МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет «Запорізька політехніка»**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Кафедра програмних засобів\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(найменування кафедри)

**КУРСОВИЙ ПРОЕКТ**

з дисципліни «Об’єктно-орієнтоване програмування»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(назва дисципліни)

на тему:\_\_\_\_ «Розробка чат-бота для запису до лікаря »\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студентів 1 курсу групи \_\_\_\_\_\_КНТ-119сп\_\_ \_\_\_\_\_\_

спеціальності Інженерія програмного забеспечення

напряму підготовки: Інженерія програмного забезпе -

чення\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Дев'ятко Б. М.\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Зінов'єв А. А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник Доцент к.н.т., Табунщик Г. В.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала

Кількість балів: \_\_\_\_\_\_\_ Оцінка: ECTS

Члени комісії Каплієнко Т. І.

(підпис) (прізвище та ініціали)

Миронова Н. О.

(підпис) (прізвище та ініціали)

Табунщик Г. В.

(підпис) (прізвище та ініціали)

м. Запоріжжя

2019 рік

**ЗАВДАННЯ  
на курсовий проект студентів**

Дев'ятко Б. М., Зінов'єв А. А.

1. Тема проекту: Розробка чат-бота для запису до лікаря

2. Термін здачі студентами закінченого проекту: 24 квітня 2020

3. Вихідні дані до проекту: створити квиток про запис до лікаря

дані: інформація про організацію роботи в лікарні, лікарів, їх розклад, кабінетів, адреси лікарень;

вихідні дані: створений квиток про запис до лікаря.

4. Зміст розрахунково-пояснювальної записки:

1. Аналіз предметної області;

2. Аналіз програмних засобів;

3. Основні рішення з реалізації компонентів системи;

4. Керівництво програміста;

5. Керівництво корисутвача;

Висновки

Додаток А. Текст програми

Додаток Б. Інтерфейс програми

Додаток В. Слайди презентації

5. Дата видачі завдання: 18 грудня 2019 року

Календарний план

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва етапів курсового  проекту (роботи) | Термін виконання етапів  проекту (роботи) | Примітка |
| 1. | Аналіз індивідуального завдання. | 1-2 тиждень |  |
| 2. | Аналіз програмних засобів, що будуть використовуватись в роботі. | 2-3 тиждень |  |
| 3. | Аналіз структур даних, що необхідно використати в курсової роботі. | 3-5 тиждень |  |
| 4. | Вивчення можливостей програмної реалізації структур даних та інтерфейсу користувача. | 5-8 тиждень |  |
| 5. | Аналіз вимог до апаратних засобів | 8 тиждень |  |
| 6. | Розробка програмного забезпечення | 8-17 тиждень |  |
| 7. | Оформлення, відповідних пунктів пояснювальної записки. | 15-17 тиждень | Розділи 1-5 ПЗ |
| 8. | Захист курсової роботи. | 17 тиждень |  |

Студент Дев'ятко Б. М.

Студент Зінов'єв А. А.

Керівник Табунщик Г. В.

«квітень» 21 2020 р.

реферат

Мета роботи – створення програмного забезпечення чат-бота для запису до лікаря.

Проведено аналіз предметної області, досліджено аналогічне програмне забезпечення, а також існуючі методи та програмні засоби для вирішення завдання.

Для реалізації програмного продукту використовувалася мова програмування С# та середа розробки Visual Studio 2019.

Здійснено опис прийнятих рішень, розроблених алгоритмів, реалізованих класів, наведено опис полів та методів реалізованих класів. Приведені основні рішення щодо реалізації компонентів системи та методика роботи користувача з системою. Приведена структурна схема програми та характеристика програмних модулів

У даній роботі було проведено дослідження організіції роботи лікарень, розглянуто особливості мови програмування С# у середовищі розробки Visual Studio 2019. Потреби у збереженні, виведенні, конвертації та роботі з даними реалізовані у роботі із реляційною базою даних MS Access.

C#, ACCESS, CLASS, VISUAL STUDIO, ЗАПИС, ЛІКАР, ЛІКАРНЯ

зміст

[Календарний план 2](#_Toc27629364)

[реферат 3](#_Toc27629365)

[зміст 4](#_Toc27629366)

[перелік умовних позначень, символів, одиниць, скорочень і термінів 6](#_Toc27629367)

[вступ 7](#_Toc27629368)

[1 Аналіз предметної області 8](#_Toc27629369)

[1.1 Аналіз начального процесу, як основи предметної області 8](#_Toc27629370)

[1.2 Огляд існуючих методів вирішення завдання 9](#_Toc27629371)

[1.2 Огляд існуючих програм та сервісів для пошуку найкоротшого шляху 9](#_Toc27629372)

[1.2.1 Передмова 9](#_Toc27629373)

[1.2.2 Система «Битрикс24» 10](#_Toc27629374)

[1.2.3 Система «МЕДУЧЕТ SQL» 10](#_Toc27629375)

[2 Аналіз програмних засобів 12](#_Toc27629377)

[2.1 Огляд особливостей мови програмування 12](#_Toc27629378)

[2.2 Огляд особливостей обраного компілятору 12](#_Toc27629379)

[2.3 Огляд класової ієрархії WPF 13](#_Toc27629380)

[2.4 Висновки з розділу 13](#_Toc27629381)

[3 основні рішення з реалізації компонентів системи 14](#_Toc27629382)

[3.1 Основні рішення щодо розроблених класів 14](#_Toc27629383)

[3.1.1 Клас Human 15](#_Toc27629384)

[3.1.2 Клас Doctor 15](#_Toc27629385)

[3.1.3 Клас Client 16](#_Toc27629386)

[3.2 Основні розроблені алгоритми 16](#_Toc27629401)

[3.2.1 Аналіз головного алгоритму 16](#_Toc27629402)

[3.3 Основні рішення щодо розробки інтерфейсу 19](#_Toc27629408)

[3.3.1 Вибір платформи. 19](#_Toc27629409)

[3.3.2 Основні рішення з впорядкування елементів інтерфейсу 19](#_Toc27629410)

[3.3.3 Основні рішення з комунікації з користувачем 19](#_Toc27629411)

[3.3.4 Основні рішення з створенням інтерфейсу 20](#_Toc27629412)

[3.4 Основні рішення щодо розробки бази даних 21](#_Toc27629413)

[3.4.1 Основні рішення щодо вибору бази даних 21](#_Toc27629414)

[3.4.2 Структура збереження даних у БД 22](#_Toc27629415)

[3.4.2 Основні запити SQL, що було використано 25](#_Toc27629416)

[4 керівництво програміста 26](#_Toc27629417)

[4.1 Призначення та умови застосування програми 26](#_Toc27629418)

[4.1.1 Призначення програми 26](#_Toc27629419)

[4.1.2 Функції програми 26](#_Toc27629420)

[4.1.3 Умови застосування програми 26](#_Toc27629421)

[4.2 Характеристика програми 26](#_Toc27629422)

[4.2.1 Структура програми 27](#_Toc27629423)

[4.2.2 Алгоритм роботи програми 27](#_Toc27629424)

[4.3 Звертання до програми 30](#_Toc27629426)

[4.4 Вхідні та вихідні дані 30](#_Toc27629427)

[4.4.1 Вхідні дані 30](#_Toc27629428)

[4.4.2 Вихідні дані 31](#_Toc27629429)

[5 керівництво користувача 32](#_Toc27629431)

[5.1 Призначення програми 32](#_Toc27629432)

[5.2 Умови виконання програми 32](#_Toc27629433)

[5.2.1 Апаратні вимоги програми 32](#_Toc27629434)

[5.2.2 Вимоги до користувача 32](#_Toc27629435)

[5.3 Виконання програми 32](#_Toc27629436)

[5.3.1 Запуск програми 32](#_Toc27629437)

[5.3.2 Виконання роботи з програмою 33](#_Toc27629438)

[Висновки 36](#_Toc27629441)

[ПЕРЕЛІК ПОСИЛАНЬ 37](#_Toc27629442)

[Додаток А Діаграма класів 38](#_Toc27629443)

[Додаток Б Код Програми 39](#_Toc27629444)

[Додаток В інтерфейс програми 53](#_Toc27629445)

перелік умовних позначень, символів, одиниць, скорочень і термінів

БД – база даних.

Запит – запит в базу даних.

ООП - об’єктно-орієнтоване програмування.

СУБД – система управління базами даних.

SQL – Structured Query Language.

