**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА**

**Кафедра Систем автоматизованого проектування**

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

**До бакалаврської кваліфікаційної роботи на тему**

|  |
| --- |
| Розробка платформи для благодійних пожертвувань та волонтерської діяльності |
|  |
|  |
|  |

|  |
| --- |
| КН-409, Погуляєва В.В. |

Студента групи

(шифр, прізвище та ініціали)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Керівник роботи** |  |  |  |  |
| **Консультанти** |  |  |  | (підпис) |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **Завідувач кафедри** |  |  |  |  |
|  |  |  |  |  |
|  |  | «\_\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ р. | | |
| **Рецензент** |  |  |  |  |

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА**

|  |
| --- |
| *ІКНІ* |

|  |
| --- |
| САПР |

Інститут Кафедра

|  |
| --- |
| 6.050101 «Комп’ютерні науки" |

Спеціальність

ЗАТВЕРДЖУЮ

Зав. кафедрою\_\_\_\_\_\_\_\_\_\_\_\_

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

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 р.

**ЗАВДАННЯ**

|  |
| --- |
| КН-409 |

|  |
| --- |
| бакалавр |

на кваліфікаційну роботу (проект) студента групи ОКР

|  |
| --- |
| Погуляєва Владислава Володимировича |

(прізвище, ім’я, по батькові)

|  |
| --- |
| Розробка платформи для благодійних пожертвувань та волонтерської діяльності |

1. Тема проекту (роботи)

затверджена наказом по університету від «\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_ р. №\_\_\_\_\_\_\_\_\_

1. Тернім здачі студентом закінченого проекту (роботи) «\_\_\_» \_\_\_\_\_\_\_\_ 20\_\_ р.
2. Вихідні дані до проекту (роботи) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

1. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить робити)

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

1. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

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

1. Консультанти по проекту (роботі) із зазначенням розділів проекту, що стосується їх.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Розділ | Консультанти | Завдання видав | | Завдання прийняв | |
| Підпис | Дата | Підпис | Дата |
| Економ. |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

1. Дата видачі завдання «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Завдання прийняв до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назв і етапів дипломного проекту (роботи) | Термін виконання етапів проекту (роботи) | Примітка |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Анотація**

Погуляєв Владислав. Розробка платформи для благодійних пожертвувань та волонтерської діяльності. В результаті даної бакалаврської кваліфікаційної роботи було розроблено платформу для благодійних пожертвувань та волонтерської діяльності, яка оперує даними на локальній або віддаленій базі даних. Після вдалого з’єднання з базою даних користувач може переглядати оголошення і сортувати їх.

Також користувач може зареєструвати обліковий запис у системі. Отримати статус волонтера. Також відкрита можливості приєднуватись до організацій, для користувача волонтера а також створення оголошень для волонтером.

Інтерфейс розроблений таким чином аби бути легким та не набридливим, а також звичайно зручним.

Після проведеного аналізу системи та економічної оцінки можна сказати, що даний продукт є потрібним та в його розробку потрібно вкласти необхідні ресурси.

**Abstract**

Pohulyaev Vladyslav. Development of a platform for charitable donations and volunteering. As a result of this bachelor's qualification work, a platform for charitable donations and volunteer activities was developed, which operates with data on a local or remote database. After a successful connection to the database, the user can view ads and sort them.

The user can also register an account in the system. Get volunteer status. It is also possible to join organizations, for the volunteer user, as well as to create ads for the volunteer.

The interface is designed in such a way to be easy and not annoying, and also of course convenient.

After the analysis of the system and the economic evaluation, it can be said that this product is necessary and the necessary resources should be invested in its development.

**Список позначень і скорочень**

ПЗ – Програмне забезпечення

UML – Unified Modeling Language (Уніфікована мова моделювання)

VS – Visual Studio

БД – База даних

DB – Data base (База даних)

API – Application Programming Interface (Прикладни́й програ́мний інтерфе́йс)

ER – Entity-Relationship (Сутність – Зв’язок)

ID – Identity (Ідентифікатор)

ОС – Операційна система

UI – User interface (Інтерфейс користувача)

XML – Extensible Markup Language (Розши́рювана мо́ва розмі́тки)

JSON – JavaScript Object Notation (об'єктний запис JavaScript)

ЗМІСТ

[ВСТУП 9](#_Toc484461264)

[РОЗДІЛ 1. СИСТЕМИ ОБЛІКУ ПАЦІЄНТІВ 10](#_Toc484461265)

[1.1. Системи обліку пацієнтів. Поняття та види 10](#_Toc484461266)

[1.2. Приклади існуючих систем обліку пацієнтів. 11](#_Toc484461267)

[1.2.1. Медична інформаційна система 11](#_Toc484461268)

[1.2.2 Медична система «Доктор Елекс» 12](#_Toc484461269)

[1.2.3 МІС «ЕМСіМЕД» 13](#_Toc484461270)

[1.2.4 Система Meditex 14](#_Toc484461271)

[1.2.5 Програмне забезпечення Myrian 15](#_Toc484461272)

[1.2.6 Система Centrak 16](#_Toc484461273)

[1.3 Збереження даних 17](#_Toc484461274)

[1.3.1 XML-формат 17](#_Toc484461275)

[1.3.2 JSON-формат 18](#_Toc484461276)

[1.3.3 Бінарний формат 18](#_Toc484461277)

[1.3.4 База даних 19](#_Toc484461278)

[Висновок 22](#_Toc484461279)

[РОЗДІЛ 2. ВИКОРИСТАНІ ЗАСОБИ РОЗРОБКИ 23](#_Toc484461280)

[2.1. С++ мова 23](#_Toc484461281)

[2.2. Visual Studio 26](#_Toc484461282)

[2.3. Qt Framework 31](#_Toc484461283)

[2.4. SQL мова 34](#_Toc484461284)

[РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ 37](#_Toc484461285)

[3.1. Розробка архітектури системи 37](#_Toc484461286)

[3.2. Розробка бази даних 38](#_Toc484461287)

[3.3. Розробка дизайну програми 41](#_Toc484461288)

[3.4. Розробка системи 43](#_Toc484461289)

[3.5. Збірка та розгортання 45](#_Toc484461290)

[3.6. Основний функціонал системи 52](#_Toc484461291)

[Висновок 59](#_Toc484461292)

[РОЗДІЛ 4. ЕКОНОМІЧНА ЧАСТИНА 60](#_Toc484461293)

[4.1. Економічна характеристика програмного продукту 60](#_Toc484461294)

[4.2. Інформаційне забезпечення та формування гіпотез щодо потреби проведення дослідження 60](#_Toc484461295)

[4.3. Оцінювання та аналіз факторів зовнішнього та внутрішнього середовищ. 61](#_Toc484461296)

[4.4. Формування стратегічних альтернатив 63](#_Toc484461297)

[4.5. Бюджетування 65](#_Toc484461298)

[4.6. Вибір стратегії 71](#_Toc484461299)

[Висновок 72](#_Toc484461300)

[ВИСНОВОКИ 73](#_Toc484461301)

[Використана література. 74](#_Toc484461302)

Додаток А. Графічні матеріали……………………………………………………76

# ВСТУП

Проблеми, які може вирішити платформа для благодійних пожертвувань та волонтерської діяльності, безліч. На жаль, багато з них стосуються необхідності забезпечення ефективної співпраці між благодійними організаціями та волонтерами, а також необхідності підвищення ступеню довіри до процесу благодійних пожертвувань.

Одним з ключових завдань платформи для благодійних пожертвувань та волонтерської діяльності є забезпечення можливості вибору найбільш ефективної та надійної організації для пожертвувань та підтримки проектів, які відповідають інтересам волонтерів.

Також важливо забезпечити зручний та безпечний процес збору та передачі коштів, який забезпечує максимальну прозорість та довіру до пожертвувань. Відсутність цих моментів може стати перешкодою для багатьох людей, які бажають підтримати благодійні проекти, але не мають достатньої інформації або довіри до процесу пожертвування.

Платформа також може допомогти у залученні нових волонтерів та управлінні проектами, що дозволить організаціям зосередитися на виконанні своїх місій та цілей, а не на адміністративних питаннях.

Окрім цього, платформа може допомогти вирішувати проблеми пов'язані з необхідністю координації діяльності різних благодійних організацій та волонтерів, які можуть працювати на одному проекті. Завдяки цьому можна забезпечити біль РОЗДІЛ 1. СИСТЕМИ ОБЛІКУ ПАЦІЄНТІВ

# 1.1. Платформа для благодійних пожертвувань та волонтерської діяльності. Поняття та види

Платформа для благодійних пожертвувань та волонтерської діяльності – це програмне забезпечення, яке дозволяє створювати волонтерські оголошення та переглядати їх , використовуючи базу даних.

База даних визначається як зберігальне приміщення для даних, які організовані за конкретними правилами. Існують різні критерії класифікації баз даних. За моделлю організації даних можна виділити наступні типи:

* Ієрархічна база даних, яка складається з об'єктів різних рівнів та має зв'язки "предок-нащадок".
* Мережна база даних, яка має схожу структуру з ієрархічною, але об'єкт може мати декілька предків.
* Реляційна база даних, що зберігає дані у вигляді таблиць та є найпоширенішою моделлю серед СКБД.
* Об'єктно-орієнтована база даних, де дані відображаються у вигляді об'єктів. За розміщенням даних, бази даних поділяються на локальні (централізовані) та розподілені, які можуть бути виконані на різних платформах, таких як веб, стаціонарні системи, мобільні додатки та системи для влаштованих пристроїв.

# 1.2. Приклади існуючих платформ для благодійних пожертвувань та волонтерської діяльності.

# 1.2.1. GoFundMe

GoFundMe - це онлайн-платформа збору коштів для різних цілей, від медичних потреб до благодійності та технологій. Заснована в 2010 році, платформа допомагає людям зібрати кошти на проекти та цілі, що мають значення для них та їхніх близьких.

Рис.1.2.1.1 Логотип GoFundMe.

За допомогою GoFundMe користувачі можуть створювати особисті сторінки та розповідати про свої історії, мету та потребу в коштах. Люди можуть додавати фотографії та відео, щоб краще розповісти про своє запитання на допомогу. Користувачі також можуть ділитися своїми сторінками з друзями та родичами, які можуть пожертвувати на їхню підтримку.

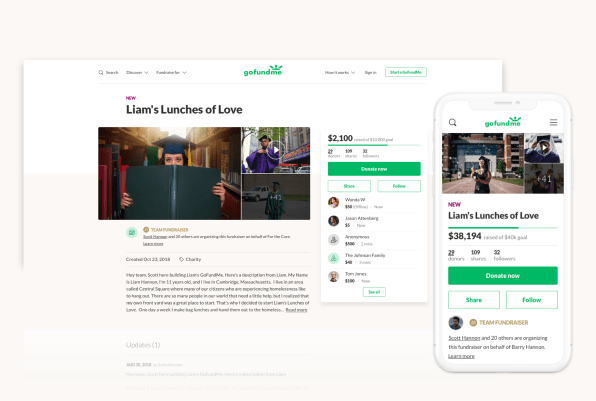


Рис.1.2.1.2 Інтерфейс GoFundMe.

Одна з найбільш важливих функцій GoFundMe полягає в тому, що користувачі можуть збирати гроші швидко та ефективно, не залежно від того, де вони живуть. Таким чином, люди можуть отримати допомогу та підтримку з усього світу.

GoFundMe також є корисною для благодійних організацій та некомерційних організацій, які можуть створювати свої власні сторінки та збирати кошти на благодійні проекти та інші потреби. Це може допомогти підвищити увагу до їхніх проблем та підтримати їхню місію.

# 1.2.2 DonorsChoose

# DonorsChoose - це онлайн-платформа для збору коштів на проекти, які пов'язані з освітою. Заснована в 2000 році, платформа допомагає вчителям з США створювати проекти та збирати кошти на придбання матеріалів та обладнання для своїх класів.

