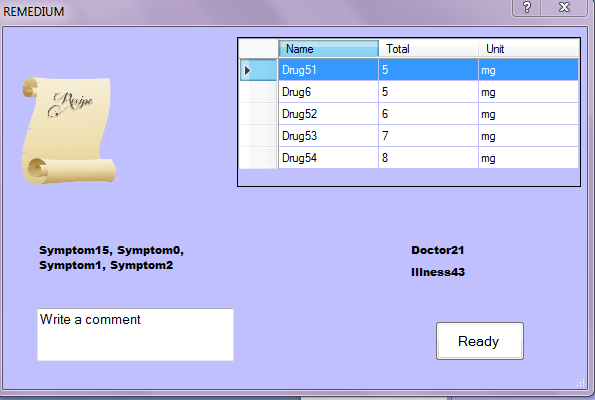


Рисунок 1.6 - Вікно для формування рецепту

1.3.4 Функція «Редагування рецепта»

Перед збереженням рецепт повинен бути показаний лікарю. Кожен випадок індивідуальний, тому лікар може редагувати рецепт. У рецепті написано ім'я лікаря, поточна хвороба та її симптоми. Поряд з медикаментом зазначена необхідна кількість та його одиниця виміру.

Основний сценарій:

1. Лікар вибирає медикаменти, які необхідні.
2. Він пише свій коментар.
3. Натискає кнопку «Ready». Текстова копія звіту зберігається у пам'яті комп'ютера. На рисунку 1.7 показано, як виглядає рецепт.

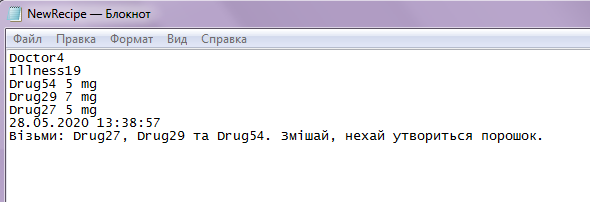


Рисунок 1.7 - Приклад рецепту

1.4 Функції старшої медсестри

1.4.1 Функція «Вхід у програму»

Старша медсестра входить в програму за допомогою імені та паролю. Функції старшої медсестри повинні бути приховані від основних користувачів програми - лікарів. Тому вхід медсестри відбувається після натискання відповідної кнопки у вікні входу.

Основний сценарій:

1. Медсестра натискає кнопку «Nurse» (див. рис. 1.1), і відкривається вікно входу медсестри (див. рис. 1.8).
2. Старша медсестра вводить ім'я і пароль і натискає кнопку «Ready».
3. Введені дані перевіряються, і відкривається головне вікно медсестри (див. рис. 1.9).
4. Якщо дані не пройшли перевірку, на формі введення з'являється повідомлення про це (див. рис. 1.10) і можна спробувати увійти знову. Число спроб не обмежена.

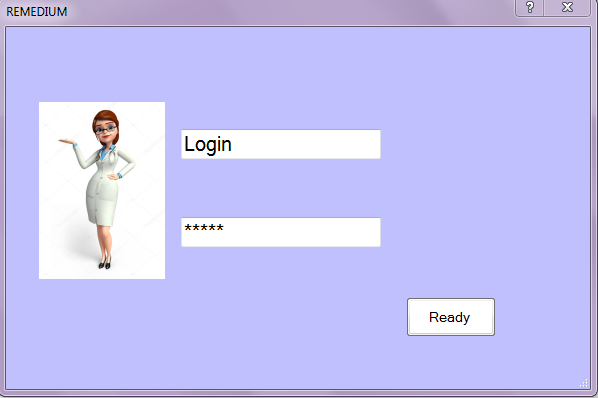


Рисунок 1.8 - Вікно для входу медсестри

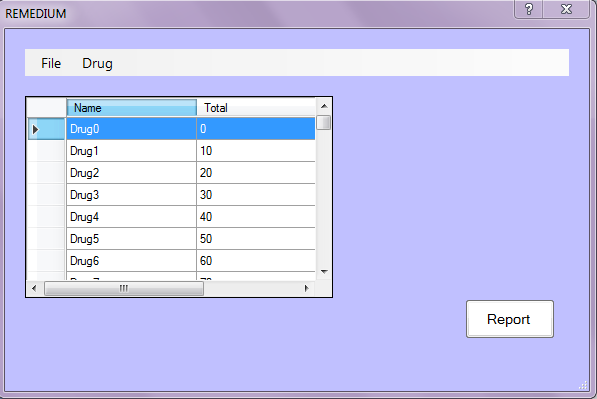


Рисунок 1.9 - Головне меню медсестри

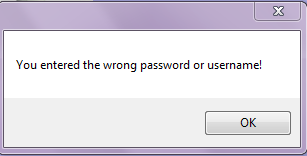


Рисунок 1.10 - Повідомлення при неправильному введені паролю або імені

1.4.2 Функція «Зміна медикаменту»

Список медикаментів включає в себе всі ліки, а не тільки ті, які зараз є на складі. Можна змінити не тільки кількість медикаменту, а й їх назву, взаємозамінність і одиницю виміру.