VS – Visual Studio

вступ

У наш час для прискорення наукового і технічного прогресу в усіх сферах життєдіяльності людини водночас з іншими заходами передбачається широке впровадження інформаційних технологій як фундаменту подальшого розвитку людства.

Крівництво будь-якого закладу сьогодні буде непроти зменшення впливу людського фактору на процесс роботи а також збільшення прибутку. Це досягається за рахунок підвищення продуктивності праці. Саме тому спостерігається інтенсивне впровадження у виробництво досягнень науково-технічного прогресу, застосування новітніх революційних технологій. Впровадження автоматичної реєстрації є одним зі способів скорочення штату працівників. А це означає не тільки зменшення кількості помилок а ще й менші затрати на виплату заробітної плати та ін.

Подібний застосунок може використовуватися у будь-якій сфері людської діяльности, у нашому випадку сферою використання являється охорона здоров’я.

Отже, актуальність даної курсової роботи, яка полягає у створенні можливлості автоматизації запису до лікаря, складно переоцінити. Тепер кожен житель м. Запоріжжя зможе отримати актуальну та вичерпну інформацію стосовно лікарень у своему районі, крім того він зможе записатися на прийом до потрібного йому лікаря у зручний для себе час дистанційно та в автоматичному режимі. А для керівництва закладів впровадження цього програмного продукту означає зменшення впливу людського фактору на організацію процесу праці, а саме зменшення вирогідності допущення помилок при запису пацієнтів та зменшення затрат на утримання персоналу, який відповідає за цей процес.

1 Аналіз предметної області

В даному розділі приведені приклади існуючих за стосунків з вирішення завдань автоматизації запису та програмне забезпечення, що реалізує дану потребу, а саме запис до лікаря.

1.1 Аналіз процесу надання медичних послуг, як основи предметної області

На сьогодні охорона здоров’я – одна з провідних галузей сфери виробництва послуг. Ст. 49 Конституції України закріплює право кожного на охорону здоров’я та медичну допомогу, зміст яких, зокрема, становить право на медичні послуги. Водночас визначення сутності медичних послуг має першочергове значення, адже від того наскільки вони є ефективними, залежить загальний рівень задоволеності якістю життя в нашому суспільстві.

Медична практика включає:

- консультаційні послуги та лікування, що надається лікарями всіх спеціальностей в лікувальних установах (амбулаторіях) і в умовах приватної практики;

- діагностичні послуги,

- проведення аналізів;

- послуги приватних консультантів;

- послуги швидкої та невідкладної медичної допомоги;

- послуги санаторно-курортних організацій;

- послуги інших лікувальних установ для відновлення здоров’я людини.

Там же зазначається, що медичні послуги можуть бути короткострокові та довгострокові.

1.2 Огляд існуючих методів вирішення завдання

Людям подобається записуватися онлайн. 25% записів відбувається в неробочий час або коли незручно дзвонити з робочого місця. Гірше, якщо телефон зайнятий або ніхто не взяв трубку, тому доводиться шукати інше місце в Інтернеті.

Фактично, в даний час, існують приклади вирішення таких завдань. Але вони мають певні недоліки, серед яких складний до опанування інтерфейс. Наш додаток вирішує цю проблему, адже запис проходить у максимально зручний для користувача спосіб – у “розмові” з чат-ботом.

**1.2 Огляд існуючих програм та сервісів для пошуку найкоротшого шляху**

1.2.1 Передмова

Перед розгядом аналогів, слід зазначити, що переважна більшість існуючих рішень розповсюджуються за платною ліцензією, що ускладнює процес порівняння.

Серед програмних засобів, що здійснюють автоматичний запис на прийом розглянемо наступні:

* «Битрикс24»;
* «МЕДУЧЕТ SQL»;

В ході написання роботи був проведений аналіз основних програм для складання розкладу, який проводився за настуними критеріями:

* виконання завдань запису на прийом;
* вартість ліцензії;
* платформа використання;
* зручність і простота інтерфейсу.

1.2.2 Система «Битрикс24»

Компанія-видавець: PinALL.

Посилання на продукт:

https://pinall.ru/catalog/uslugi/prilozheniye-dly-zapisi-na-priyem-online/

Короткий опис: додаток «Запись на прием к врачу» - це онлайн-регістратура для медичної организації.

Переваги:

* + запис з самого додатка;
  + установка розкладу лікаря в самому додатку;
  + можливість скасувати з програми запис.

Недоліки

* + складинй інтерфейс;
  + купівля та встановлення багатьох додатків для повноціної роботи.

**1.2.3** Система «МЕДУЧЕТ SQL»;

Компанія-видавець: МЕДУЧЕТ.

Посилання:

http://www.meduchet.com/page/additionalpages/Online\_appointment\_additional\_settings.php.

Короткий опис: Модуль «Запис на прийом» дозволяє легко знаходити вільні інтервали і проводити одночасний запис відразу в кілька кабінетів. Записаних пацієнтів бачать всі лікарі, і можуть самі записувати своїх пацієнтів. Записи на прийом можна розщеплювати, об'єднувати, переносити, дублювати і т.п. Графіки прийому на кожен день можна роздрукувати в декількох варіантах.

Переваги:

* + За фактом оплати пацієнтові можна роздрукувати касовий чек або нічого не друкувати. Якщо пацієнт не сплатив замовлення повністю, то в окремому модулі можна переглянути список боржників і надійшли оплат за будь-який період часу, з розбивкою за категоріями пацієнтів і підрозділам клініки;
  + Для кожної послуги можна вказати перелік витратних матеріалів, необхідних для її надання та, таким чином, розрахувати собівартість послуг.

Недоліки:

* + складинй до опанування інтерфейс;
  + встановлення багатьох додатків для повноціної роботи.

Робота програми продемонстрована на рисунку 1.1.

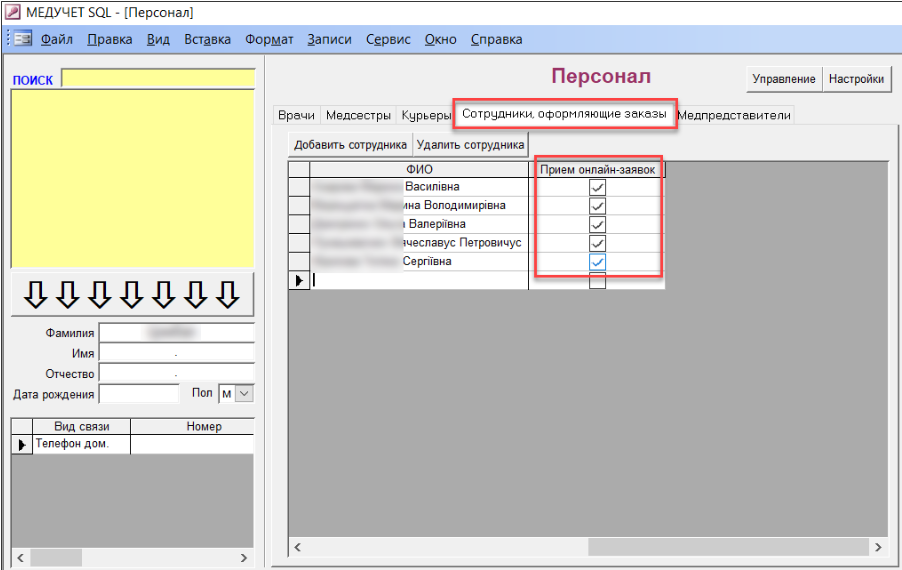


Рисунок 1.1 – Робота програми «МЕДУЧЕТ SQL»

2 Аналіз програмних засобів

В даному розділі розглянуті основні особливості програмних засобів, за допомогою який реалізовано курсовий проект.

2.1 Огляд особливостей мови програмування

C# - об'єктно-орієнтована мова програмування прикладного рівня.

Вона розроблена в 1998-2001 роках групою інженерів компанії Microsoft під керівництвом Андерса Хейлсберг і Скотта Вільтаумота як мову розробки додатків для платформи Microsoft .NET Framework.

C # відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C ++ і Java.

Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML.

2.2 Огляд особливостей обраного компілятору

Microsoft Visual Studio - лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також web-сайти, web-додатки, web-служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight.

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Решта вбудовуються інструменти включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, web-редактор, дизайнер класів і дизайнер схеми бази даних. Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, наприклад, Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server).

2.3 Огляд класової ієрархії

При розробці використовувалися наступні класи: абстрактний клас Human та дочірні йому класи Doctor та Client.