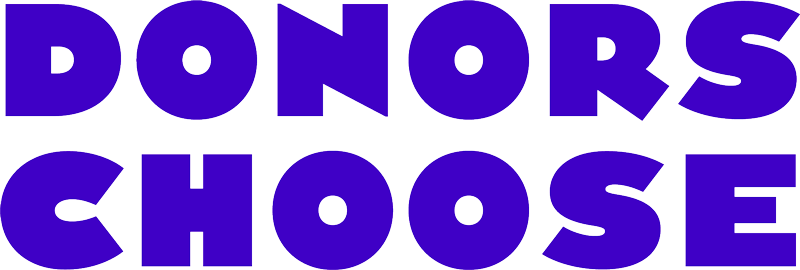


Рис.1.2.2.1 Логотип DonorsChoose.

За допомогою DonorsChoose вчителі можуть створювати особисті сторінки та описувати свої проекти та потреби. Люди можуть знайти проекти, які їм цікаві та пожертвувати на їхню підтримку. Користувачі також можуть ділитися своїми сторінками з друзями та родичами, щоб залучити ще більше підтримки.

Одна з головних переваг DonorsChoose полягає в тому, що це дозволяє вчителям отримати необхідні матеріали та обладнання для своїх класів, які можуть допомогти покращити якість освіти та навчання учнів. Крім того, це допомагає підвищити зацікавленість учнів та збільшити їхню мотивацію до навчання.

DonorsChoose також є корисною для тих, хто хоче пожертвувати на освітні проекти та допомогти вчителям та учням отримати необхідні ресурси. Це може допомогти зробити позитивний внесок у майбутнє дітей та підтримати освітню місію вчителів.

Узагалі, DonorsChoose є потужним інструментом для збору коштів та підтримки освітніх проектів. Це дозволяє вчителям отримати необхідні матеріали та обладнання, а учням - якісну освіту та навчання.

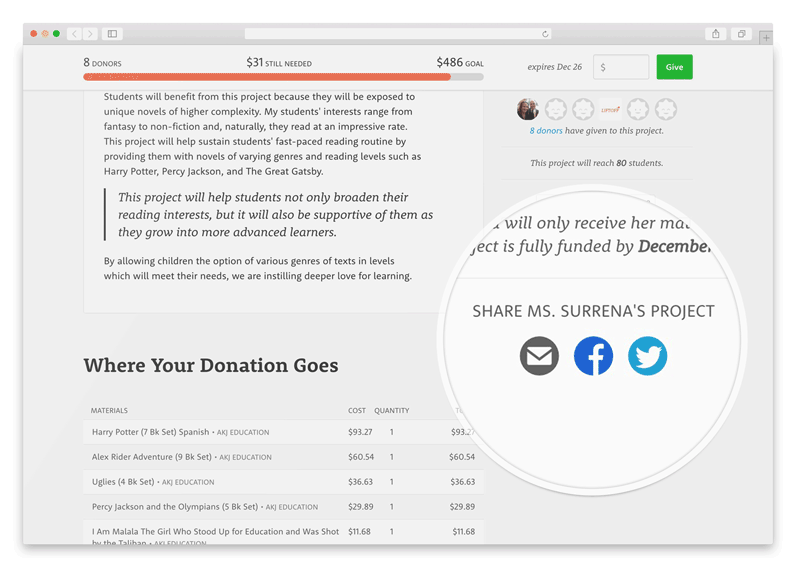


Рис.1.2.2.2 Інтерфейс DonorsChoose.

# 1.2.3 VolunteerMatch

VolunteerMatch - це онлайн-платформа, яка з'єднує волонтерів та неприбуткові організації. Заснована в 1998 році, платформа допомагає знаходити волонтерські можливості у багатьох країнах світу, допомагаючи організаціям та волонтерам знаходити один одного.



Рис.1.2.3.1 Логотип VolunteerMatch.

За допомогою VolunteerMatch неприбуткові організації можуть розміщувати вакансії для волонтерів та знаходити людей, які готові надати свою безоплатну допомогу. Волонтери також можуть шукати відповідні вакансії на платформі та зв'язуватися з організаціями.

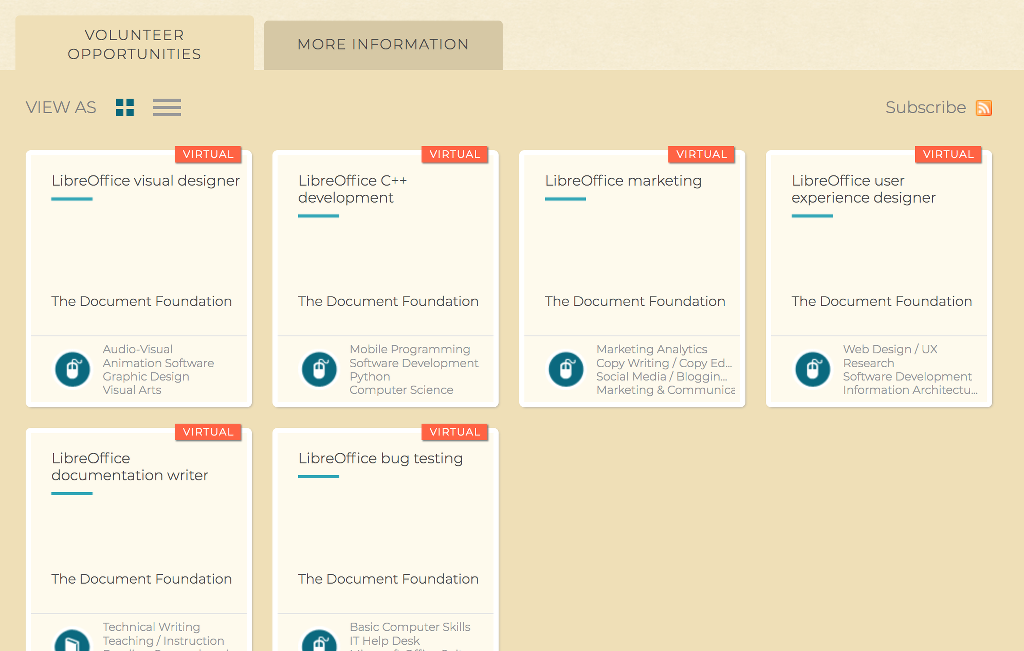


Рис.1.2.3.2 Інтерфейс VolunteerMatch.

Одна з головних переваг VolunteerMatch полягає в тому, що волонтери можуть знайти відповідну волонтерську роботу, яка відповідає їхнім інтересам та здібностям. Також це допомагає неприбутковим організаціям знаходити відповідних волонтерів, які можуть допомогти їм виконувати їхні місії.

VolunteerMatch також є корисною для тих, хто хоче зайнятися волонтерською діяльністю та зробити позитивний внесок у свій спільноту та світ. Це може допомогти відчути себе корисним, знайти нових друзів та розвиватися в процесі надання допомоги.

Узагалі, VolunteerMatch є потужним інструментом для пошуку волонтерських можливостей та збільшення відчуття корисності та зацікавленості в наданні допомоги. Це дозволяє організаціям та волонтерам знаходити один одного та виконувати важливу місію для спільної користі.

# 1.2.4 GlobalGiving

GlobalGiving - це онлайн-платформа для збору коштів, яка забезпечує зв'язок між неприбутковими організаціями та людьми, які бажають надати допомогу. Заснована в 2002 році, GlobalGiving дозволяє організаціям розміщувати свої проекти на платформі та збирати кошти від людей, які хочуть пожертвувати на підтримку цих проектів.



Рис.1.2.4.1 Логотип GlobalGiving.

Організації можуть зареєструватися на платформі та створювати сторінки для своїх проектів, де вони можуть ділитися інформацією про свою місію, цілі та плани. Люди з усього світу можуть переглядати проекти та пожертвувати на ті, які вони хочуть підтримати.

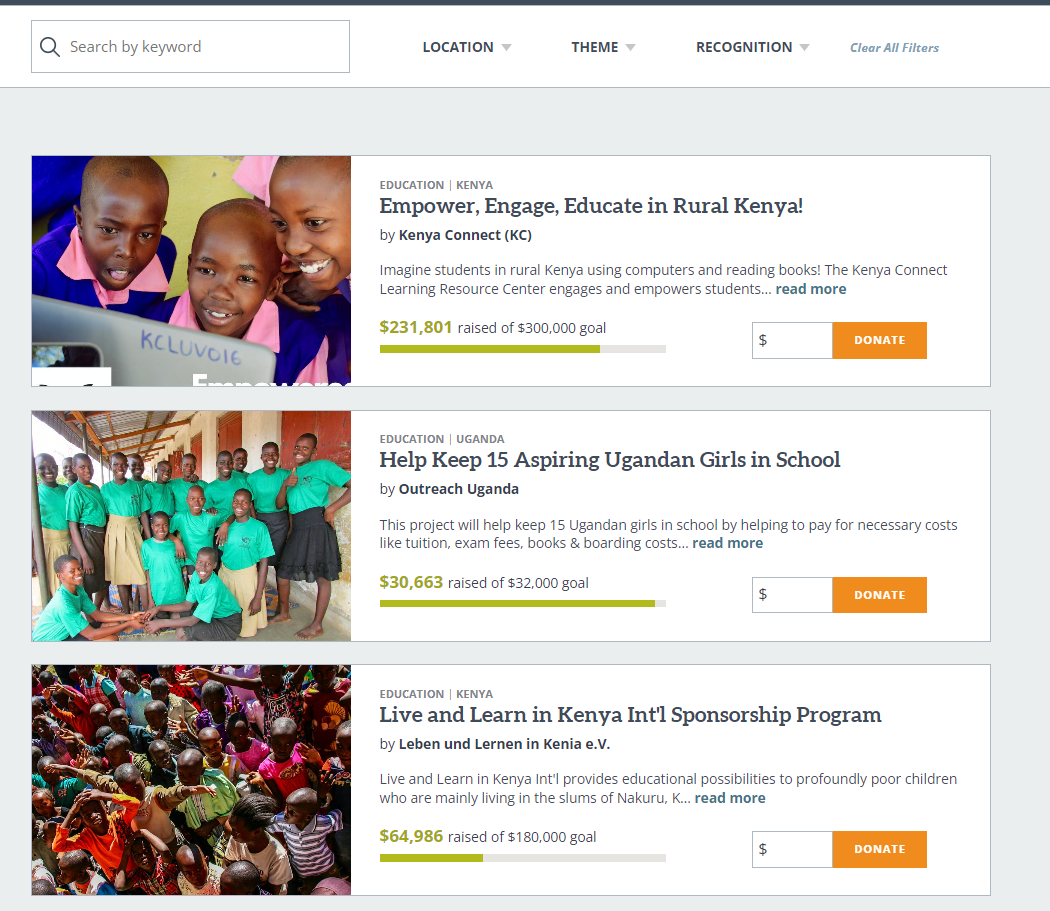


Рис.1.2.4.2 Інтерфейс GlobalGiving.

Одна з переваг GlobalGiving полягає в тому, що платформа пропонує можливості збору коштів для проектів у більш ніж 170 країнах світу. Крім того, організації можуть отримувати фінансову підтримку не тільки від індивідуальних спонсорів, але й від компаній та інших організацій, які бажають підтримати благодійні проекти.

Ще одна перевага GlobalGiving полягає в тому, що платформа дозволяє неприбутковим організаціям отримувати доступ до різних ресурсів та інструментів для розвитку та підвищення ефективності їхньої діяльності. Наприклад, організації можуть отримувати курси з фінансового менеджменту, маркетингу та фондової роботи.

# 1.2.5 Patreon

Patreon - це онлайн-платформа, яка допомагає творчим людям отримувати гроші від своїх шанувальників, щоб вони могли продовжувати свою роботу. На платформі є багато творців від музикантів, художників та блогерів до письменників та журналістів.



Рис.1.2.5.1 Логотип Patreon.

Платформа дозволяє творцям створювати різні рівні підписки, на які можуть підписатися їхні шанувальники. Кожен рівень підписки має свої вигоди та винагороди, які творець може пропонувати своїм підписникам, наприклад, ексклюзивний контент, ранній доступ до нових робіт або особистий доступ до творця.

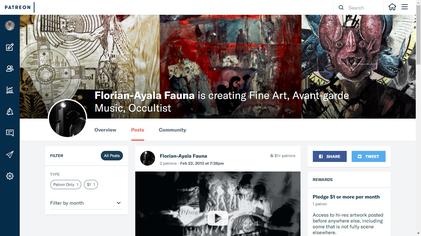


Рис.1.2.5.2 Інтерфейс Patreon.