Сценарій «Зміна медикаменту»:

1. Медсестра вибирає медикамент із загального переліку.
2. Натискає зверху на «Drug», а потім «Change».
3. Відкривається форма зміни медикаменту (див. рис. 1.11). Поля форми заповнені.
4. Змінює поля форми.
5. Натискає кнопку «Ok».

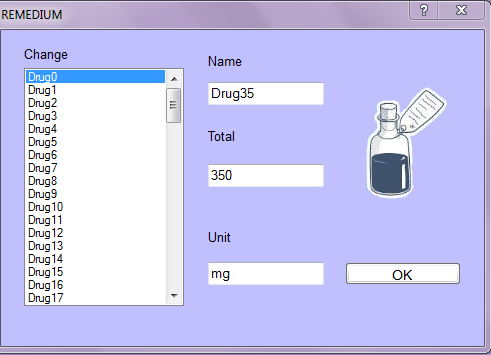


Рисунок 1.11 - Форма зміни медикаменту

Сценарій «Додавання медикаменту»:

1. Медсестра натискає кнопку «Drug», а потім «New».
2. Відкривається така ж форма, як і при зміні медикаменту, але поля порожні (див. рис. 1.11).
3. Заповнює поля форми.
4. Натискає кнопку «Ok».

Сценарій «Видалення медикаменту»:

1. Медсестра натискає кнопку «Drug», а потім «Delete».
2. З'являється повідомлення
3. При натисканні кнопки «Yes», медикамент видалиться.
4. При натисканні кнопки «No», медикамент залишається.

1.4.3 Функція «Звіт використаних медикаментів»

Після того, як лікар оформив рецепт, він з'являється у звіті. Медсестра не може міняти звіт, а тільки його дивиться.

Сценарій:

1. У головному меню медсестри, вона натискає кнопку «Report».
2. Відкривається форма (див. рис. 1.12).
3. Медсестра вибирає дату та натискає кнопку «Ok».
4. Поряд з'являється список медикаментів, які були використані в цей час.
5. Якщо медсестра натискає «Exit», то відкривається головне вікно медсестри.

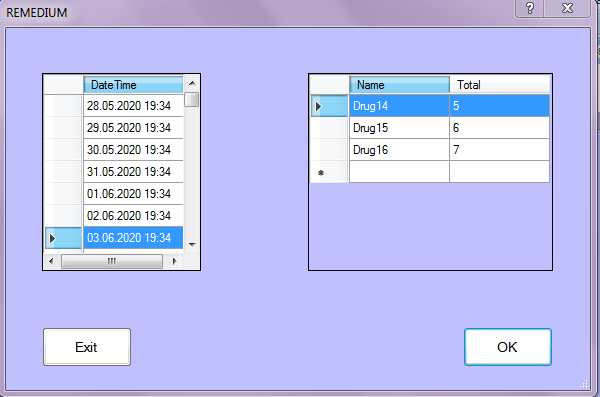


Рисунок 1.12 - Вікно звітів

2 ПРОЕКТНА СПЕЦИФІКАЦІЯ

2.1 Архітектура

Для створення програми буде використано мову C# та графічний інтерфейс програмування додатків Windows Forms на платформі .NET Framework 4. 7. 2, середовище розробки Microsoft Visual Studio 2019.

Вибір мови обумовлений об'єктно-орієнтованим підходом до написання програми. Мова C# ─ одна із популярніших, зручніших та повноцінних серед об'єктно-орієнтованих мов програмування.

У процесі проектування було розглянуто 2 платформи, з якими C# найкраще взаємодіє, а саме .NET Core і .NET Framework. В проекті було обрано останню, тому що вона має більш високий рівень розвитку та кількість керуючих елементів. Це означає, що наш додаток буде працювати лише на комп'ютерах з операційної системою Windows.

Для реалізації інтерфейсу користувача було вирішено використати графічний інтерфейс, бо він більш простий для сприйняття та поняття. Для реалізації інтерфейсу будемо використовувати Windows Forms, який є частиною платформи .NET Framework. З його допомогою можна створити повноцінний інтерфейс для користувачів додатка.

2.2Структура проекту

У проекті є дві категорії користувачів, яким необхідні різні функції програми ─ це лікар та медсестра. Тому наш проект поділений на два окремих додатка. Для тестування програми необхідно на етапі програмування створити декілька тестових даних для перевірки правильності роботи програми.