2.4 Висновки з розділу

У даному розділі було описано мову програмування C# та її особливості. Також було описано середовище розробки VS та засоби розробки, які вона надає. Також були описані класи, створенні під час розробки програмного засобу.

3 основні рішення з реалізації компонентів системи

В даному розділі розглянуті основні рішення з розробки класів, основні розроблені алгоритми, рішення щодо розробки інтерфейсу користувача, рішення щодо збереження даних та використання бази даних.

3.1 Основні рішення щодо розроблених класів

Структура програми складається з 17 класів, 3 з яких – користувацькі, 3 – класи форм, також присутні класи роботи з базою даних, таблицями. Один з користувацьких класів є абстрактним. Діаграми користувацьких класів знаходяться у додатку А.

Класову структуру програми зображено на рисунку 3.1.

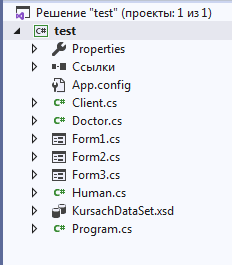


Рисунок 3.1 – Класова структура програми

3.1.1 Клас Human

Клас Human – це абстрактний клас, який є батьківським для класів Doctor та Client. У ньому визначений параметр повного ім’я людини. Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.1.

Таблиця 3.1 – Опис полів та методів класу Human

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| fullName | Змінна, в якій міститься ім’я людини. |

3.1.2 Клас Doctor

Doctor – дочірній від класу Human клас, об’эктами якого являються лікарі, що представлені в додатку. Він містить інформацію про спеціальність лікаря та його робочий кабінет. Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.2.

Таблиця 3.2 – Опис полів та методів класу Doctor

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| specialty | Змінна, в якій міститься спеціальність лікаря. |
| cabinet | Змінна, в якій міститься номер робочого кабінету лікаря. |

3.1.3 Клас Client

Client – дочірній від класу Human клас, об’эктами якого являються клієнти, тобто, персони, які проходять реєстрацію. Він містить інформацію про телефон клієнта. Код файлу визначення класу знаходиться в додатку Б. Дані та методи класу наведені у таблиці 3.3.

Таблиця 3.3 – Опис полів та методів класу Client

|  |  |
| --- | --- |
| Поля та методи класу | Опис |
| **1** | **2** |
| ***public:*** | |
| telephone | Змінна, в якій міститься номер телефону клієнта. |

3.2 Основні розроблені алгоритми

3.2.1 Аналіз головного алгоритму

В ході написання курсової роботи був створений власний алгоритм запису на прийом до лікаря.

Розроблений алгоритм піддається під подальшу оптимізацію та можливе розширення виконуваних завдань.

Вхідними даними для алгоритму є:

* стать користувача;
* район міста Запоріжжя;
* лікарня;
* потрібний лікар;
* дата та час запису на прийом;
* прізвище та ім’я;
* номер телефону користувача.

Під час підрахунку остаточного результату алгоритм працює з проміжними даними:

* графік лікаря.

У кінці виконання алгоритм формує вихідні данні:

* прізвище, ім’я та по батькові лікаря;
* кабінет лікаря;
* дата та час запису.

Усі значення мають бути коректними та строго формалізованими (це виконується на етапі створення бази даних).

Увесь алгоритм можна поділити на 2 етапи:

1. Виконання запису до лікаря;
2. Друк талона.

Виконання кожного етапу базується та таких алгоритмах:

* алгоритм лінійного пошуку;
* алгоритм сортування вставками.

Нижче представлено алгоритм, розроблений під час виконання курсової роботи (рис.3.2)

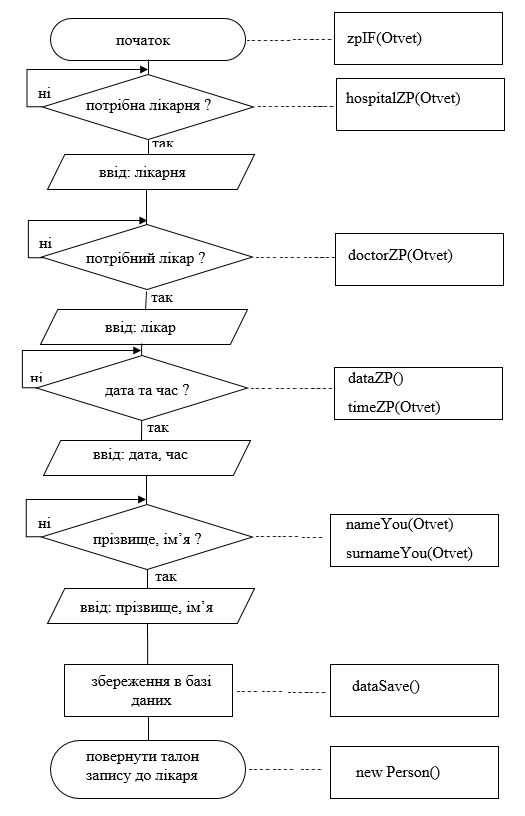


Рисунок 3.2 – Алгоритм запису на прийом до лікаря

Після виконання алгоритму маємо талон для користувача на прийом до лікаря в обраній лікарні.

3.3 Основні рішення щодо розробки інтерфейсу

3.3.1 Вибір платформи.

При розробці було прийнято рішення використовувати у якості платформи Windows Forms.

Windows Forms — інтерфейс програмування додатків (API), відповідальний за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за допомогою створення обгортки для Win32 API в керованому коді.

3.3.2 Основні рішення з впорядкування елементів інтерфейсу

Під час розробки інтерфейсу користувача було використано наступні елементі графічного інтерфейсу:

* label;
* textBox;
* dataTimePicker;
* panel;
* button;
* checkBox.

3.3.3 Основні рішення з комунікації з користувачем

Комунікація з користувачем відбувається методом вписуванням користувачем відповідей на запитання бота. Крім того, задля запобігання помилок у програмі є діалогові вікна, що мають на меті направляти користувача та попередити про можливості вводу некоректних даних.

3.3.4 Основні рішення з створенням інтерфейсу

Оскільки програмний продукт розрахований на широку публіку, інтерфейс є адаптивним по критерію статі. Приклади зовнішнього вигляду, який змінюється в залежності від обраної статі наведено на рисунках 3.3 та 3.4.

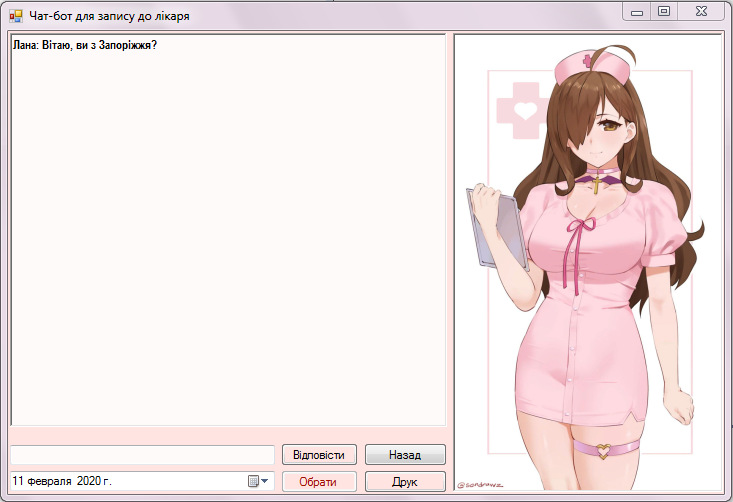


Рисунок 3.3 Вигляд головного вікна при обранні чоловічої статі

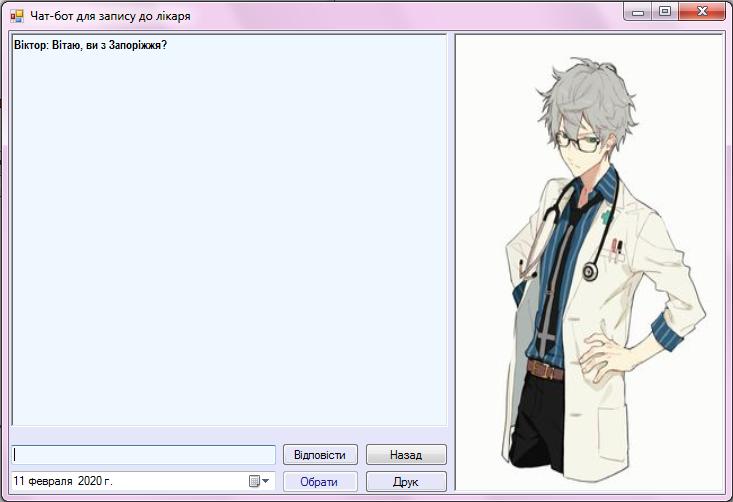


Рисунок 3.4 Вигляд головного вікна при виборі жіночої статі

3.4 Основні рішення щодо розробки бази даних

3.4.1 Основні рішення щодо вибору бази даних

Для розробки бази даних було використано Microsoft Access - реляційна СУБД корпорації Microsoft. Має широкий спектр функцій, включаючи зв'язані запити, сортування по різних полях, зв'язок із зовнішніми таблицями і базами даних. Завдяки вбудованій мові VBA, в самому Access можна писати програми, що працюють з базами даних.