Творці можуть встановлювати місячну плату за кожен рівень підписки або встановлювати мінімальний поріг для платежів від своїх шанувальників. Також є можливість зробити одноразовий внесок на розвиток творчості певного автора.

Patreon забезпечує творцям можливість заробляти на своїй роботі, зокрема на зйомках відео, виготовленні музики, написанні книг чи роботі над іншими проектами. Платформа дозволяє творцям зосередитися на творенні контенту, не витрачаючи час на пошук спонсорів та збір коштів.

Підтримка платформи Patreon надає творцям можливість мати стабільний прибуток та продовжувати свою творчість, що допомагає забезпечити стабільність та розвиток культурної та творчої сфери.

# 1.3 Збереження даних

При роботі з великими обсягами даних потрібно забезпечити їх структуризоване зберігання. Зберігання може бути локальним або віддаленим, на сервері, і для цього використовуються різні системи, такі як Block storage, File storage, Object storage та Key-value storage. Щоб вибрати оптимальну систему зберігання, потрібно провести тести та перевірити певні параметри в різних умовах.

Основні характеристики, які потрібно врахувати при виборі системи зберігання, включають:

• Надійність в роботі та відмовостійкість

• Швидкість відновлення даних в разі виникнення проблем

• Продуктивність, відповідна вашим запитам та потребам

• Консистентність даних для забезпечення їх коректності та цілісності.

**1.3.1 JSON**

JSON є текстовим форматом обміну даними між комп'ютерами, який може бути зрозумілим людиною і дозволяє описувати об'єкти та інші структури даних. Основне призначення JSON полягає в передачі структурованої інформації через мережу, завдяки процесу серіалізації. Формат був розроблений і популяризований Дугласом Крокфордом і головним чином використовується для написання веб-програм, зокрема технології AJAX. У порівнянні з XML, JSON займає менше місця, дозволяє складні структури в атрибутах та прямо інтерпретується за допомогою JavaScript в об'єкти. Це робить його більш придатним для серіалізації складних структур. JSON також підходить для зберігання складних динамічних структур в реляційних базах даних або файловому кеші. У контексті веб-застосунків JSON доречний для обміну даними між браузером і сервером (AJAX) або між серверами (програмні HTTP-інтерфейси).

# 1.3.2 База даних

База даних - це набір даних, які організовані відповідно до певної концепції, що описує їх характеристики та зв'язки між ними, і містить щонайменше одну область застосування (відповідно до стандарту ISO/IEC 2382:2015). Загалом, база даних включає схеми, таблиці, представлення, збережені процедури та інші об'єкти, і її дані організовані згідно з моделлю організації даних. Сучасні бази даних також можуть містити засоби для обробки даних та їх опис.

База даних (БД) - це структурована набір даних, що відображає стан об'єктів певної предметної області та зв'язки між ними. Система керування базами даних (СКБД) - це програма, яка забезпечує можливість створення БД та виконання операцій з даними, які в ній зберігаються. СКБД не є окремим видом програмного забезпечення, а лише документи, з якими оперують спеціальні прикладні програми.

В загальному випадку будь-який впорядкований набір даних може вважатися базою даних, наприклад, паперова картотека про працівників підприємства в відділі кадрів. Однак, ця стаття зосереджена на використанні баз даних в інформаційних системах. Застосунки для роботи з базами даних є дуже поширеними прикладними програмами, і сучасні інформаційні системи використовують СКБД для забезпечення роботи з базами даних. СКБД - це система, що заснована на програмних та технічних засобах, яка забезпечує визначення, створення, маніпулювання, контроль, керування

та збереженням баз даних.

# Висновок

Для розробки Платформи для благодійних пожертвувань та волонтерської діяльності перш за все необхідно провести аналіз існуючих додатків на ринку. Я розгянув найпопулярніші системи, які можливо знайти, використовуючи пошуковик Google. Всі системи мають свої переваги та неділоки, деякі платформи мають доволі застарілий інтерфейс що робить не дуже зручним їх використання, але на томість платформа Patreon має дуже хороший пророблений інтерфейс та хорошу вітальну сторінку. Всі перелічені платформи були створені в форматі веб-додатків, що є логічно так як це достатньо зручно для людини яка вирішить зробити пожертвування і самих авторів оголошень.

При реалізації проекту було вирішено базу даних. Я зупинився на сервері MSSQL. Для початку роботи з базою даних, її необхідно створити з новими таблицями та налаштувати зв’язки між ними.

# РОЗДІЛ 2. ВИКОРИСТАНІ ЗАСОБИ РОЗРОБКИ

Комп'ютери є універсальними знаряддями, які здатні здійснювати складні обчислення, полегшувати повсякденні завдання, автоматизувати нудні процеси та дозволяти легко обмінюватись інформацією, незалежно від місцезнаходження. Проте, комп'ютери не мають "інтелекту", як люди, що створює мовний бар'єр між користувачем та машинами. Для того, щоб комп'ютер виконав певну дію, необхідно подати набір інструкцій, створених людиною, оскільки сам комп'ютер не може нічого зробити самостійно.

# 2.1. ASP.NET

ASP.NET - це один з найпопулярніших фреймворків для створення веб-додатків. Він розробляється Microsoft і використовується для створення веб-сайтів, веб-додатків та веб-сервісів. ASP.NET базується на платформі .NET і використовує мови програмування C# або Visual Basic .NET для розробки веб-додатків.



*Рис.2.1 Логотип ASP.NET.*

ASP.NET має багато переваг, серед яких варто відзначити:

Швидкість та продуктивність - ASP.NET забезпечує високу продуктивність завдяки використанню принципу компіляції на етапі встановлення та підтримки кешування на сервері.

Безпека - ASP.NET забезпечує безпеку веб-додатків завдяки вбудованим механізмам автентифікації, авторизації та обробки помилок. Крім того, він має вбудований механізм захисту від атак Cross-Site Scripting (XSS) та Cross-Site Request Forgery (CSRF).

Масштабованість - ASP.NET забезпечує масштабованість веб-додатків завдяки використанню розподіленої архітектури та можливості використання кластерів серверів.

Легкість використання - ASP.NET має широкий вибір інструментів для розробки веб-додатків, включаючи інтегровану розробку середовища (IDE) Visual Studio, що дозволяє швидко створювати веб-додатки.

Підтримка - Microsoft надає широку підтримку ASP.NET, включаючи документацію, навчальні матеріали та форуми спільноти, що дозволяє розробникам швидко вирішувати проблеми.

ASP.NET дозволяє розробникам створювати веб-додатки різних типів, включаючи веб-сайти, веб-додатки та веб-сервіси. Для цього ASP.NET пропонує різні типи проектів, такі як ASP.NET Web Forms, ASP.NET MVC та ASP.NET Web API. Кожен з цих типів проектів має свої особливості та спрощує розробку веб-додатків певного типу.

ASP.NET Web Forms - це тип проекту, який базується на концепції сторінок, тобто кожна сторінка має своє представлення та код, що відповідає за її функціональність. ASP.NET Web Forms дозволяє швидко створювати веб-сайти та веб-додатки з використанням компонентів, таких як текстові поля, кнопки та інші.

ASP.NET MVC - це тип проекту, який базується на концепції розділення сторінки на моделі, представлення та контролери. Кожен з цих компонентів має свою функцію в процесі обробки запитів та відображення відповіді. ASP.NET MVC дозволяє швидко створювати веб-додатки з більш гнучким управлінням представленням та логікою додатку.

ASP.NET Web API - це тип проекту, який дозволяє створювати веб-сервіси з використанням технологій RESTful. Це дозволяє розробникам створювати веб-сервіси з використанням HTTP-протоколу та стандартних форматів даних, таких як JSON та XML.

Для розробки веб-додатків з використанням ASP.NET розробникам потрібно мати знання мови програмування C# або Visual Basic .NET та веб-технологій, таких як HTML, CSS та JavaScript. Крім того, розробникам слід мати знання про архітектуру веб-додатків та підходи до розробки з використанням ASP.NET.

ASP.NET - це потужний та популярний фреймворк для розробки веб-додатків, який забезпечує швидкість, безпечні та масштабовані веб-додатки. ASP.NET має вбудовану підтримку для багатьох інших технологій, таких як Entity Framework, SignalR, Razor та інших. Це дозволяє розробникам швидко створювати веб-додатки з використанням сучасних технологій та підходів.

Один з головних переваг ASP.NET полягає у тому, що він забезпечує високу продуктивність та ефективність роботи з базами даних. ASP.NET має вбудовану підтримку для баз даних, таких як Microsoft SQL Server, та може використовувати інші бази даних, такі як MySQL та Oracle.

Крім того, ASP.NET забезпечує високу безпеку веб-додатків, шляхом використання різних методів, таких як автентифікація та авторизація, валідація введення користувача та захист від атак типу Cross-Site Scripting (XSS) та SQL Injection.

Загалом, ASP.NET - це потужний та високопродуктивний фреймворк для розробки веб-додатків, який забезпечує розробникам широкі можливості для створення різноманітних веб-додатків та веб-сервісів. Будучи частиною платформи .NET, ASP.NET дозволяє розробникам використовувати різноманітні засоби, бібліотеки та сервіси для створення високоякісних веб-додатків.

# 2.2. Entity Framework

Entity Framework - це технологія об'єктно-орієнтованого доступу до даних (ORM), яка дозволяє розробникам працювати з базами даних, як з об'єктами у програмі. Це забезпечує високу продуктивність та ефективність роботи з базами даних та зменшує кількість коду, необхідного для створення та роботи з базами даних.



*Рис.2.2 Логотип EF Core.*

Entity Framework має вбудовану підтримку різних баз даних, включаючи Microsoft SQL Server, MySQL та PostgreSQL. Він дозволяє розробникам працювати з даними за допомогою LINQ (Language Integrated Query) - розширення мови C#, яке дозволяє виконувати запити до бази даних з використанням звичайних мовних конструкцій.

Entity Framework має кілька різновидів:

EF Core - це платформонезалежний, легкий та швидкий виконавчий код, який працює на Windows, Linux та Mac.

EF6 - це більш старіший варіант EF, який працює тільки на платформі .NET Framework та має менше можливостей, ніж EF Core.

Entity Framework дозволяє розробникам створювати різні типи запитів до баз даних, такі як вибірка, вставка, оновлення та видалення. Він також забезпечує можливість використовувати транзакції для збереження даних в базі даних та забезпечує підтримку багатокористувацької роботи з даними.

Entity Framework дозволяє використовувати підхід "Code First", коли розробники можуть створювати моделі даних на основі коду класів, а потім за допомогою міграцій змінювати структуру бази даних. Крім того, EF дозволяє використовувати підхід "Database First", коли розробники можуть створювати моделі даних на основі структури бази даних.

Узагальнюючи, Entity Framework, є потужним та універсальним інструментом для роботи з базами даних, який дозволяє розробникам швидко та ефективно працювати з даними у своїх програмах. Він дозволяє створювати складні запити до баз даних та використовувати міграції для зміни структури баз даних, що робить роботу з базами даних більш зручною та простою.

Entity Framework також підтримує підключення до баз даних з допомогою різних провайдерів, що дозволяє розробникам використовувати той провайдер, який найбільш підходить для їх потреб.

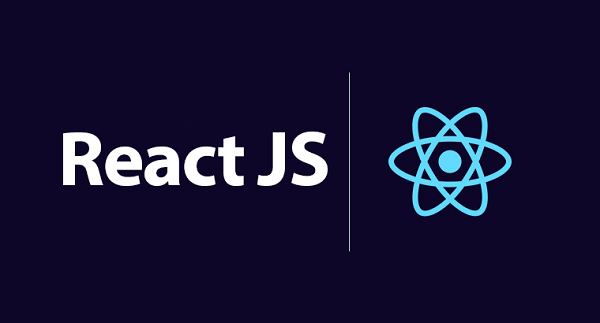
Однією з великих переваг Entity Framework є можливість використовувати його у складі ASP.NET, що дозволяє розробникам створювати високопродуктивні веб-додатки зі зручним доступом до баз даних.

Крім того, Entity Framework має декілька інструментів для покращення продуктивності, таких як кешування та пакетні операції. Вони дозволяють збільшити швидкість роботи з базами даних та зменшити кількість запитів до них, що позитивно впливає на продуктивність та ефективність додатка в цілому.