Із вище сказаного я можемо скласти таку структуру проекту:

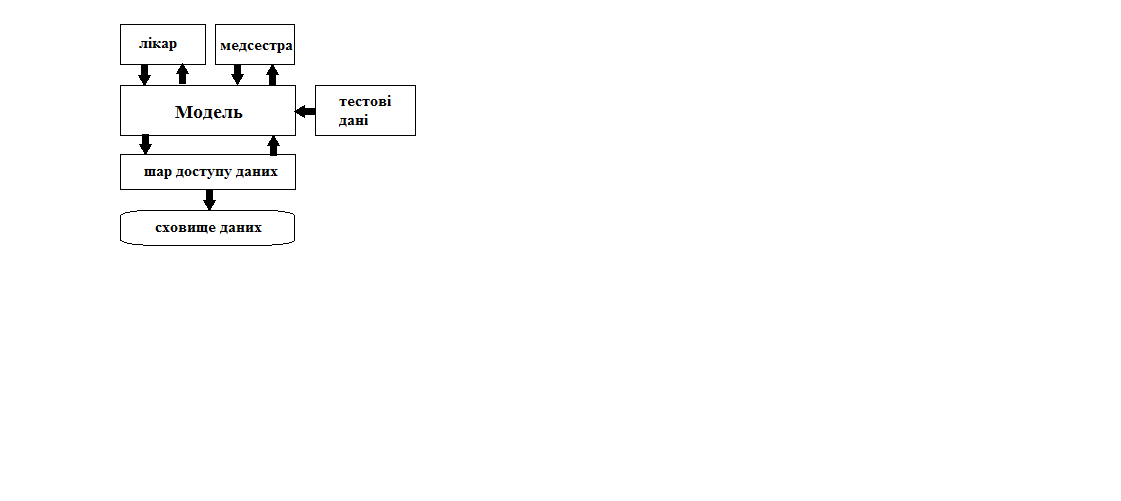


Рисунок 2.1 - Структура проекту

2.3 Об'єктна модель. Опис

У довіднику лікаря є хвороби і медикаменти.

Лікар вибирає симптоми, визначає хворобу, перевіряє наявність медикаментів і формує рецепт.

Старша медсестра додає нові медикаменти, змінює кількість всіх медикаментів і може подивитися звіт використаних ліків.

Про хворобу відомо: назва, симптоми, ліки та їх кількість.

Про медикамент (ліки) відомо: назва, його загальна кількість на складі, одиниця виміру та взаємозамінність (назва ліків, якими можна його замінити).

Про лікаря та медсестру відомо: ім'я та пароль.

До простих типів не відносяться:

Довідник, медикамент, медсестра, лікар, хвороби та звіт.

Колекцією не являється і колекцій не має:

Медсестра та лікар.

Медикамент - назва + колекція взаємозамінних ліків + загальна кількість + одиниця виміру.

Лікар - ім'я + пароль.

Старша медсестра - ім'я + пароль.

Для зручності:

Порція - медикамент + його кількість.

Хвороба - назва + колекція симптомів + колекція порцій.

Рецепт - доктор + хвороба + дата.

Довідник лікаря - це декілька колекцій: лікарів, хвороб, ліків та рецептів.

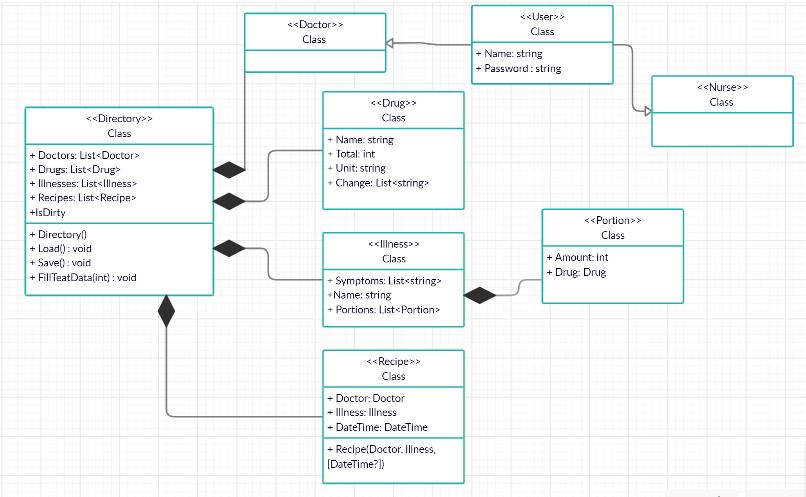


Рисунок 2.2 - Діаграма класів

2.4 Сховище та доступ до даних

Усі данні будуть представлені у вигляді окремих бінарних файлів в каталозі проекту. Данні будуть переводитись із текстових у потік байтів за допомогою серіалізації з використанням вбудованої бібліотеки BinaryFormatter. Захист даних у проекті не розглядається.

3 ІНСТРУКЦІЯ КОРИСТУВАЧА

Інтерфейс програми реалізований у графічному режимі. Інтерфейс простий у користуванні й не потребує значних комп'ютерних ресурсів для роботи у ньому.