Основні компоненти MS Access: майстер таблиць; будівник екранних форм; будівник SQL-запитів (мова SQL в MS Access не відповідає стандарту ANSI); будівник звітів, що виводяться на друк. Вони можуть викликати скрипти на мові VBA, тому MS Access дозволяє розробляти програми БД практично «з нуля» або написати оболонку для зовнішньої БД.

3.4.2 Структура збереження даних у БД

У процесі збереження даних використовуються таблиці, як ключові носії інформації, та представлені наступними екземплярами:

«*Areas*» - таблиця для збереження даних районів міста, а саме його назви (рис. 3.5).



Рисунок 3.5 – Структура таблиці «Areas»

Первинним ключем є поле Код, яке ідентифікує кожен запис та допомагає зекономити пам'ять у інших таблицях. Поле «Назва» зберігає назву відповідного району.

«*Doctors*» - таблиця для збереження даних лікарів, а саме його спеціальності, ФІО, робочого кабінету та кодом лікарні, у якій він працює (рис. 3.6).

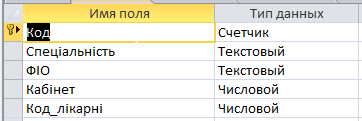


Рисунок 3.6 – Структура таблиці «Doctors»

Первинним ключем, який необхідний для ідентифікації запису є поле Код, поле Код\_лікарні зв’язане з таблицею Hospitals та дозволяє економити пам’ять не записуючи повторно назви лікарень.

«*Hospitals*» - таблиця для збереження даних лікарень, а саме їх назви, адреси, номеру телефону та кодом району, у якому лікарня знаходиться (рис. 3.7).

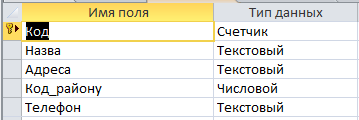
.

Рисунок 3.7 – Структура таблиці «Hospitals»

Первинним ключем, який необхідний для ідентифікації запису є поле Код, поле Код\_району зв’язане з таблицею Area та дозволяє економити пам’ять не записуючи повторно назви районів.

«*Registration*» - таблиця для збереження реєстрацій, а саме коду лікаря, до якого записався пацієнт, дати, часу, коду користувача, який записався до лікаря та поля, що показує чи зайнятий даний час (рис. 3.8).

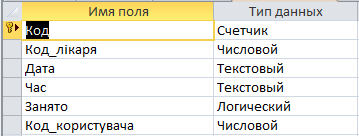


Рисунок 3.8 – Структура таблиці «Registration»

Первинним ключем, який необхідний для ідентифікації запису є поле Код, поля Код\_лікаря та Код\_користувача дозволяють економити пам’ять не записуючи повторно інформацію про лікаря та пацієнта.

«Time\_reg» - таблиця для збереження часу, на який можлива реєстрація (рис. 3.9).



Рисунок 3.8 – Структура таблиці «Time\_reg»

Первинним ключем, який необхідний для ідентифікації запису є поле Код, у полі Час\_запису містяться можливі варіанти часу запису серед яких клієнт обирає зручний для нього час.

«*Users*» - таблиця для збереження зареєстрованих клієнтів, а саме коду клієнта, який здійснив запис, ім’я, прізвища, та його телефону (рис. 3.8).

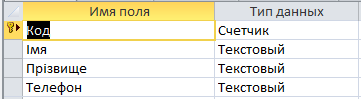
.

Рисунок 3.8 – Структура таблиці «Users»

Первинним ключем, який необхідний для ідентифікації запису є поле Код. Таблиця містить інформацію про клієнтів, що здійснили запис до лікаря.

3.4.2 Основні запити SQL, що було використано

Для виконання запитів до бази даних була обрана мова SQL.

SQL (англ. Structured query language - мова структурованих запитів) - декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних і її модифікація, система контролю за доступом до бази даних. Сам по собі SQL не є ні системою керування базами даних, ні окремим програмним продуктом. Не будучи мовою програмування в тому розумінні, як C або Pascal, SQL може формувати інтерактивні запити або, будучи вбудованою в прикладні програми, виступати в якості інструкцій для керування даними. Стандарт SQL, крім того, вміщує функції для визначення зміни, перевірки і захисту даних.

Під час розробки програми було створено такі запити:

-string sql = "SELECT Код, Назва FROM Areas"; - запит для виведення інформації про райони;

-string sql = "SELECT Hospitals.Назва, Hospitals.Адреса, Hospitals.Код FROM Areas, Hospitals WHERE Areas.Код = Hospitals.Код\_району AND " + "Areas.Код = " + bol; - запит, що виводить інформацію про лікарні, що відповідають вказаним параметрам;

-string sql = "INSERT INTO [User] ( [Імя] ,[Прізвище], [Телефон] ) VALUES ( '" + nameP + "' , '" + surnameP + "' , '" + numberP + "' )"; - запит, що вносить зареєстрованих користувачів до бази;

- та інші.

4 керівництво програміста

В даному розділі розглянуті призначення, умови застосування, характеристика програми, звертання до програми, початкові та вихідні дані та представлені повідомлення.

4.1 Призначення та умови застосування програми

4.1.1 Призначення програми

Програма “Чат-бот для запису до лікаря” призначена для автоматичного віддаленого запису на прийом до лікаря.

4.1.2 Функції програми

Функції, що виконує програма:

* запис до потрібного лікаря у вибраній лакрні на обрані дату та час;
* друк талону запису.

4.1.3 Умови застосування програми

Для роботи програми необхідно мати IBM-сумісний ПК з встановленою операційною системою Windows 7 та вище, з мікропроцесором Intel Core I3

2 ГHz та вище, оперативна пам’ять 2Gb і вище, вільного простору на жорсткому диску 100 Мб, SVGA-монітор, клавіатура та маніпулятор типу “миша”. Для виводу на друк потрібен принтер.

4.2 Характеристика програми

Програма виконана за допомогою мови програмування високого рівня С# в середовищі розробки VS 2019 за допомогою платформи Vindows Forms.

4.2.1 Структура програми

Структура програми складається з 6 класів, 3 з яких – користувацькі, 3 – класи форм. Один з користувацьких класів є абстрактним. Проект містить ці класи та їх реалізацію, файли ресурсів, базу даних SQL .

Ієрархію зображено на рисунку 4.1.

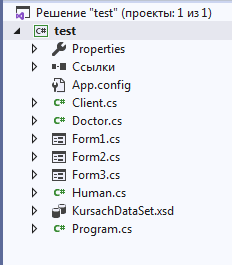


Рисунок 4.1 – Класова ієрархія проєкту

4.2.2 Алгоритм роботи програми

Після запуску виконуваного файлу test.exe з’являється форма входу, що свідчить про початок роботи з програмою. Форма входу зображена на рисунку 4.2.

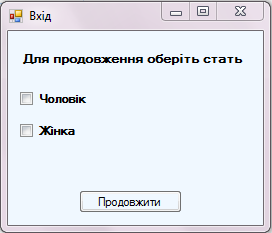


Рисунок 4.2 – Форма входу

Користувач має обрати свою стать та натиснути кнопку “Продовжити”. Після цього відкриється головна форма застосунку(рис.4.3).

На головній формі користувач має вводити відповіді на питання бота у поле, що знаходиться нижче поля виводу. Коли користувач завершить введення особистої інформації наступним кроком буде натиснення кнопки “Друк”.