Загалом, Entity Framework є потужним та корисним інструментом для роботи з базами даних, який дозволяє розробникам швидко та легко створювати високопродуктивні та надійні програми. Він має багато можливостей та підтримує різні провайдери баз даних, що дозволяє розробникам використовувати його в різних проектах та у різних середовищах.

# 2.3. React.JS

React.js - це бібліотека для створення інтерфейсів користувача, що дозволяє розробникам створювати динамічні веб-сторінки та веб-додатки. Вона була розроблена Facebook в 2011 році і з тих пір стала дуже популярною у розробці фронтенду.



*Рис.2.3 Логотип React.JS.*

React.js працює з компонентами, які є самодостатніми частинами веб-сторінки. Кожен компонент містить HTML-код, CSS-стилі та JavaScript-логіку. За допомогою компонентів, розробники можуть створювати складні інтерфейси, розділяючи їх на простіші блоки.

Один з головних принципів React.js - це "один джерело правди" (single source of truth). Це означає, що стан компонента повинен бути збережений в одному місці, а не розподілений по всій програмі. Це забезпечує більш просте управління станом та запобігає появі непередбачуваного поведінки.

Ще одним важливим аспектом React.js є використання віртуального DOM (Document Object Model). Він є копією реального DOM, але не містить реальних елементів веб-сторінки. Віртуальний DOM зберігає стан всіх компонентів та забезпечує більш ефективну роботу з реальним DOM. Коли стан компонента змінюється, React.js змінює тільки необхідні елементи реального DOM, а не всю сторінку.

Ще одним важливим аспектом React.js є можливість використовувати JSX - синтаксис, що дозволяє вбудовувати JavaScript-код в HTML-код. Це забезпечує більш просту та зрозумілу розробку інтерфейсу.

React.js також підтримує широкий діапазон інструментів для розробки, включаючи багато корисних бібліотек, таких як Redux, React Router та Material-UI. Redux дозволяє зберігати стан додатка в одному місці, що спрощує його управління, React Router - це бібліотека для роутингу веб-додатків, що дозволяє змінювати вміст сторінки без перезавантаження сторінки, а Material-UI - це бібліотека для розробки стильних та сучасних інтерфейсів.

React.js також працює дуже добре з іншими технологіями, такими як Node.js та Express.js. Node.js дозволяє розробникам створювати серверну частину веб-додатків, а Express.js - це фреймворк для розробки серверних додатків на Node.js. За допомогою React.js та Node.js можна створювати повноцінні веб-додатки з різними можливостями, такими як авторизація користувачів, відправка електронної пошти, робота з базами даних та багато іншого.

React.js має також велику спільноту розробників, яка допомагає розвивати та підтримувати цю бібліотеку. У спільноті розробників React.js є багато корисних матеріалів для вивчення та розробки, таких як документація, курси, відеоуроки, блоги та інші ресурси.

У підсумку, React.js - це потужна технологія для розробки інтерфейсів користувача, яка дозволяє створювати динамічні та привабливі веб-додатки. Вона підтримує широкий діапазон інструментів та має велику спільноту розробників, що допомагає вивчати та розвивати цю технологію. Якщо ви шукаєте потужну технологію для розробки фронтенду, React.js може бути ідеальним вибором для вас.

# 2.4. Redux.JS

Redux.js - це бібліотека для керування станом додатка у React-додатках. Вона була розроблена, щоб спростити управління станом додатка та забезпечити однорідність в управлінні станом для різних компонентів React-додатків.



*Рис.2.4 Логотип Redux.JS.*

Redux заснований на патерні Flux, який був розроблений в Facebook. Патерн Flux включає односторонню потік даних, де дії (actions) виконуються з компонентів React та пересилаються в диспетчер (dispatcher), який відповідає за розподіл цих дій до магазину (store). Магазин містить стан додатка та логіку, яка відповідає за зміну стану за допомогою редукторів (reducers), які виконуються на кожній дії та змінюють стан додатка відповідно до логіки, яка описана в них. Компоненти React можуть підписуватись на зміни стану магазину та оновлювати свій стан відповідно до змін.

Redux дозволяє зберігати стан додатка в одному місці, що спрощує його управління та підтримку. Також він дозволяє зберігати історію змін стану, що дозволяє повернутись до будь-якої попередньої версії стану додатка. Це може бути корисно для налагодження та відлагодження додатка.

Redux також дозволяє розбити стан додатка на окремі частини та зберігати їх в окремих редукторах. Це дозволяє спростити логіку додатка та підтримку коду. Крім того, Redux дозволяє розробникам використовувати бібліотеки middleware для додавання додаткової логіки до процесу зміни стану додатка.

Redux має велику спільноту розробників, що допомага є вирішувати проблеми та надавати підтримку для розробників, що робить його ще більш привабливим для використання.

Redux може бути використаний для будь-якого типу додатка, від простих до складних, забезпечуючи однорідний підхід до управління станом додатка. Зокрема, Redux дозволяє писати тестований та декларативний код, що забезпечує більшу надійність та стабільність додатку.

Іншою корисною можливістю Redux є можливість розширення за допомогою плагінів та middleware. Middleware є функціями, які дозволяють додавати додаткову логіку до процесу зміни стану додатка, наприклад, логування або зберігання історії змін. Це дозволяє розробникам зберігати чистоту та простоту коду додатка, знаючи, що додаткова логіка може бути додана у будь-який час.

Redux також інтегрується добре з React-додатками, що дозволяє розробникам зберігати стан додатка та логіку в Redux, а компоненти React можуть підписуватись на зміни стану та оновлювати свій стан відповідно до змін. Це дозволяє розробникам використовувати привабливі функції React, такі як контекст та власники, та використовувати Redux для управління станом.

Redux є потужною бібліотекою для управління станом додатка в React-додатках. Він забезпечує простий та однорідний підхід до управління станом, дозволяє зберігати стан додатка в одному місці та підтримує розширення за допомогою плагінів та middleware. Redux інтегрується добре з React-додатках та дозволяє розробникам використовувати функціональність React, щоб створювати привабливі та потужні додатки. Крім того, Redux забезпечує можливість писати тестований та декларативний код, що допомагає зберегти надійність та стабільність додатку.

Однак, на вибір Redux впливає складність додатка та його потреби в управлінні станом. Якщо додаток є простим та має небагато стану, можливо, не потрібно використовувати Redux. Однак, якщо додаток має складні взаємодії між компонентами та багато стану, то Redux може бути потужним інструментом для управління станом.

У будь-якому випадку, якщо ви використовуєте Redux, важливо мати на увазі, що він не є срібною кулею. Іноді розробники можуть піти занадто далеко з використанням Redux, створивши занадто багато додаткової логіки та middleware. Тому важливо зберігати баланс між простотою та потужністю, використовуючи Redux у доречних випадках та зберігаючи код додатку простим та легким для розуміння.

Загалом, Redux є потужним інструментом для управління станом додатка, який дозволяє зберігати стан в одному місці та забезпечує декларативний та тестований код. Він інтегрується добре з React-додатками та дозволяє розробникам створювати привабливі та потужні додатки. Однак, використовуйте Redux розумно, зберігаючи баланс між простотою та потужністю, щоб забезпечити стабільність та надійність додатку.

# 2.5. Visual Studio

Набір інструментів Microsoft Visual Studio призначений для створення програмного забезпечення, включаючи планування, розробку інтерфейсу користувача, написання коду, тестування, налагодження, аналіз якості коду та продуктивності, розгортання у клієнтських середовищах та збір телеметрії щодо використання. Всі ці інструменти можна знайти в одному інтегрованому середовищі розробки (IDE) - Visual Studio. Вони були створені з метою максимально ефективної спільної роботи.



*Рис. 2.5 Логотип Visual Studio*

Visual Studio є універсальним інструментарієм для розробки різних типів додатків, від простих мобільних додатків і ігор до складних підприємств та центрів обробки даних. Наприклад, ви можете створювати додатки та ігри для Windows, Android та iOS, веб-сайти та веб-служби на основі платформ ASP.NET, JQuery та AngularJS, додатки для різних платформ та пристроїв, таких як Office, Sharepoint, Hololens, Kinect та "Інтернет речей", а також ігри та графічні додатки для Windows пристроїв, включаючи Xbox, з підтримкою DirectX.

Visual Studio за замовчуванням підтримує C#, C, C++, JavaScript, F# та Visual Basic, і легко інтегрується зі сторонніми додатками, такими як Unity та Apache Cordova, за допомогою розширень, таких як Набір засобів Visual Studio для Unity та Інструменти Visual Studio для Apache Cordova відповідно. Ви можете самостійно розширити Visual Studio, створивши власні інструменти для виконання спеціалізованих завдань.

**2.6. Git**

Git - це система контролю версій, яка дозволяє розробникам зберігати та керувати змінами в програмному коді та інших файлах. Вона стала незамінною частиною сучасного програмного розроблення та дозволяє командам розробників ефективно співпрацювати, зберігати історію змін, відслідковувати помилки та швидко відновлювати стан додатків.



*Рис.2.6 Логотип Git.*

Git дозволяє зберігати код у віддаленому репозиторії, який можна синхронізувати з локальними копіями. Кожен розробник може створювати свою власну гілку (branch), що дозволяє працювати зі своїми власними змінами, не впливаючи на головний код. Після завершення роботи, зміни можна об'єднати (merge) з головною гілкою, забезпечуючи збереження коду в одному місці та уникнення конфліктів.

Однією з найважливіших функцій Git є можливість повернення до попередніх версій коду. Якщо під час розробки виникають проблеми, Git дозволяє відновити попередню версію коду та продовжити роботу з цієї точки. Крім того, Git дозволяє співпрацювати з іншими розробниками, які можуть вносити зміни до коду та вирішувати проблеми шляхом взаємодії та обговорення.

Git також має багато інших корисних функцій, таких як відслідковування змін та дозвіл на доступ до репозиторію. Він інтегрується з багатьма редакторами коду та іншими інструментами, що дозволяє розробникам легко і ефективно працювати з Git.

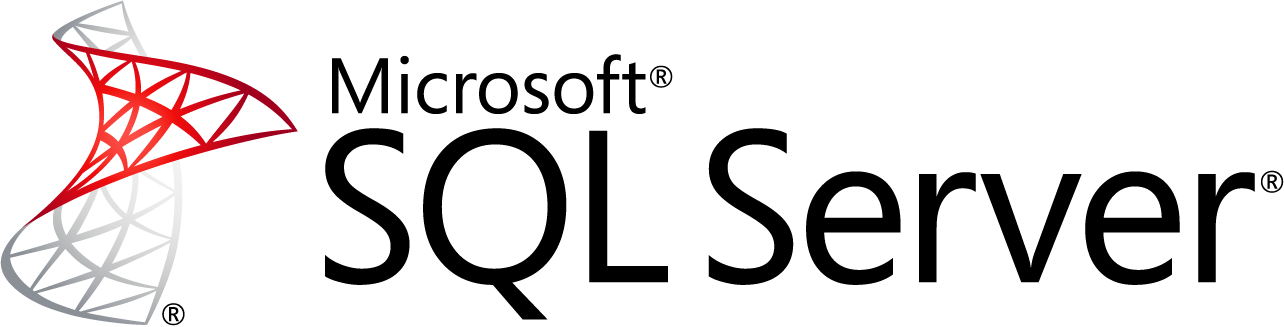
Хоча Git може здатися складним на початку, він має дуже добре документовану та розгалужену систему допомоги, що дозволяє вирішувати будь-які проблеми, які можуть виникнути. Багато розробників вважають Git незамінною частиною своїх робочих процесів та рекомендують його для будь-яких проектів.

Одним з найбільших переваг Git є те, що він є відкритою системою, що означає, що ви можете безкоштовно використовувати його та вносити свої власні зміни в систему. Багато відомих компаній, таких як Google, Facebook, Microsoft та інші, використовують Git для своїх проектів та активно співпрацюють зі спільнотою розробників, що дозволяє їм розвивати цю систему контролю версій та покращувати її функціональні можливості.