Щоб запустити програму через операційну систему потрібно знайти файл DoctorApp.exe і два рази клацнути по ньому. В результаті програма буде запущена у вікні під куруванням операційної системи.

Недоліки цього методу є те, що неможливо вносити зміни до програмного коду.

Коли програма запуститься, з'явиться вікно входу для лікарів (див. рис. 1.1).

Посередині знаходиться список, де відображається перелік усіх лікарів. Щоб обрати потрібне ім'я, треба один раз клацнути на нього. Під переліком лікарів знаходиться поле для введення паролю. Достатньо один раз клікнути на поле або натиснути «Tab» для початку запису. Вводити можна тільки цифри. У верхньому правому куті є кнопка для медсестри. Туди треба тиснути, якщо треба змінити будь-який медикамент або подивитися звіт використаних медикаментів. У нижньому правому куті знаходиться кнопка для продовження роботи. Достатньо натиснути на неї один раз для відкриття наступної форми (див. рис. 1.2).

Щоб обрати симптом, треба на нього натиснути. Симптомів можна вибирати будь-яку кількість. Потім натиснути на клавіатурі «Tab» і «Enter» або клікнути один раз на кнопку в нижньому правому куті для відкриття наступної форми (див. рис. 1.4).

Повторюємо дії, які робили на минулій формі, але можна вибирати тільки одну хворобу. Відкривається вікно для формуванню рецепту (див. рис. 1.6).

У верхньому правому куті знаходиться список ліків. Щоб обрати декілька, треба зажати на клавіатурі «Ctrl» і мишкою клацнути на потрібний медикамент. Можна вибрати довільну кількість ліків. У нижньому лівому куті можна написати коментарій до рецепту. Якщо коментарій не потрібний, то можна не чіпати текст, який вже написаний, або видалити його. Для збереження рецепта треба натиснути на кнопку в нижньому правому куті.

Щоб подивитися рецепт потрібно знайти файл NewRecipe.txt і два рази клікнути на нього. Приклад рецепту показано на рис. 1.7.

Щоб змінити будь-який медикамент або подивитися звіт використаних товарів. На формі входу лікарів (див. рис. 1.1) потрібно натиснути кнопку у верхньому правому куті. Відкриється форма входу медсестри (див. рис. 1.8).

Натискаємо по черзі на поля та вводимо ім'я користувача та пароль. Потім натиснути на клавіатурі «Tab» і «Enter» або клікнути один раз на кнопку в нижньому правому куті для відкриття головного меню медсестри (див. рис. 1.9).

Для виходу треба натиснути «File», а потім - «Exit» або на клавіатурі «Alt», «Enter», «Tab», «Enter». Відкриється найперше вікно (див. рис. 1.1).

Для збереження змін натиснути «File», а потім - «Exit» або на клавіатурі «Alt», «Enter», «Enter».

Для видалення медикаменту виберіть його у переліку, а потім натиснути «Drug», а потім - «Delete». З'явиться повідомлення про це, якщо результат буде позитивний, то медикамент буде видалений.

Для створення нового медикаменту натисніть «Drug», «New». Відкриється форма для створення нового медикаменту(див. рис. 3.1).

У переліку зліва вибираємо ліки, якими можна замінити поточний. Щоб вибрати взаємозамінний препарат достатньо на нього один раз клікнути. У першому полі посередині пишемо назву медикаменту, у другому - його кількість на складу, у третьому - одиницю виміру. Після закінчення тиснемо на кнопку у нижньому правому куті.

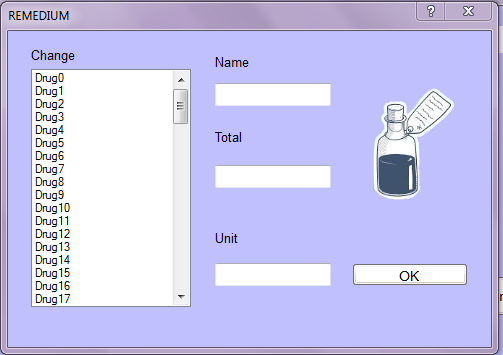


Рис. 3.1 - Форма для створення нового медикаменту

Для зміни медикаменту натискаємо «Drug», «Change». Відкривається форма, як на рисунку 3.1, але з заповненими полями, які треба змінювати. Принцип дії такий же, як і при створенні нового медикаменту.

Щоб подивитися звіт використаних медикаментів, треба натиснути у нижньому правому куті кнопку «Report». Відкриється вікно звітів (див. рис. 1.12).

Праворуч буде список дат, в які лікарі використовували медикаменти. Можна вибрати тільки одну дату, а потім натиснути кнопку у нижньому правому куті. Зверху в правому куті з'являється список використаних ліків у цей час. Для виходу натисніть кнопку у нижньому лівому куті. Знову відкриється головне меню медсестри (див. рис. 1.9).