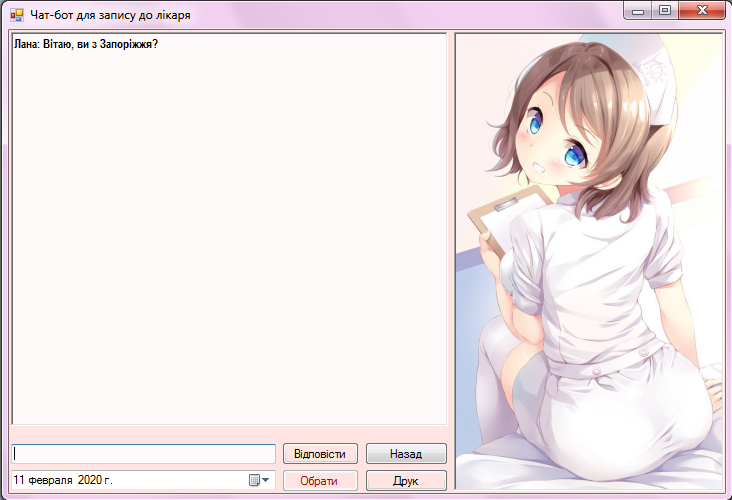


Рисунок 4.3 Головна форма програми

Після того, як користувач натисне кнопку “Друк” з’явиться діалогове вікно підтвердження правильності введених даних (рис.4.4)

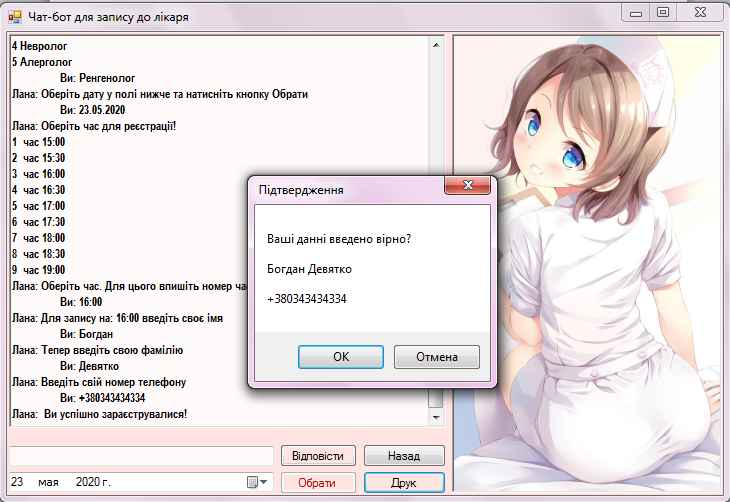


Рисунок 4.4 Діалогове вікно підтвердження

Якщо дані введено вірно, користувач натискає кнопку “ОК” та переходить на вікно друку талона (рис 4.5). При натиснені на кнопку “Роздрукувати” при наявності принтера буде роздруковано талон запису.

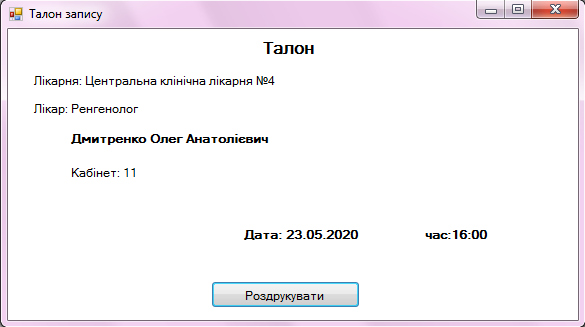


Рисунок 4.5 Вікно друку талона

4.3 Звертання до програми

Для запуску програми потрібно попередньо упевнитися в відповідності характеристик комп’ютера, на який встановлена програма, до системних вимог програми. Потрібно упевнитися у наявності всіх файлів програми.

Звертання виконується через виконуваний файл tеst.exe.

4.4 Вхідні та вихідні дані

4.4.1 Вхідні дані

Вхідні дані уявляють собою файл бази даних, у якому міститься інформація про лікарні, лікарів, їх розклад та ін. Також до вхідних даних належать відповіді на питання від особи, що записується на прийом до лікаря.

4.4.2 Вихідні дані

Роздрукований талон запису та запис про пацієнта у базі даних.

5 керівництво користувача

В даному розділі розглянуто призначення програмного забезпечення, умови та процес його виконання а також повідомлення для користувача.

5.1 Призначення програми

Програма “Чат-бот для запису до лікаря” призначена для автоматичного віддаленого запису на прийом до лікаря.

5.2 Умови виконання програми

5.2.1 Апаратні вимоги програми

Для роботи програми необхідно мати IBM-сумісний ПК з встановленою операційною системою Windows 7 та вище, з мікропроцесором Intel Core I3 2 ГHz та вище, оперативна пам’ять 2Gb і вище, вільного простору на жорсткому диску 100 Мб, SVGA-монітор, клавіатура та маніпулятор типу “миша”. Для виводу на друк потрібен принтер.

5.2.2 Вимоги до користувача

Оскільки програмний продукт створенно для широкої аудиторії спеціальні навички для його використання не потрібні. Єдиною умовою для використання програмного продукту є проживання у м. Запоріжжі.

5.3 Виконання програми

5.3.1 Запуск програми

Для запуску програми потрібно попередньо упевнитися в відповідності характеристик комп’ютера, на який встановлена програма, до системних вимог програми. Другим кроком потрібно упевнитися в наявності всіх бібліотек для роботи програми та в наявності бази даних для повноцінного функціонування.

Звертання до програми передбачене одним способом: через виконуваний файл.

Після запуску виконуваного файлу test.exe з’являється форма входу, що свідчить про початок роботи з програмою. Форма входу зображена на рисунку 5.1.

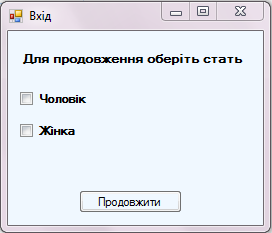


Рисунок 5.1 – Форма входу

5.3.2 Виконання роботи з програмою

Спершу користувач має обрати свою стать та натиснути кнопку “Продовжити”. Після цього відкриється головна форма застосунку(рис.5.2).

На головній формі користувач має вводити відповіді на питання бота у поле, що знаходиться нижче поля виводу. Коли користувач завершить введення особистої інформації наступним кроком буде натиснення кнопки “Друк”.

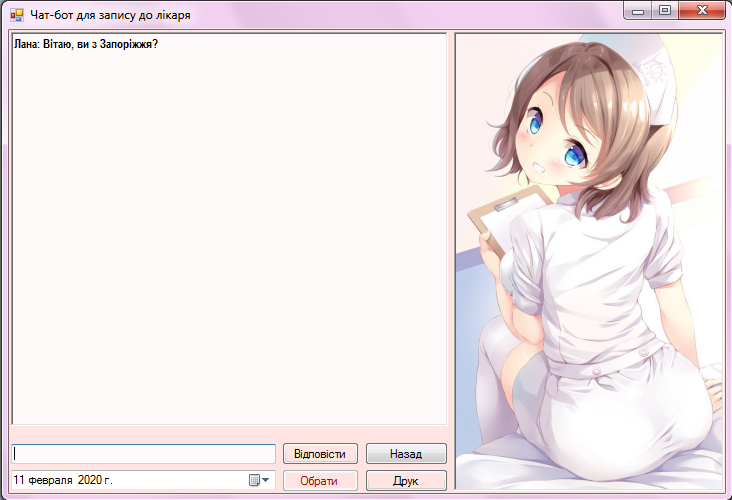


Рисунок 5.2 Головна форма програми

Після того, як користувач натисне кнопку “Друк” з’явиться діалогове вікно підтвердження правильності введених даних (рис.5.3)

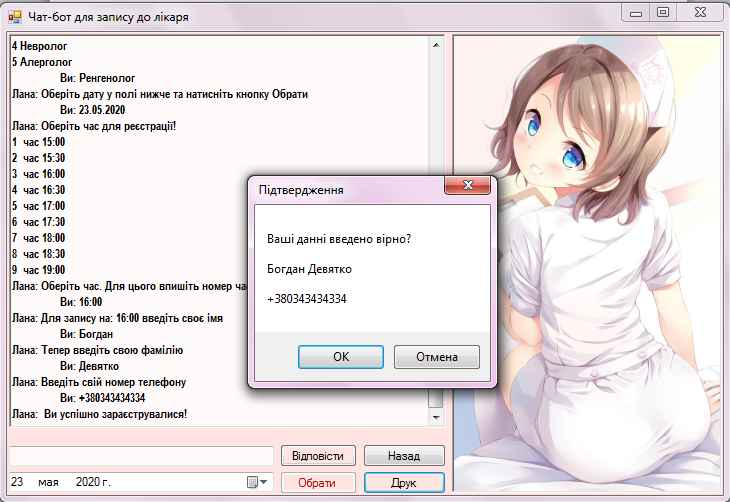


Рисунок 5.3 Діалогове вікно підтвердження

Якщо дані введено вірно, користувач натискає кнопку “ОК” та переходить на вікно друку талона (рис 5.4). При натиснені на кнопку “Роздрукувати” при наявності принтера буде роздруковано талон запису.