Загалом, Git є дуже корисним інструментом для будь-якої команди розробників, яка працює над програмними проектами. Він дозволяє розробникам зберігати та керувати змінами в коді, співпрацювати з іншими розробниками, відслідковувати помилки та швидко відновлювати стан додатків. Хоча Git може здатися складним на початку, він має дуже потужну та детально документовану систему, яка дозволяє вирішувати будь-які проблеми, які можуть виникнути. Тому використання Git є необхідним для будь-якого проекту програмного розроблення та допомагає зберігати якість коду та робочий процес команди на високому рівні.

# 2.7. SQL мова

SQL (Structured Query Language) - це мова, яка дозволяє створювати та працювати з реляційними базами даних, які є наборами пов'язаної інформації, збереженої у таблицях.



*Рис. 2.7 Microsoft SQL Server*

Завдяки постійному зростанню інформаційного простору, з'явилась необхідність у створенні стандартної мови, яка могла б бути використана в різних типах комп'ютерних середовищ. Стандартна мова дає можливість користувачам, які знають один набір команд, використовувати їх для створення, пошуку, зміни та передачі інформації, незалежно від типу комп'ютера.

У нашому все більш і більш взаємопов'язаному комп'ютерному світі, користувач, який володіє цією мовою, має значну перевагу у використанні та узагальненні інформації з різних джерел.

SQL є основною стандартною мовою технології реляційних баз даних завдяки своїй елегантності та незалежності від специфіки комп'ютерних технологій. Стандарт SQL визначається Американським Національним Інститутом Стандартів (ANSI) і Міжнародною Організацією по Стандартизації (ISO).

Більшість комерційних програм баз даних розширюють SQL, додаючи різні особливості в цю мову, які вони вважають корисними. Іноді ці особливості порушують стандарт мови, але деякі з них стають стандартами "ринку" через свою корисність.

Microsoft SQL Server є високопродуктивною платформою для обробки та аналізу даних будь-якого розміру, яка відповідає всім сучасним вимогам роботи з даними будь якої складності. SQL Server має широкий спектр функцій та інструментів, що дозволяють ефективно управляти даними, виконувати запити, забезпечувати безпеку та доступність даних.

За допомогою SQL Server можна створювати реляційні бази даних, виконувати транзакції, встановлювати зв'язки між таблицями, індексувати дані для швидкого пошуку та виконувати різноманітні запити. Також платформа підтримує роботу з геоданими, що дозволяє створювати геопросторові додатки та аналізувати дані за географічними параметрами.

SQL Server надає зручні інструменти для резервного копіювання та відновлення даних, а також має вбудовану систему безпеки, яка дозволяє керувати доступом до даних та забезпечувати їх захист від несанкціонованого доступу. Платформа також має вбудовану підтримку реплікації, що дозволяє забезпечити доступність даних в режимі реального часу.

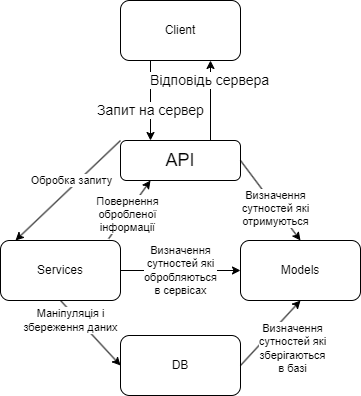
Окрім того, SQL Server підтримує інтеграцію з різноманітними додатками, включаючи різні сервіси та інші бази даних. SQL Server можна використовувати для розробки та реалізації багатошарових додатків, які працюють з великим обсягом даних та мають складну структуру. Це робить SQL Server потужним інструментом для розробки підприємницьких додатків та систем управління даними будь-якого розміру та складності.

# РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ

# 3.1. Розробка архітектури системи

Структура програми організована у вигляді сукупності незалежних блоків, що називаються модулями. Кожен модуль має свою структуру та поведінку, яка відповідає встановленим правилам, що сприяє спрощенню тестування та виявленню помилок. До того ж, завдання, що можуть виконуватись не лише в веб-додатку, можуть бути чітко відокремлені від інших завдань, що забезпечує більшу універсальність серверного ПЗ згідно правил REST API. Кожен модуль виконує свою ізольовану роль в застосунку, та має власне місце в проекті. Архітектура програми включає такі модулі, як модуль для управління БД, модуль API, модуль сервісів та модуль моделей.

Нижче можна побачити краще взаємозв’язок модулів, де клієнт взаємодіє з сервісами які обробляють дані клієнт через передачу за допомогою API модуля, який приймає вхідні запити на сервер. В свою чергу кожег серверний модуль пов’язаний з модулем моделей, який необхідний для того аби визначити всі сутності які будуть використовуватись в усіх модулях.



*Рис. 3.1 Діаграма взаємозв’язоку модулів системи*

# 3.2. Use-case діаграма Use case діаграма є одним із видів діаграм, що використовуються в області розробки програмного забезпечення для моделювання функціональної взаємодії між системою та її користувачами. Вона дозволяє описати поведінку системи на основі різних сценаріїв взаємодії між користувачами та системою, тобто вказати, які дії можуть бути виконані користувачами, які відповіді може повертати система та як вона може взаємодіяти з іншими системами або сервісами. Use case діаграма допомагає зрозуміти функціональні вимоги до системи та забезпечує основу для подальшого розроблення архітектури системи.

Зокрема use-case діаграма вимагає наведення самих так званих use-кейсів, тобто дій які зможуть здійснювати представлені в застосунку сутності. Нижче є перераховані use-кейси для всіх сутностей.  
 Користувач:

-Створення облікового запису  
 -Вхід в обліковий запис

-Редагувати обліковий запис

-Здійснити пожертвування

-Поповнити рахунок

-Подати заявку на волонтерський статус

Волонтер:

-Створити оголошення

-Редагувати оголошення

-Закрити оголошення

-Видалити оголошення

- Заповнювати і редагувати волонтерську сторінку

Адміністратор:  
 -Видалити користувача

-Скасувати для волонтера його статус

-Перевірити запит

Нижче наведена use-case діарама для нашого застосунку.



*Рис.3.2 Use-case діаграма*

Створивши use-case діаграму та перерахувавши use-cases, ми досягли кількох цілей. По-перше, ми з'ясували, які конкретно функції та можливості повинна мати наша система для забезпечення задоволення потреб користувачів. По-друге, ми отримали зрозуміння взаємодії між користувачами та системою, що дозволить нам розробляти кращі інтерфейси та взаємодію з системою. Нарешті, ми створили основу для подальшого розроблення архітектури системи, яка буде відповідати вимогам користувачів та забезпечувати ефективну та безперебійну роботу системи.

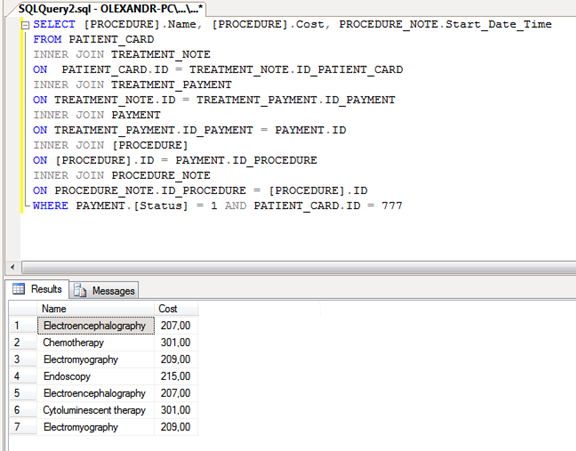
# 3.3. Розробка бази даних

Розробка бази даних це досить складна задача, так як від архітектури бази даних залежить продуктивність та доступність даних. Для початку потрібно розробити діаграму сутностей та зв’язків бази даних, так звану ER-діаграма.

Розроблена діаграма подана нижче:

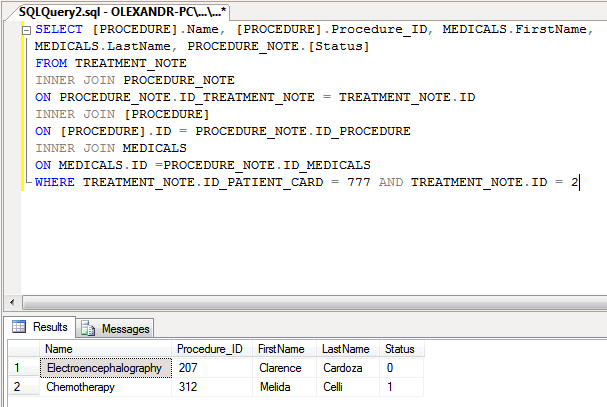
*D:\Temp\SS\OLEXANDR\UML\База даних Hospital-6.pngРис. 3.2 ER-діаграма**бази даних*

Після створення бази даних потрібно написати скрипти для отримання потрібних даних, для прикладу розглянемо отримання даних про оплачені процедури для пацієнта для прикладу, що має ID 777:

****

*Рис. 3.3 Отримані дані про оплачені процедури для пацієнта з ID 777*

Також для цього користувача ми отримаємо дані про історію лікування, а саме які процедури були виконані:

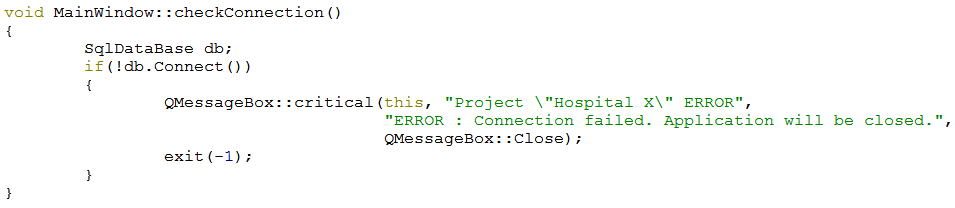


*Рис. 3.4 Отримані дані про історію лікування пацієнта з ID 777*

# 3.4. Розробка дизайну програми

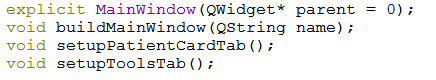
# 3.5. Розробка системи

Перед входом в систему потрібно перевірити чи програма може підключитись до бази даних, це робиться наступним чином:

 *Рис. 3.6 Метод перевірки з’єднання*

Якщо з’єднання не вдалось, програма показує повідомлення про помилку, та після його закриття, завершується з кодом помилки (-1).

В класі MainWindow проходить компонування усіх модулів. Це створення усіх вкладок та їх елементів, а саме в таких методах як:



*Рис. 3.7 Фрагмент заголовку для компонування*

Якщо користувач змінить вкладку, то викличеться підключений до сигналу зміни вкладки метод:



*Рис. 3.8 Прототип функції перемикання вкладок*

При зміні пацієнта випускається сигнал з його id - PatientSelected(const QString &id), що підключений на такі методи:

* connect(m\_patient\_select, &PatientSelect::PatientSelected, m\_calendar, &Callendar::on\_patient\_id\_changed);
* connect(m\_patient\_select, &PatientSelect::PatientSelected, m\_finance\_journal, &FinanceJournal::RefreshFinanceJournal);
* connect(m\_patient\_select, &PatientSelect::PatientSelected, m\_patient\_info, &PatientInfo::get\_patient\_info);
* connect(m\_patient\_select, &PatientSelect::PatientSelected, m\_treatment\_note\_tree, &TreatmentNoteTree::SlotUpdateWindow);
* connect(m\_patient\_select, &PatientSelect::PatientSelected, m\_add\_payment, &AddPayment::slotUpdatePatientId);

що викликає оновлення всіх модулів, що містять дані про пацієнта.