ВИСНОВКИ

Отримані знання стануть істотною підмогою в розумінні методів створення програмних систем на основі об'єктно-орієнтованого програмування, також були набуті навички роботи з програмною системою, створення зв'язків між об'єктами.

У результаті виходить програма під назвою «Довідник лікаря». Розроблена програма полегшить роботу лікарів і медсестер, а пацієнти зможуть краще розуміти призначені рецепти. У цій програмі лікар зможе виписати рецепт залежно від обраних симптомів та наявності лікарських засобів на складі, а медична сестра - відкоригувати запаси лікарських засобів та подивитися звіти про оцінку. У програми є дві категорії користувачів, яким необхідні різні функції програми. Тому наш проект поділений на два додатки.

Таким чином робота виконана в повному обсязі, мета досягнута.

Програма написана мовою програмування C# на .NET Framework 4.7.2 з використанням технологій Windows Forms в середовищі розробки Microsoft Visual Studio 2019. Всі дані програми зберігаються локально. Програму можна вдосконалити за допомогою реальної бази даних, яка синхронізується з базою даних хвороб та медикаментів. Галузь застосування - одна місцева лікарня.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бондарев В. М. Объектно-ориентированное программирование на C#: учеб. Пособие / В. М. Бондарев. - Харьков: СМИТ, 2009. 224 с.

2. Височанський В. С. , Кардаш А. І., Костєв В. С. та ін. Елементи інформатики: довідник. К. :Наук. думка, 2003. 92 с.

3. Джон Пол Мюллер. C# для чайников/ Ігор Красіков: Діелектрика, 2018. 608 с.

4. Мэтт Вайсфельд. Книга Объектно-ориентированное мышление: Питер, 2018, 304 с.

5. Полное руководство по языку программирования C# 8 и платформе .NET Core [Електронний ресурс] - Режим доступу до ресурсу: <https://metanit.com/sharp/tutorial/>

ДОДАТОК А

Код бібліотеки

using DirectoryLibrary.Models;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.Serialization.Formatters.Binary;

using Directory = DirectoryLibrary.Models.Directory;

namespace DirectoryLibrary.DATA

{

public class Dao

{

private readonly Directory directory;

const string filePath = "directory.bin";

public Dao(Directory directory)

{

this.directory = directory;

}

public void Save()

{

using (Stream stream = File.Create(filePath))

{

var serializer = new BinaryFormatter();

serializer.Serialize(stream, directory);

}

}

public void Load()

{

using (Stream stream = File.OpenRead(filePath))

{

var serializer = new BinaryFormatter();

Directory st = (Directory)serializer.Deserialize(stream);

Copy(st.Doctors, directory.Doctors);

Copy(st.Drugs, directory.Drugs);

Copy(st.Illnesses, directory.Illnesses);

Copy(st.Recipes, directory.Recipes);

}

void Copy<T>(List<T> from, List<T> to)

{

to.Clear();

to.AddRange(from);

}

}

}

}

using DirectoryLibrary.DATA;

using System;

using System.Collections.Generic;

namespace DirectoryLibrary.Models

{

[Serializable]

// Класс спавочик врача

public class Directory

{

public List<Doctor> Doctors { private set; get; }

public List<Drug> Drugs { private set; get; }

public List<Illness> Illnesses { private set; get; }

public List<Recipe> Recipes { private set; get; }

public bool IsDirty;

public Directory()

{

Doctors = new List<Doctor>();

Drugs = new List<Drug>();

Illnesses = new List<Illness>();

Recipes = new List<Recipe>();

}

public void FillTestData(int n)

{

// Doctor

Doctors = new List<Doctor>();

for (int i = 0; i < n/2; i++)

{

Doctors.Add(new Doctor { Name = $"Doctor{i}", Password = "123" });

}

// Drug

Drugs = new List<Drug>();

const int m = 4;

for (int i = 0; i <= n - m; i++)

{

var ch = new List<string>();

for (int j = 0; j < i % m; j++)

{

if ((i \* j) % (n - m) != i)

ch.Add($"Drug{(i \* j) % (n - m)}");

}

Drugs.Add(new Drug()

{

Name = $"Drug{i}",

Total = i \* 10,

Unit = "mg",

Change = ch

});

}

// Illness

Illnesses = new List<Illness>();

for (int i = 0; i <= n - m; i++)

{

var sym = new List<string>();

var ps = new List<Portion>();

for (int j = 0; j < i % m + 1; j++)

{

ps.Add(new Portion { Drug = Drugs[(m + n + i + j) % (n - m)], Amount = (j + m)%35 + 1});

sym.Add($"Symptom{(j + i + m) % 16}");

}

Illnesses.Add(new Illness { Name = $"Illness{i}", Portions = ps, Symptoms = sym });

}

// Recipes

Recipes = new List<Recipe>();

for (int i = 0; i <= n - m; i++)

{

Recipes.Add(new Recipe(Doctors[i%50], Illnesses[i], DateTime.Now + TimeSpan.FromDays(i)));