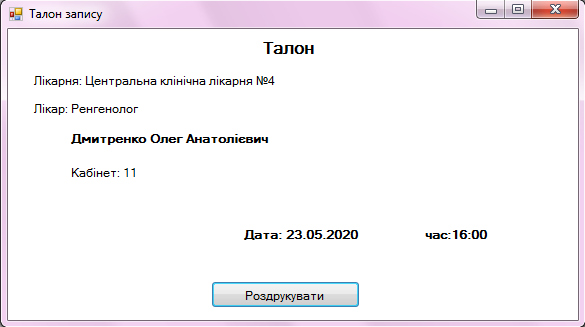


Рисунок 5.4 Вікно друку талона

Висновки

Результатом виконання курсового проекту є застосунок чат-бот для запису до лікаря.

Був виконаний огляд існуючих рішень завдання та аналогічного програмного забезпечення, проаналізовано його переваги та недоліки і розроблено власний програмний продукт.

Для програми було розроблено базу даних у СКБД MS Access.

Було розроблено алгоритм введення користувачем особистої інформації шляхом “розмови” з чат-ботом:

* результат, який відповідає вимогам користувача;
* просте та швидке введення даних та їх зберігання у базі.

Було розроблено максимально прости та зрозумілий інтерфейс, не перевантажений зайвими функціями.

Результатом виконання курсової роботи стало створення можливлості автоматизації запису до лікаря. Тепер кожен житель м. Запоріжжя зможе отримати актуальну та вичерпну інформацію стосовно лікарень у своему районі, крім того він зможе записатися на прийом до потрібного йому лікаря у зручний для себе час дистанційно та в автоматичному режимі. А для керівництва закладів впровадження цього програмного продукту означає зменшення впливу людського фактору на організацію процесу праці, а саме зменшення вирогідності допущення помилок при запису пацієнтів та зменшення затрат на утримання персоналу, який відповідає за цей процес.

Розроблений програмний продукт є актуальним та багатопрофільним.

ПЕРЕЛІК ПОСИЛАНЬ

1)Кириллов В.В. Структуризованный язык запросов (SQL). - СПб.: ИТМО, 1994. - 80 с.

2)Кузнецов С.Д. Стандарты языка реляционных баз данных SQL: краткий обзор //СУБД. - 1996. - №2. - С.6-36

3)Макарова, Нилова, Зеленина - Основы программирования. Учебник с практикумом. -2017. -С. 265-317

4) «Керівництво з мови C#» [Електрон. ресурс]. – Режим доступу: https://docs.microsoft.com/ru-ru/dotnet/csharp/index

5) «Підручник з Microsoft Visual Studio» [Електрон. ресурс]. – Режим доступу: http://www.bhv.ru/books/book.php?id=188476

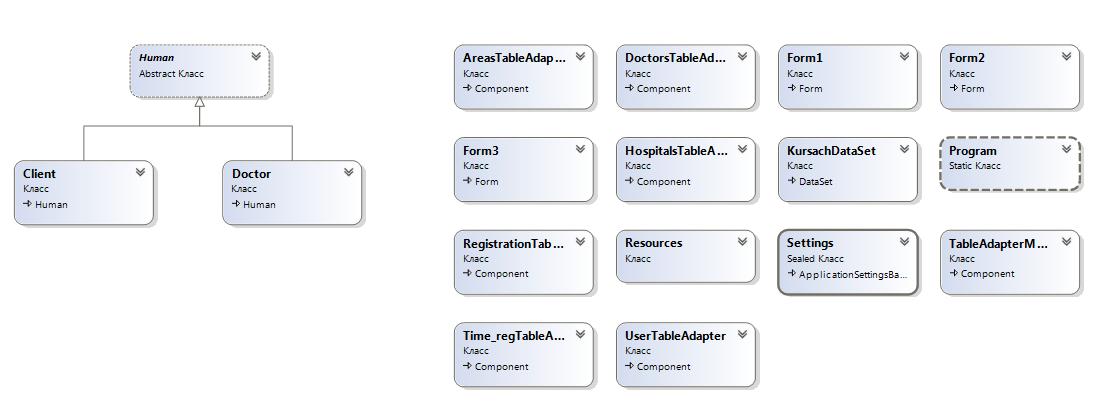
6) «Підручник з мови С#» [Електрон. ресурс]. – Режим доступу: https://metanit.com/sharp/tutorial/

7)Ник Рендольф, Дэвид Гарднер, Майкл Минутилло, Крис Андерсон Microsoft Visual Studio для профессионалов. -2011. -С. 141-166

8)Гэри Хансен, Джэймс Хансен. Базы данных: разработка и управление: Пер. с англ. – М.: ЗАО «Издательство БИНОМ», 1999.

9) «Список лікарень м. Запоріжжя» [Електрон. ресурс]. – Режим доступу: http://zabor.zp.ua/med/bolnici.htm

Додаток А Діаграма класів



Додаток Б Код Програми

Б1 – Клас Human

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace test

{

abstract public class Human

{

public string fullName;

}

}

Б2 – Клас Doctor

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace test

{

public class Doctor : Human

{

public string specialty;

public string cabinet;

}

}

Б3 – Клас Client

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace test

{

public class Client : Human

{

public string telephone;

}

}

Б4 – Форма form1

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.OleDb;

using System.Runtime.InteropServices;

namespace test

{

public partial class Form1 : Form

{

public string bot;

public static string ConnectionString = @"Provider=Microsoft.Jet.OLEDB.4.0; " +

"Data Source= Kursach.mdb";

private OleDbConnection connection;

[DllImport("kernel32.dll", SetLastError = true)]

[return: MarshalAs(UnmanagedType.Bool)]

static extern bool AllocConsole();

[DllImport("kernel32.dll", SetLastError = true)]

[return: MarshalAs(UnmanagedType.Bool)]

static extern bool FreeConsole();

public Form1()

{

InitializeComponent();

connection = new OleDbConnection();

connection.ConnectionString = ConnectionString;

connection.Open();

}

private volatile string Otvet;

private volatile int p1;

private volatile int p2;

private volatile int p3;

private volatile int p4;

private volatile int timeDetector = 0;

private volatile string rayenP;

private volatile string nameBol;

private volatile string nameDoc;

private volatile string dataR;

private volatile string timeR;

private volatile string nameP;

private volatile string surnameP;

private volatile string numberP;

private volatile int k\_P;

private volatile int animeVopros = 0;

private volatile int pBol;

private volatile int k\_Doc;

private volatile string[] names\_doc = new string[10];

private volatile string[] names\_bol = new string[10];

private volatile string[] name\_rayena = new string[10];

private volatile string[] kod\_bol = new string[10];

private volatile string[] zapisy = new string[20];

private volatile string[] zapisy\_2 = new string[200];

private volatile int pZap;

private volatile int pZap\_2;

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

{

Otvet = textBox1.Text;

textBox1.Text = "";

if (animeVopros == 0)

{

Otvet = Otvet.ToLower();

zpIF(Otvet);

}

else if (animeVopros == 1)

{

areaZP(Otvet);

}

else if (animeVopros == 2)

{

Otvet = Otvet.ToLower();

hospitalZP(Otvet);

}

else if (animeVopros == 3)

{

doctorZP(Otvet);

}

else if (animeVopros == 4)

{

dataZP();

}

else if (animeVopros == 5)

{

button6.Enabled = false;

timeZP(Otvet);

}

else if (animeVopros == 6)

{

nameYou(Otvet);

}

else if (animeVopros == 7)

{

textBox1.Text = "+380";

textBox1.MaxLength = 13;

surnameYou(Otvet);

}

else if (animeVopros == 8)

{

numberYou(Otvet);

}

else if (animeVopros == 9)

{

richTextBox1.Text = richTextBox1.Text + "\n" + bot + " Ви вже записалися!";