Також розроблено модуль для бази даних DB\_API, він складається з двох класів:

* class SqlDataBase;
* class DBSqlQuery;

Клас SqlDataBase призначений для встановлення з’єднання з сервером і має такий набір публічних методів як:

1. bool Connect(string server, string database, string login, string password);
2. bool Connect(string server, string database);
3. bool Connect();
4. bool Disconnect();
5. bool IsOpen() const;

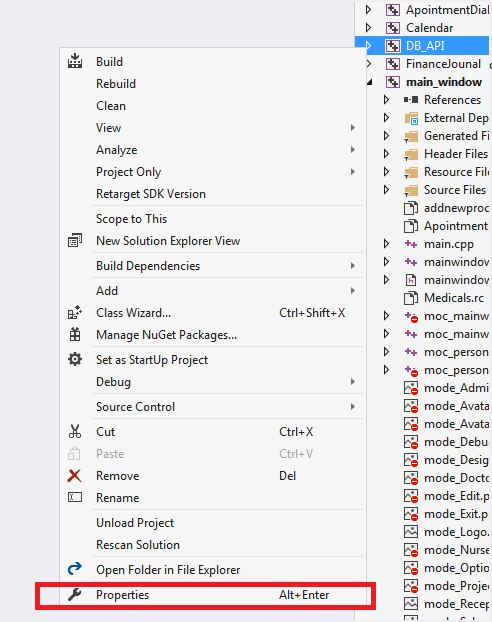
А клас DBSqlQueryвикористовується для виконання SQL запиту до сервера, і має такий набір публічних методів як:

* DBSqlQuery();
* DBSqlQuery(SqlDataBase& db);
* bool LinkWithDB(SqlDataBase& db);
* bool Exec(string query);
* void SetQueryString(string query);
* void BindValue(string placeholder, string val);
* bool Exec();
* bool Update(string table, string set\_columns, string set\_values, string where\_column, string sign, string s\_value);
* bool Update(string table, string set\_columns, string set\_values, string where\_condition);
* bool Select(string what\_columns, string from\_table, string where\_column, string sign, string s\_value);
* bool Select(string what\_columns, string from\_table, string where\_condition);
* bool Select(string what\_columns, string from\_table);
* bool Delete(string from\_table, string where\_column, string sign, string s\_value);
* bool Delete(string from\_table, string where\_condition);
* bool Insert(string into\_table, string values);//whole table
* bool Insert(string into\_table, string columns, string values);
* bool Next();
* string Value(int index);
* int get\_number\_of\_columns() const;

# 3.6. Збірка та розгортання

В даній системі кожен модуль є підпроектом, тому потрібно правильно налаштувати збірку всіх модулів та встановити правильну чергу збірки.

Для прикладу візьмемо модуль DB\_API **,** заходимо в Solution Explorer і вибираємо Properties для даного проекту:

****

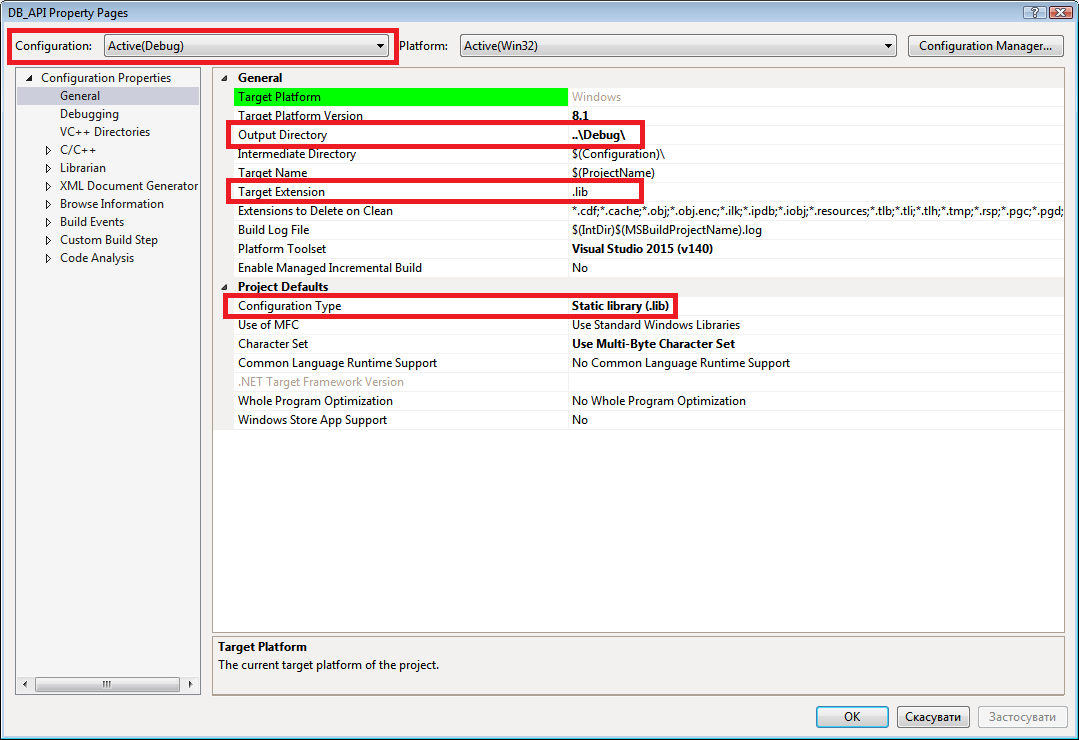
*Рис. 3.9 Налаштування проекту*

Далі потрібно для Configutation – Debug, в елементі списку General задати наступні параметри:

\* Вихідна тека (Output Directory) встановити значення ..\Debug\

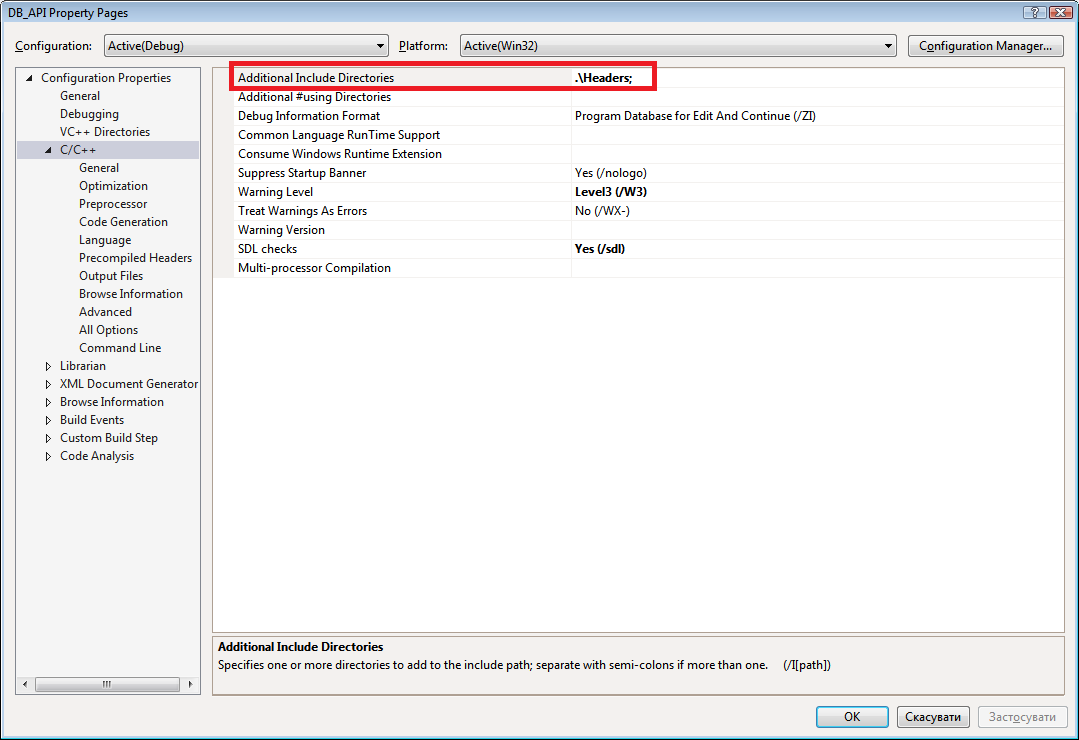
\* Розширення цілі (Target Extension) встановити .lib

\* Тип конфігурації (Сonfiguration Type) встановити Static library (.lib)



*Рис. 3.10 Загальні налаштування проекту*

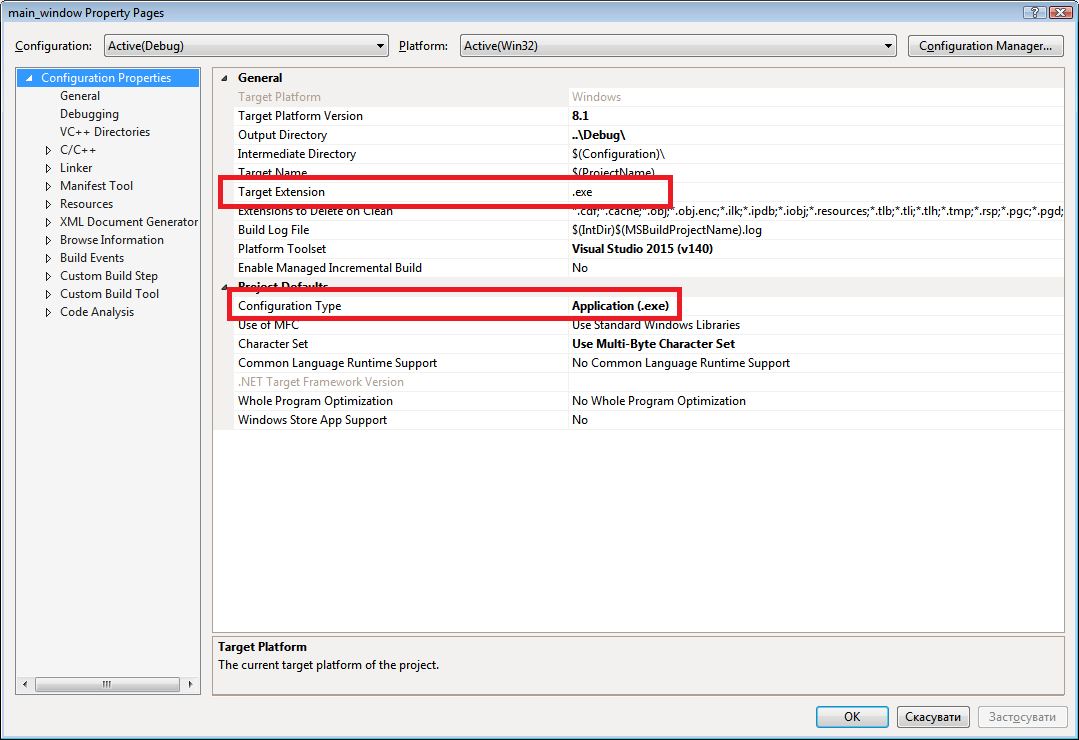
Наступним кроком буде налаштування додаткових включень бібліотек, так як DB\_API не використовує інші модулі системи, тут буде тільки власні бібліотеки:



*Рис. 3.11 Налаштування мови проекту*

Ці ж дії потрібно виконати і для Release пункту в Configuration меню. Але в вихідній теці потрібно встановити ..\Release\ значення.

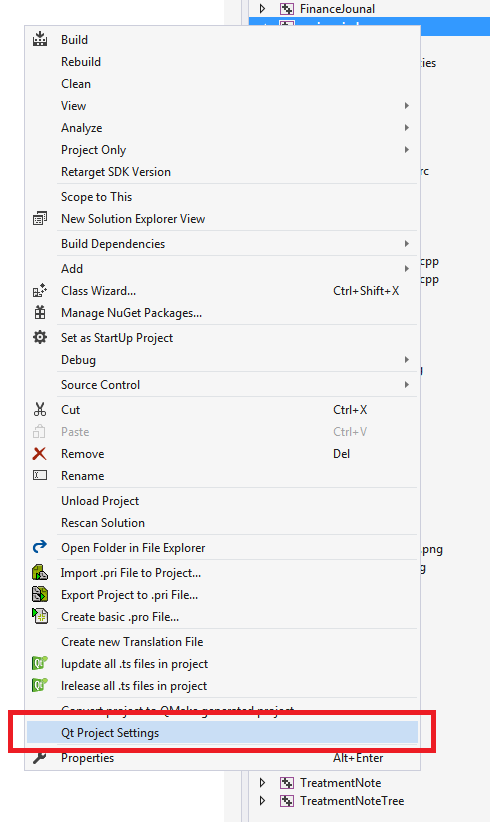
Даний приклад демонструє як зібрати модуль як статичну бібліотеку, також нижче буде розглянуто як зібрати основний модуль, як виконуваний файл. Для цього відкриваємо властивості проекту main\_window, як це було зроблено раніше. Так як даний проект містить функцію входу main(…) вихідний файл збірки буде в ролі виконуваного файлу на системі Windows з типом .exe.