}

}

public void Save()

{

new Dao(this).Save();

}

public void Load()

{

new Dao(this).Load();

}

}

}

using System;

namespace DirectoryLibrary.Models

{

[Serializable]

// Общая часть для пользователей

public abstract class User

{

public string Name { get; set; }

public string Password { get; set; }

}

}

using System;

namespace DirectoryLibrary.Models

{

[Serializable]

// Класс для докторов

public class Doctor : User

{

}

}

using System;

namespace DirectoryLibrary.Models

{

[Serializable]

// Класс для медсестры

public class Nurse : User

{

}

}

using System;

using System.Collections.Generic;

namespace DirectoryLibrary.Models

{

[Serializable]

// Класс для рекарств

public class Drug

{

public string Name { get; set; }

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

public int Total { set; get; }

public string Unit { set; get; }

}

}

using System;

namespace DirectoryLibrary.Models

{

[Serializable]

// Класс для рецепта

public class Recipe

{

public Recipe(Doctor doctor, Illness illness, DateTime? t = null)

{

Illness = illness;

Doctor = doctor;

if (t == null)

DateTime = DateTime.Now;

else

DateTime = (DateTime)t;

}

public Doctor Doctor { private set; get; }

public Illness Illness { private set; get; }

public DateTime DateTime { private set; get; }

}

}

using System;

using System.Collections.Generic;

namespace DirectoryLibrary.Models

{

[Serializable]

// Класс для болезни

public class Illness

{

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

public string Name { get; set; }

public List<Portion> Portions { get; set; }

}

}

using System;

namespace DirectoryLibrary.Models

{

[Serializable]

// Класс для порции

public class Portion

{

public Drug Drug { get; set; }

public int Amount { get; set; }

}

}

ДОДАТОК Б

Код тестів програми

using DirectoryLibrary.Models;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace DoctorDirectoryTest

{

[TestClass]

// Проверка правильности тестовых данных

public class DirectoryTest1

{

[TestMethod]

public void FillData1()

{

Directory directory = new Directory();

directory.FillTestData(100);

Assert.AreEqual(directory.Drugs[0].Name, "Drug0");

}

public void FillData2()

{

Directory directory = new Directory();

directory.FillTestData(100);

Assert.AreEqual(directory.Recipes[0].Illness.Name, "Illness0");

}

public void FillData3()

{

Directory directory = new Directory();

directory.FillTestData(100);

Assert.AreEqual(directory.Illnesses[2].Name, "Illness2");

}

public void FillData4()

{

Directory directory = new Directory();

directory.FillTestData(100);

Assert.AreEqual(directory.Illnesses[4].Symptoms[0], "Symptom8");

}

public void FillData5()

{

Directory directory = new Directory();

directory.FillTestData(100);

Assert.AreEqual(directory.Doctors[0].Name, "Doctor0");

}

}

}

ДОДАТОК В

Код DoctorApp

using System;

using System.IO;

using System.Windows.Forms;

namespace DoctorApp

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

if (File.Exists("directory.bin"))

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new FormStart());

}

}

}

}

using DirectoryLibrary.Models;

using System;

using System.IO;

using System.Windows.Forms;

using Directory = DirectoryLibrary.Models.Directory;

namespace DoctorApp

{

// Форма для входа

public partial class FormStart : Form

{

Directory directory;

Doctor doctorNow;

public FormStart()

{

InitializeComponent();

directory = new Directory();

directory.FillTestData(100);

ListDoctor.DataSource = directory.Doctors;

ListDoctor.DisplayMember = "Name";

}

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

{

doctorNow = ListDoctor.SelectedItem as Doctor;

var pass = Password.Text;

if (pass == doctorNow.Password)

{

var f2 = new FormSympt(directory, doctorNow);

Hide();

f2.ShowDialog();

this.Visible = true;

}

else

MessageBox.Show("You entered the wrong password!");

}

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

{

var f2 = new NurseApp.LoginForm(directory);

Hide();

f2.ShowDialog();

this.Visible = true;

}

}

}

using DirectoryLibrary.Models;

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace DoctorApp

{

// Форма, где врач выбирает симптомы

public partial class FormSympt : Form

{

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

public Directory direct;

public Doctor doctor;

public FormSympt(Directory direct, Doctor doctor)

{

InitializeComponent();

this.direct = direct;

this.doctor = doctor;

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

for (int i = 0; i < 16; i++)

sym.Add($"Symptom{i}");

listSympt.DataSource = sym;

}

private void Changed(object sender, EventArgs e)

{

symptomNow.Add((string)listSympt.SelectedValue);

}

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

{

listSympt.SelectedIndexChanged += Changed;

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

{

symptomNow.Add($"{listSympt.SelectedItems[i]}");

}

if (listSympt.SelectedItems.Count == 0)