}

textBox1.Focus();

richTextBox1.SelectionStart = richTextBox1.Text.Length;

richTextBox1.ScrollToCaret();

}

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

{

textBox1.Focus();

}

public async void zpIF(string otvet)

{

richTextBox1.SelectionAlignment = HorizontalAlignment.Right;

if (otvet == "так")

{

richTextBox1.Text = richTextBox1.Text + "\n" + "\t" + "Ви: Так";

animeVopros++;

richTextBox1.Text = richTextBox1.Text + "\n" + bot + "В якому районі ви проживаєте?";

string sql = "SELECT Код, Назва FROM Areas";

OleDbCommand myCommand = new OleDbCommand(sql, connection);

OleDbDataReader reader = myCommand.ExecuteReader();

p1 = 0;

richTextBox1.Text = richTextBox1.Text + "\n" + bot + "Для того, щоб обрати потрібний район введіть відповідний йому номер";

while (reader.Read())

{

richTextBox1.Text = richTextBox1.Text + "\n" + reader[0].ToString() + " " + reader[1].ToString();

name\_rayena[p1++] = reader[1].ToString();

}

reader.Close();

p1--;

}

else if (otvet == "ні")

{

richTextBox1.Text = richTextBox1.Text + "\n" + "\t" + "Ви: Ні";

richTextBox1.Text = richTextBox1.Text + "\n" + bot + "Нажаль я можу працювати лише з лікарнями міста Запоріжжя";

}

else

{

richTextBox1.Text = richTextBox1.Text + "\n" + "\t" + "Ви: " + otvet;

richTextBox1.Text = richTextBox1.Text + "\n" + bot + "Будь-ласка дайте відповідь так/ні";

}

}

public async void areaZP(string otvet)

{//

int a;

try

{

a = Int32.Parse(otvet) - 1;

if (a >= 0 && a <= p1)

{

rayenP = name\_rayena[a];

richTextBox1.Text = richTextBox1.Text + "\n" + "\t" + "Ви: " + rayenP;

areaZP2(Int32.Parse(otvet));

}

else

{

richTextBox1.Text = richTextBox1.Text + "\n" + bot + " вы вводите что то не то, введите номер раена 22";

}

}

catch (FormatException)

{

richTextBox1.Text = richTextBox1.Text + "\n" + bot + " вы вводите что то не то, введите номер раена";

}

//

}

public async void areaZP2(int bol)

{

string sql = "SELECT Hospitals.Назва, Hospitals.Адреса, Hospitals.Код FROM Areas, Hospitals WHERE Areas.Код = Hospitals.Код\_району AND " + "Areas.Код = " + bol;

OleDbCommand myCommand = new OleDbCommand(sql, connection);

OleDbDataReader reader = myCommand.ExecuteReader();

int p = 1;

p2 = 0;

richTextBox1.Text = richTextBox1.Text + "\n" + bot;

while (reader.Read())

{

richTextBox1.Text = richTextBox1.Text + "\n" + p++ + " " + reader[0].ToString() + "\t" + reader[1].ToString();

kod\_bol[p - 2] = reader[2].ToString();

names\_bol[p2++] = reader[0].ToString();

}

reader.Close();

pBol = p - 1;

p2--;

animeVopros++;

}

public async void hospitalZP(string otvet)

{

if (pBol == 1)

{

if(otvet == "так")

{

nameBol = names\_bol[0];

richTextBox1.Text = richTextBox1.Text + "\n" + "\t" + "Ви: Так";

hospitalZP\_2(kod\_bol[0]);

}

else if (otvet == "ні")

{

richTextBox1.Text = richTextBox1.Text + "\n" + "\t" + "Ви: Ні";

richTextBox1.Text = richTextBox1.Text + "\n" + bot + " Інших лікарень в цьому районі немає";

}

else

{

richTextBox1.Text = richTextBox1.Text + "\n" + "\t" + "Ви: " + otvet;

richTextBox1.Text = richTextBox1.Text + "\n" + bot + "Будь-ласка дайте відповідь так/ні";

}

}

else

{

//

int a;

try

{

a = Int32.Parse(otvet) - 1;

if (a >= 0 && a <= p2)

{

nameBol = names\_bol[a];

richTextBox1.Text = richTextBox1.Text + "\n" + "\t" + "Ви: " + nameBol;

int с = Int32.Parse(otvet) - 1;

hospitalZP\_2(kod\_bol[с]);

}

else

{

richTextBox1.Text = richTextBox1.Text + "\n" + bot + " вы вводите что то не то, введите номер больницы 22";

}

}

catch (FormatException)

{

richTextBox1.Text = richTextBox1.Text + "\n" + bot + " вы вводите что то не то, введите номер больницы";

}

//

}

}

public async void hospitalZP\_2(string bol)

{

label1.Text = nameBol;

string sql = "SELECT Doctors.Спеціальність, Doctors.Код FROM Doctors, Hospitals WHERE Doctors.Код\_лікарні = Hospitals.Код AND " + "Doctors.Код\_лікарні = " + bol;

OleDbCommand myCommand = new OleDbCommand(sql, connection);

OleDbDataReader reader = myCommand.ExecuteReader();

int p = 1;

p3 = 0;

richTextBox1.Text = richTextBox1.Text + "\n" + bot;

while (reader.Read())

{

richTextBox1.Text = richTextBox1.Text + "\n" + p++ + " " + reader[0].ToString();

kod\_bol[p - 2] = reader[1].ToString();

names\_doc[p3++] = reader[0].ToString();

}

reader.Close();

p3--;

animeVopros++;

}

public async void doctorZP(string Otvet)

{

//

int a;

try

{

a = Int32.Parse(Otvet) - 1;

if (a >= 0 && a <= p3)

{

nameDoc = names\_doc[a];

richTextBox1.Text = richTextBox1.Text + "\n" + "\t" + "Ви: " + nameDoc;

int с = Int32.Parse(Otvet) - 1;

k\_Doc = Int32.Parse(kod\_bol[с]);

label2.Text = "" + nameDoc;

animeVopros++;

dataZP();

}

else

{

richTextBox1.Text = richTextBox1.Text + "\n" + bot + " вы вводите что то не то, введите номер доктора 22";

}

}

catch (FormatException)

{

richTextBox1.Text = richTextBox1.Text + "\n" + bot + " вы вводите что то не то, введите номер доктора";

}

//

}

public async void dataZP()

{

richTextBox1.Text = richTextBox1.Text + "\n" + bot + "Оберіть дату у полі нижче та натисніть кнопку Обрати";

button6.Enabled = true;

}

public async void timeZP(string otvet)

{

//

int a;

try

{

a = Int32.Parse(otvet) - 1;

if (a >= 0 && a <= p4)

{

timeR = zapisy[a];

label6.Text = timeR;

richTextBox1.Text = richTextBox1.Text + "\n" + "\t" + "Ви: " + timeR;

richTextBox1.Text = richTextBox1.Text + "\n" + bot + "Для запису на: " + timeR + " введіть своє імя";

animeVopros++;

}

else

{

richTextBox1.Text = richTextBox1.Text + "\n" + bot + " вы вводите что то не то, введите номер времени";

}

}

catch (FormatException)

{

richTextBox1.Text = richTextBox1.Text + "\n" + bot + " вы вводите что то не то, введите номер времени";

}

}

public async void nameYou(string otvet)

{

nameP = otvet;

richTextBox1.Text = richTextBox1.Text + "\n" + "\t" + "Ви: " + nameP;

richTextBox1.Text = richTextBox1.Text + "\n" + bot + "Тепер введіть свою фамілію";

animeVopros++;

}

public async void surnameYou(string otvet)

{

surnameP = otvet;

richTextBox1.Text = richTextBox1.Text + "\n" + "\t" + "Ви: " + surnameP;

richTextBox1.Text = richTextBox1.Text + "\n" + bot + "Введіть свій номер телефону";

animeVopros++;

}

public async void numberYou(string otvet)

{

numberP = otvet;

richTextBox1.Text = richTextBox1.Text + "\n" + "\t" + "Ви: " + numberP;

label7.Text = numberP;

string sql = "INSERT INTO [User] ( [Імя] ,[Прізвище], [Телефон] ) VALUES ( '" + nameP + "' , '" + surnameP + "' , '" + numberP + "' )";

OleDbCommand myCommand = new OleDbCommand(sql, connection);

myCommand.ExecuteNonQuery();

string sql3 = "SELECT Код FROM [User] WHERE [Імя] = '" + nameP + "' AND [Прізвище] = '" + surnameP + "' AND [Телефон] = '" + numberP + "'";

myCommand = new OleDbCommand(sql3, connection);

OleDbDataReader reader = myCommand.ExecuteReader();

while (reader.Read())

{

k\_P = Int32.Parse(reader[0].ToString());

}

reader.Close();

string sql2 = "INSERT INTO [Registration] ( [Код\_лікаря] , [Дата] , [Час], [Занято], [Код\_користувача]) VALUES ( " + k\_Doc + " , '" + dataR + "' , '" + timeR + "' , True , " + k\_P + " )";

myCommand = new OleDbCommand(sql2, connection);

myCommand.ExecuteNonQuery();

//выводит данные о докторе

string sql4 = "SELECT Спеціальність, ФІО, Кабінет FROM [Doctors] WHERE [Код] = " + 1;

myCommand = new OleDbCommand(sql4, connection);

OleDbDataReader reader2 = myCommand.ExecuteReader();

Doctor doctor = new Doctor();

while (reader2.Read())

{

doctor.specialty = reader2[0].ToString();

doctor.fullName = reader2[1].ToString();

label3.Text = doctor.fullName;

doctor.cabinet = reader2[2].ToString();

label4.Text = doctor.cabinet;

}

reader2.Close();

richTextBox1.Text = richTextBox1.Text + "\n" + bot + " Ви успішно зараєструвалися!";

animeVopros++;

}

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

{

richTextBox1.Text = bot + "Вітаю, ви з Запоріжжя?";

}

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

connection.Close();

Application.Exit();

}

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

{

}

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

{

richTextBox1.Text = richTextBox1.Text + "\n" +"\t"+ "Ви: " + dateTimePicker1.Value.ToString("dd.MM.yyyy");

dataR = dateTimePicker1.Value.ToString("dd.MM.yyyy");

DateTime date1 = dateTimePicker1.Value;

DateTime date2 = DateTime.Today;

int dateProverka = date1.CompareTo(date2);

//richTextBox1.Text = richTextBox1.Text + "\n" + "\t" + "Ви: " + dateProverka;

if (dateProverka >= 0)

{

if (timeDetector == 0)

{

animeVopros++;

timeDetector++;

}

string sql = "SELECT Time\_reg.Час\_запису FROM Time\_reg ";

OleDbCommand myCommand = new OleDbCommand(sql, connection);

OleDbDataReader reader = myCommand.ExecuteReader();

pZap = 0;

while (reader.Read())

{

zapisy[pZap++] = reader[0].ToString();

}

reader.Close();

richTextBox1.Text = richTextBox1.Text + "\n" + bot + "Оберіть час для реєстрації!";

sql = "SELECT Registration.Час FROM Doctors, Registration WHERE Doctors.Код = Registration.Код\_лікаря AND " + "Doctors.Код = " + k\_Doc + " AND Registration.Дата = '" + dataR + "'";

myCommand = new OleDbCommand(sql, connection);

OleDbDataReader reader2 = myCommand.ExecuteReader();

pZap\_2 = 0;

while (reader2.Read())

{

zapisy\_2[pZap\_2++] = reader2[0].ToString();

}

reader2.Close();

int z = 0;

int k;

while (z < pZap\_2)

{

k = 0;

while (k < pZap)

{

if (zapisy[k] == zapisy\_2[z])

{

for (int i = k; i < pZap; ++i)

{

zapisy[i] = zapisy[i + 1];

}

pZap--;

}

k++;

}

z++;

}

z = 0;

p4 = 0;

while (z < pZap)

{

richTextBox1.Text = richTextBox1.Text + "\n" + ++z + " час " + zapisy[z - 1];

p4++;

}

p4--;

richTextBox1.Text = richTextBox1.Text + "\n" + bot + "Оберіть час. Для цього впишіть номер часу, який вам підходить.";

}

else

{

richTextBox1.Text = richTextBox1.Text + "\n" + bot + "Запис на цей час неможливий.";

}

richTextBox1.SelectionStart = richTextBox1.Text.Length;

richTextBox1.ScrollToCaret();

}

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

{

if (animeVopros == 8)

{

char number = e.KeyChar;

if (!Char.IsDigit(number))

{

e.Handled = true;

}

}

else if (animeVopros == 6)

{

char l = e.KeyChar;

if ((l < 'А' || l > 'я') && l != '\b' && l != '.')

e.Handled = true;

}

else if (animeVopros == 7)

{

char l = e.KeyChar;

if ((l < 'А' || l > 'я') && l != '\b' && l != '.')

e.Handled = true;

}

}

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

{

Client client = new Client();

client.fullName = nameP+" "+surnameP;

client.telephone = numberP;

//client.GetInfo();

DialogResult dialogResult = MessageBox.Show("Ваші данні введено вірно?" + "\n\n" + client.fullName + "\n\n" + client.telephone, " Підтвердження", MessageBoxButtons.OKCancel);

if (dialogResult == DialogResult.OK)

{

Form3 Zz = new Form3();

//likarnya

Zz.label1.Text = Zz.label1.Text +" "+ label1.Text;

//likar

Zz.label2.Text = Zz.label2.Text + " " + label2.Text;

//имя доктора

Zz.label3.Text = label3.Text;

//кабинет

Zz.label4.Text = Zz.label4.Text + label4.Text;

//дата

Zz.label6.Text = Zz.label6.Text+ dataR;

//время

Zz.label7.Text = Zz.label7.Text+ timeR;

Zz.Show();

}

else if (dialogResult == DialogResult.Cancel)

{

//do something else

}

}

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

{

animeVopros--;

}

}

}

Б4 – Форма form2

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace test

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

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

{

if (checkBox1.Checked)

{

Form1 C = new Form1();

C.BackColor = Color.MistyRose;

C.bot = "Лана: ";

C.richTextBox1.BackColor = Color.Snow;

C.textBox1.BackColor = Color.Snow;

C.button1.BackColor = Color.MistyRose;

C.button6.BackColor = Color.MistyRose;

C.button2.BackColor = Color.MistyRose;

Random rnd = new Random();

int a = rnd.Next(1, 4);

C.panel1.BackgroundImage = Image.FromFile(a + ".jpg");

C.Show();

}

else if (checkBox2.Checked)

{

Form1 C = new Form1();

C.BackColor = Color.Lavender;

C.bot = "Віктор: ";

C.panel1.BackgroundImage = Image.FromFile("11.jpg");

C.richTextBox1.BackColor = Color.AliceBlue;

C.textBox1.BackColor = Color.AliceBlue;

C.button1.BackColor = Color.Lavender;

C.button6.BackColor = Color.Lavender;

C.button2.BackColor = Color.Lavender;

C.Show();

}

else

label2.Visible = true;

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox1.Checked)

checkBox2.Enabled = false;

if (checkBox1.Checked == false)

checkBox2.Enabled = true;

label2.Visible = false;

}

private void checkBox2\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox2.Checked)

checkBox1.Enabled = false;

if (checkBox2.Checked == false)

checkBox1.Enabled = true;

label2.Visible = false;

}

}

}

Б4 – Форма form3

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace test

{

public partial class Form3 : Form

{

public Form3()

{

InitializeComponent();

}

[System.Runtime.InteropServices.DllImport("gdi32.dll")]

public static extern long BitBlt(IntPtr hdcDest, int nXDest, int nYDest,

int nWidth, int nHeight, IntPtr hdcSrc, int nXSrc, int nYSrc, int dwRop);

private Bitmap memoryImage;

private void CaptureScreen()

{

Graphics mygraphics = this.CreateGraphics();

Size s = this.Size;

memoryImage = new Bitmap(s.Width, s.Height, mygraphics);

Graphics memoryGraphics = Graphics.FromImage(memoryImage);

IntPtr dc1 = mygraphics.GetHdc();

IntPtr dc2 = memoryGraphics.GetHdc();

BitBlt(dc2, 0, 0, this.ClientRectangle.Width,

this.ClientRectangle.Height, dc1, 0, 0, 13369376);

mygraphics.ReleaseHdc(dc1);

memoryGraphics.ReleaseHdc(dc2);

}

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

{

printButton.Visible = false;

CaptureScreen();

printDocument1.Print();

}

private void printDocument1\_PrintPage(object sender, System.Drawing.Printing.PrintPageEventArgs e)

{

e.Graphics.DrawImage(memoryImage, 0, 0);

}

}

}

Додаток В інтерфейс програми