MessageBox.Show("Choose symptoms!");

else

{

var f3 = new FormIll(direct, doctor, symptomNow);

Hide();

f3.ShowDialog();

this.Visible = true;

this.Close();

}

}

}

}

using DirectoryLibrary.Models;

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace DoctorApp

{

// Форма, гда врач выбирает болезнь

public partial class FormIll : Form

{

List<string> sympt;

Doctor doctor;

Directory directory;

Illness illNow;

public FormIll(Directory directory, Doctor doctor, List<string> sympt)

{

InitializeComponent();

this.doctor = doctor;

this.directory = directory;

this.sympt = sympt;

List<Illness> ill = new List<Illness>();

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

{

bool ToAdd = false;

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

{

for (int k = 0; k < directory.Illnesses[i].Symptoms.Count; k++)

{

if (directory.Illnesses[i].Symptoms[k] == sympt[j])

ToAdd = true;

}

}

if (ToAdd)

ill.Add(directory.Illnesses[i]);

}

listIll.DataSource = ill;

listIll.DisplayMember = "Name";

}

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

{

illNow = listIll.SelectedItem as Illness;

var f4 = new ReciepForm(directory, doctor, illNow);

Hide();

f4.ShowDialog();

this.Visible = true;

this.Close();

}

}

}

using DirectoryLibrary.Models;

using System;

using System.Collections.Generic;

using System.IO;

using System.Windows.Forms;

using Directory = DirectoryLibrary.Models.Directory;

namespace DoctorApp

{

// Форма, где врач выписывает рецепт

public partial class ReciepForm : Form

{

Directory directory;

Doctor doctor;

Illness ill;

List<Drug> drug = new List<Drug>();

public ReciepForm(Directory directory, Doctor doctor, Illness ill)

{

InitializeComponent();

this.directory = directory;

this.doctor = doctor;

this.ill = ill;

string symp = ill.Symptoms[0];

for (int i = 1; i < ill.Symptoms.Count; i++)

symp += $", {ill.Symptoms[i]}";

SympBox.Text = symp;

DoctorBox.Text = doctor.Name;

illBox.Text = ill.Name;

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

{

var portion = ill.Portions[i];

if (ill.Portions[i].Drug.Total - ill.Portions[i].Amount > 0)

drug.Add(new Drug { Name = portion.Drug.Name, Total = portion.Amount, Unit = portion.Drug.Unit, Change = null });

for (int j = 0; j < portion.Drug.Change.Count; j++)

{

Drug dr = GetDrugByName(portion.Drug.Change[j]);

if (dr.Total - portion.Amount > 0)

drug.Add(new Drug { Name = dr.Name, Total = portion.Amount, Unit = dr.Unit, Change = null });

}

}

if (drug == null)

DrugGriedView.DataSource = "Лекарств нет";

else

DrugGriedView.DataSource = drug;

}

private Drug GetDrugByName(string drug)

{

if (drug.Substring(0, 4) == "Drug")

{

int ind = Convert.ToInt32(drug.Substring(4));

return directory.Drugs[ind];

}

return directory.Drugs[0];

}

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

{

List<Portion> portions = new List<Portion>();

using (var wr = new StreamWriter("NewRecipe.txt"))

{

wr.WriteLine(doctor.Name);

wr.WriteLine(ill.Name);

foreach (DataGridViewRow row in DrugGriedView.SelectedRows)

{

int ind = row.Index;

wr.WriteLine($"{drug[ind].Name} {drug[ind].Total} {drug[ind].Unit}");

portions.Add(new Portion { Drug = GetDrugByName(drug[ind].Name), Amount = drug[ind].Total });

}

wr.WriteLine(DateTime.Now);

if (ComentBox.Text != "Напишите комментарий" && ComentBox.Text != null)

wr.WriteLine(ComentBox.Text);

ill.Portions = portions;

directory.Recipes.Add(new Recipe(doctor, ill));

}

Close();

}

}

}

ДОДАТОК Г

Код NurseApp

using DirectoryLibrary.Models;

using System;

using System.Windows.Forms;

namespace NurseApp

{

// Форма для входа медсестры

public partial class LoginForm : Form

{

Directory directory;

public LoginForm(Directory directory)

{

InitializeComponent();

this.directory = directory;

}

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

{

if (LoginBox.Text == "Nurse" && PasswordBox.Text == "12345")

{

var f3 = new MainForm(directory);

Hide();

f3.ShowDialog();

this.Visible = true;

this.Close();

}

else

MessageBox.Show("You entered the wrong password or username!");

}

}

}

using DirectoryLibrary.Models;

using System;

using System.Windows.Forms;

namespace NurseApp

{

// Форма для добавления и удаления медикамента

public partial class MainForm : Form

{

Directory directory;

public MainForm(Directory directory)

{

InitializeComponent();

this.directory = directory;

directory.IsDirty = false;

drugBindingSource.DataSource = directory.Drugs;

}

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

{

directory.Save();

directory.IsDirty = false;

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

if (!directory.IsDirty)

return;

var res = MessageBox.Show("Save change before exiting?", "", MessageBoxButtons.YesNoCancel);

switch (res)

{

case DialogResult.Cancel:

e.Cancel = true;

break;

case DialogResult.Yes:

directory.Save();

break;

case DialogResult.No:

break;

}

}

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

{

Close();

}

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

{

Drug del = drugGridView.SelectedRows[0].DataBoundItem as Drug;

if (del != null)

{

var res = MessageBox.Show($"Delete {del.Name}?", "", MessageBoxButtons.YesNo);

if (res == DialogResult.Yes)

{

directory.Drugs.Remove(del);

drugBindingSource.ResetBindings(false);

directory.IsDirty = true;

}

}

}

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

{

var f3 = new ReportForm(directory);

Hide();

f3.ShowDialog();

this.Visible = true;

}

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

{

var pf = new DrugForm(directory);

if (pf.ShowDialog() == DialogResult.OK) { }

drugBindingSource.ResetBindings(false);

directory.IsDirty = true;

var lastIdx = drugGridView.Rows.Count - 1;

drugGridView.Rows[lastIdx].Selected = true;

drugGridView.FirstDisplayedScrollingRowIndex = lastIdx;

}

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

{

var toEdit = drugGridView.SelectedRows[0].DataBoundItem as Drug;

if (toEdit != null)

{

var pf = new DrugForm(toEdit, directory);

if (pf.ShowDialog() == DialogResult.OK) { }

drugBindingSource.ResetBindings(false);

directory.IsDirty = true;

}

}

}

}

using DirectoryLibrary.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Windows.Forms;

namespace NurseApp

{

// Форма для изминения/добавление нового медикамента

public partial class DrugForm : Form

{

public Drug drug;

Directory directory;

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

public DrugForm(Directory directory)

{

InitializeComponent();

this.directory = directory;

listChange.DataSource = directory.Drugs;

listChange.ValueMember = "Name";

}

public DrugForm(Drug drug, Directory directory) : this(directory)

{

this.drug = drug;

nameBox.Text = drug.Name;

unitBox.Text = drug.Unit;

totalBox.Text = Convert.ToString(drug.Total);

}

private void Changed(object sender, EventArgs e)

{

ch.Add((string)listChange.SelectedValue);

}

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

{

if (nameBox.Text != "" && unitBox.Text != "" && totalBox.Text != "")

{

listChange.SelectedIndexChanged += Changed;

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

{

ch.Add($"{listChange.SelectedItems[i] as Drug}");

}

drug = new Drug { Name = nameBox.Text, Total = Convert.ToInt32(totalBox.Text), Unit = unitBox.Text, Change = ch };

directory.Drugs.Add(drug);

drug.Name = nameBox.Text;

drug.Unit = unitBox.Text;

drug.Total = Convert.ToInt32(totalBox.Text);

drug.Change = ch;

Close();

}

else

MessageBox.Show("All fields must be filled!");

}

private void totalBox\_KetPress(object sender, KeyPressEventArgs e)

{

if ((e.KeyChar <= 47 || e.KeyChar >= 58) && e.KeyChar != Convert.ToChar(8))

e.Handled = true;

}

private void nameBox\_Validation(object sender, CancelEventArgs e)

{

if (String.IsNullOrEmpty(nameBox.Text))

{

errorProvider.SetError(nameBox,"Empty field!");

}

else

{

errorProvider.Clear();

}

}

private void totalBox\_Validation(object sender, CancelEventArgs e)

{

if (String.IsNullOrEmpty(totalBox.Text))

{

errorProvider.SetError(totalBox, "Введите количество!");

}

else

{

errorProvider.Clear();

}

}

private void unitBox\_Validation(object sender, CancelEventArgs e)

{

if (String.IsNullOrEmpty(unitBox.Text))

{

errorProvider.SetError(unitBox, "Введите количество!");

}

else

{

errorProvider.Clear();

}

}

}

}

using DirectoryLibrary.Models;

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace NurseApp

{

// Форма для просмотра отчета

public partial class ReportForm : Form

{

Directory directory;

public ReportForm(Directory directory)

{

InitializeComponent();

this.directory = directory;

dataGridView.DataSource = directory.Recipes;

}

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

{

Close();

}

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

{

List<Drug> dr = new List<Drug>();

Recipe toShow = dataGridView.SelectedRows[0].DataBoundItem as Recipe;

for (int i = 0; i < toShow.Illness.Portions.Count; i++)

{

Drug drug = new Drug { Name = toShow.Illness.Portions[i].Drug.Name, Total = toShow.Illness.Portions[i].Amount };

dr.Add(drug);

}

drugBindingSource.DataSource = dr;

drugBindingSource.ResetBindings(false);

}

}

}

<https://github.com/NChervinska/directory>