*Рис. 3.12 Загальні налаштування основного проекту*

Також так як проект використовує Qt Framework потрібно зробити додаткові налаштування, для цього відриваємо меню для даного проекту

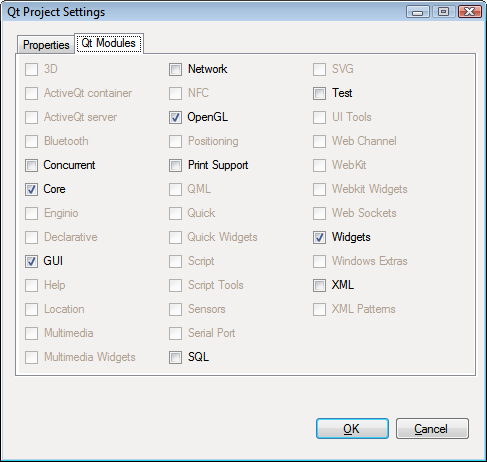
і вибираємо Qt Project Settings.

****

*Рис. 3.13 Загальні налаштування основного проекту*

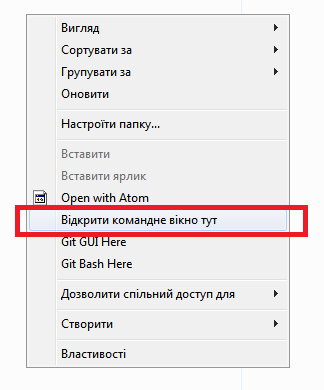
У вікні Qt Project Settings на вкладці Qt Modules вибираємо усі модулі, які були використані в даному проекті, а саме:

* Core
* GUI
* OpenGL
* Widgets



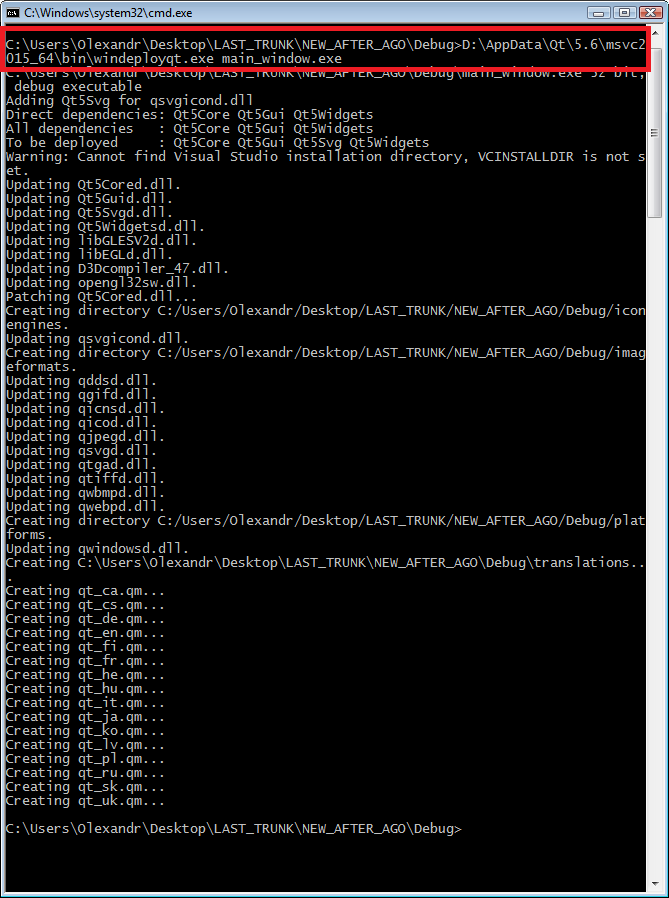
*Рис. 3.14 Вікно налаштувань “Qt Project Settings” для головного проекту*

Після успішної збірки потрібно розгорнути систему, для цього перейдемо в теку збірки, якщо це фінальна збірка , то тека повинна бути названа як Release. Відкриваємо для даної теки командну стрічку:



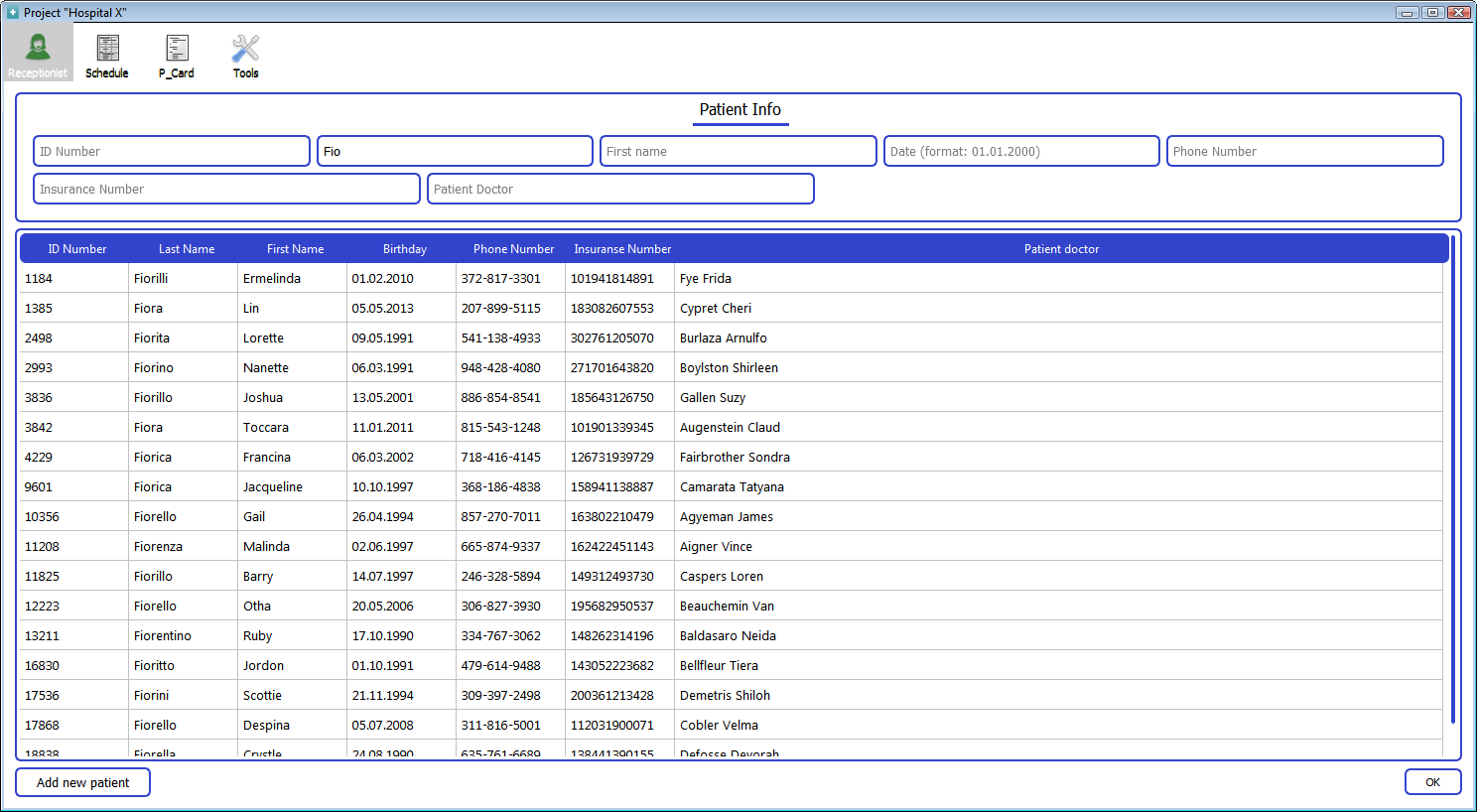
*Рис. 3.15Меню в “Explorer”*

Відкривши командну стрічку вводимо шлях для утиліти windeploy (знаходиться в теці Qt) і передаємо як агрумент назву виконуваного файлу:

****

*Рис. 3.16 Розгортання системи через командну стрічку*

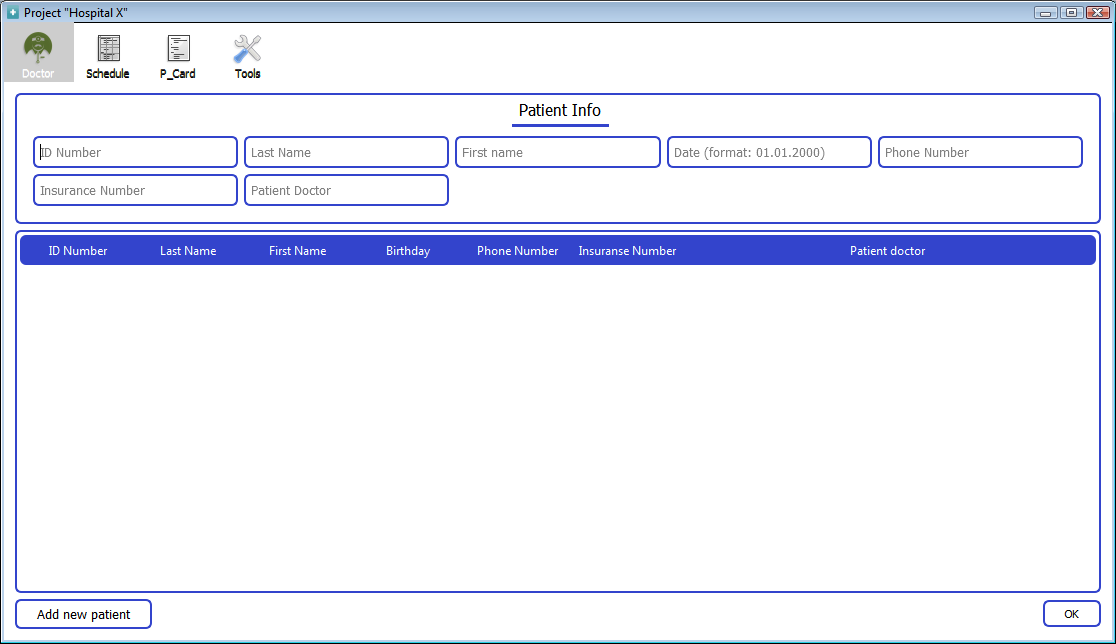
В результаті отримуємо повноцінний виконуваний файл, яким можна ділитись.Запускаємо програму для перевірки робочого стану:



*Рис. 3.17 Запуск системи з виконуваного файлу*

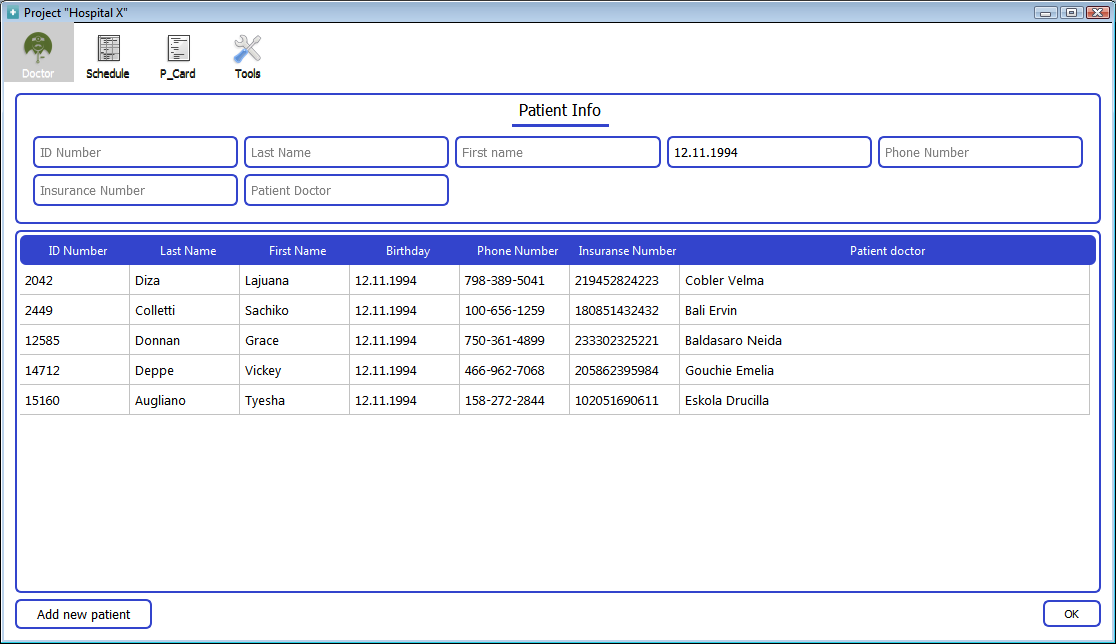
# 3.7. Основний функціонал системи

Після успішного запуску системи користувач має вибрати вкладу для роботи, для прикладу медичному працівнику потрібно знайти в базі пацієнта, він має вибрати першу вкладку:



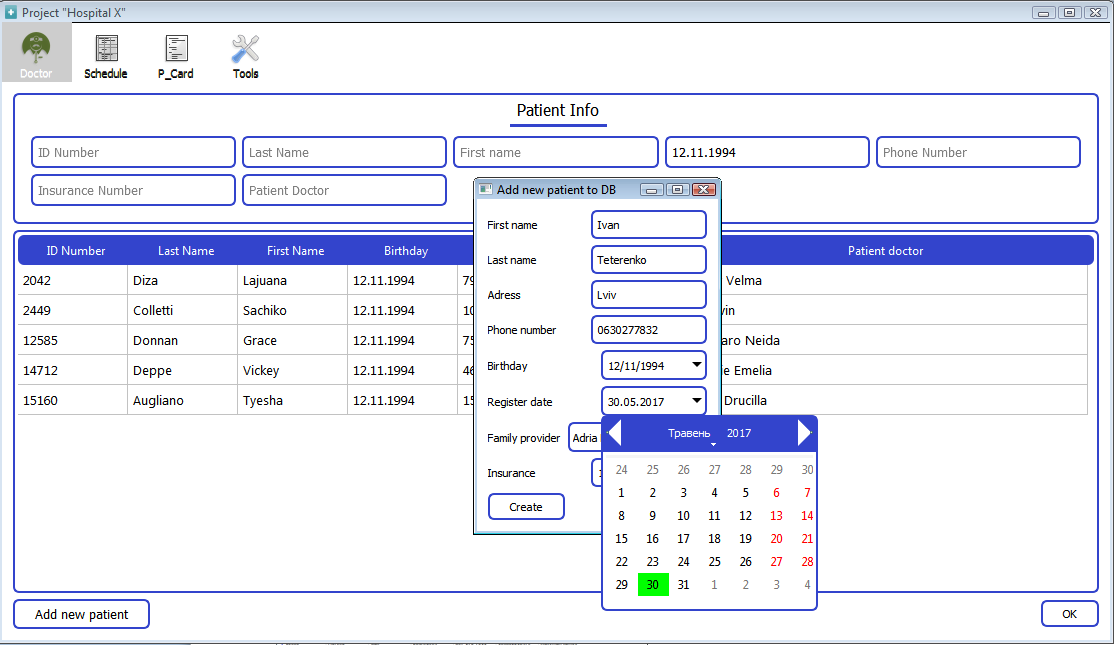
*Рис. 3.18 Перша вкладка системи «Пошук»*

Далі користувач шукає пацієнта по заданих даних, це може бути як ПІБ, так і телефон, дата народження або номер телефону. Для прикладу користувач почав пошук по даті 12.11.1994, і отримає результат:



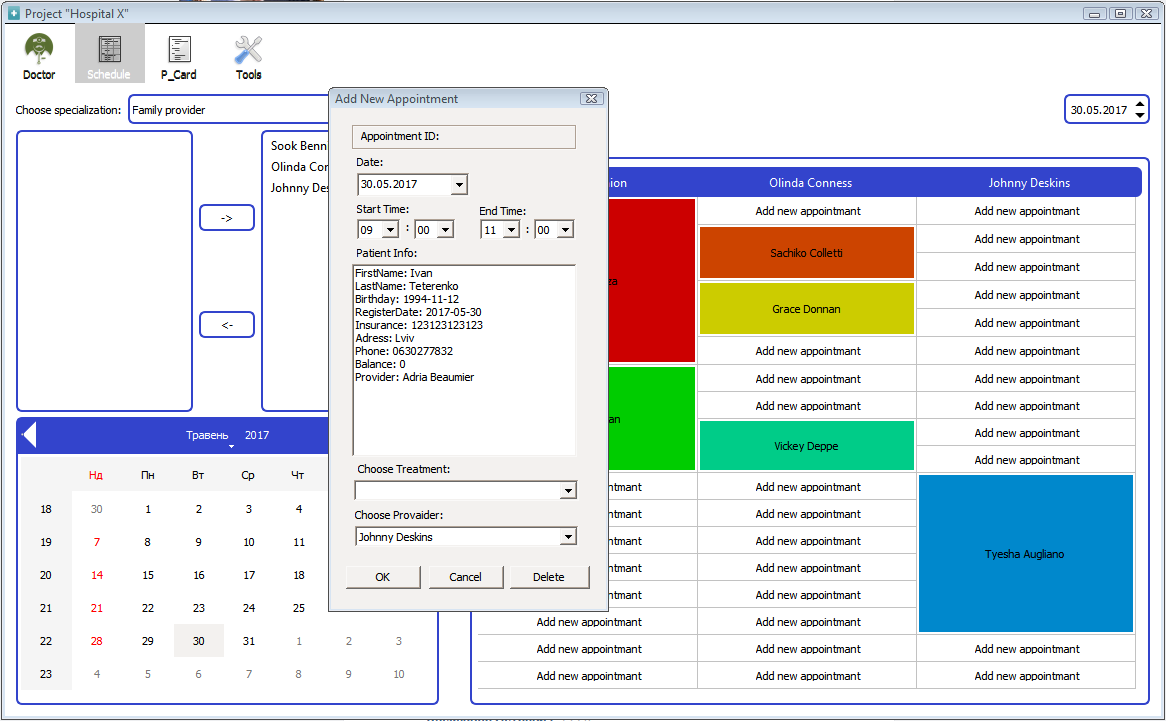
*Рис. 3.19 Результат пошуку по даті*

Але для прикладу пацієнт вказав, що його ім’я Іван Тетеренко і як можна побачити його в даній таблиці немає. Тому користувач системи має додати нового пацієнта за допомогою кнопки Add new patient, при успішному створенні буде виведено повідомлення про це, якщо ні, то буде виведено повідомлення про помилку:

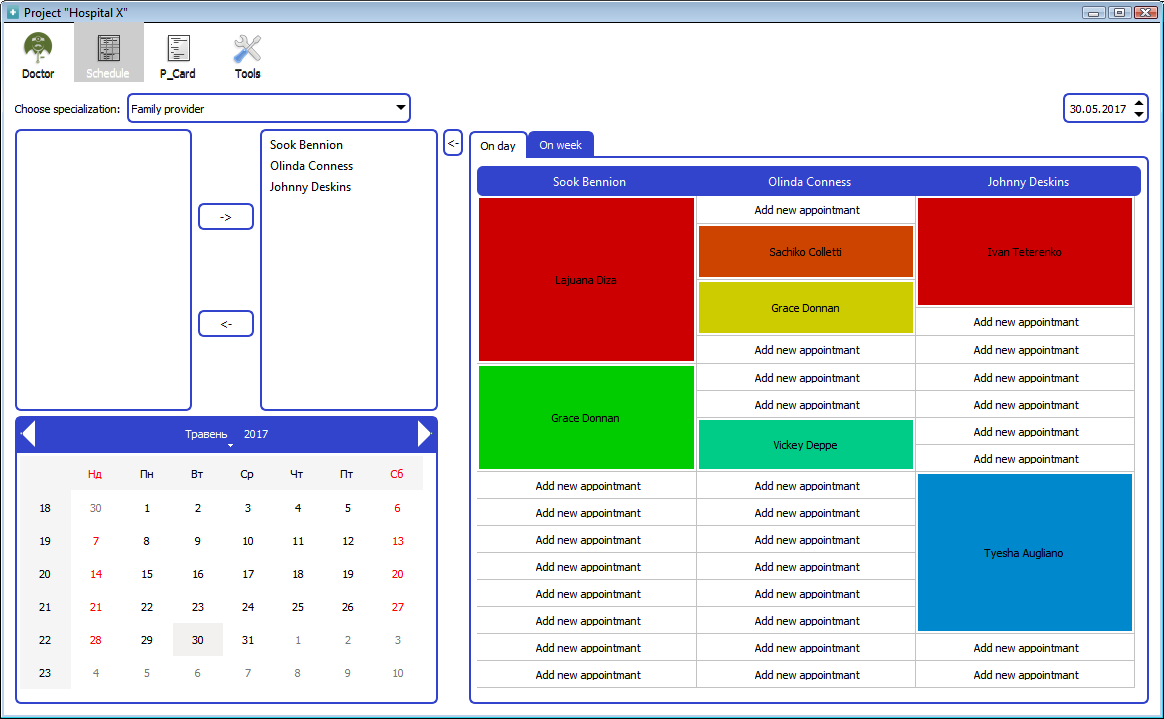


*Рис. 3.20 Створення нового пацієнта в системі*

Припустимо новому пацієнту, надалі Івану Тетеренку, потрібно відвідати сімейного лікаря, тому його потрібно призначити прийом до лікаря, це робиться на наступній вкладці:

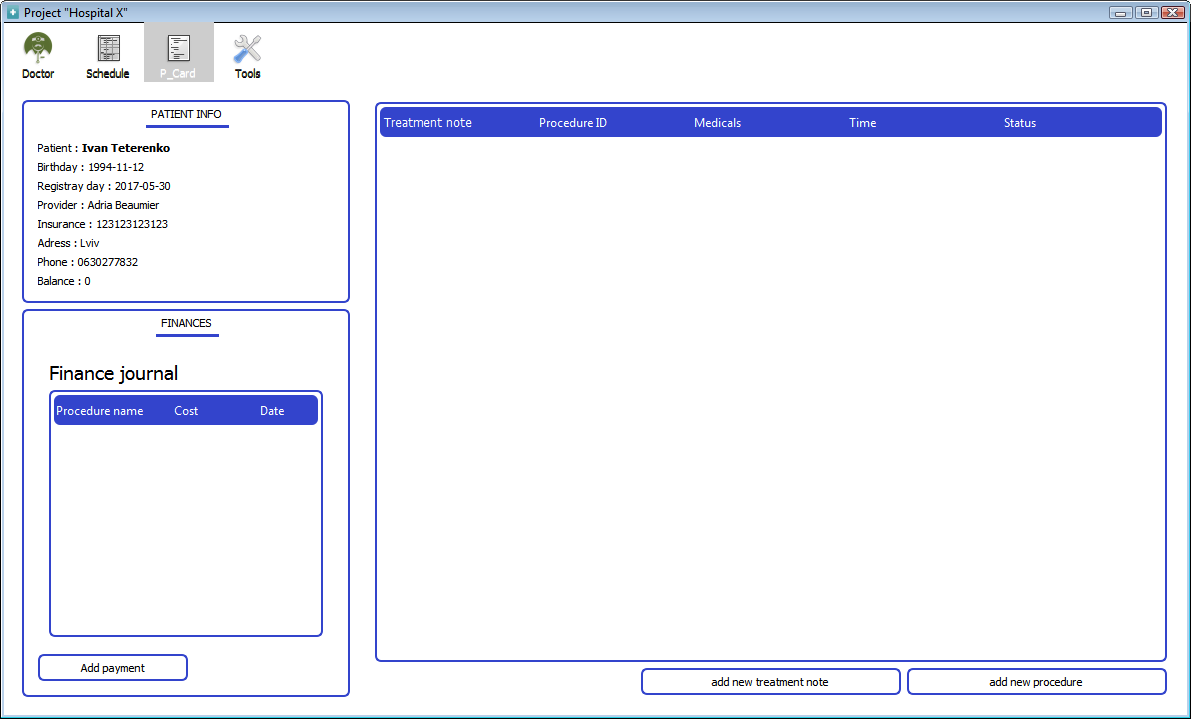


*Рис. 3.21 Призначення прийому*

Прийом призначено на дві години починаючи з 9 ранку на певного лікаря, який у ці години вільний. Також користувач системи може змінити властивості прийому, такі як початковий час, кінцевий час, дату та лікаря, що проводитиме прийом. 

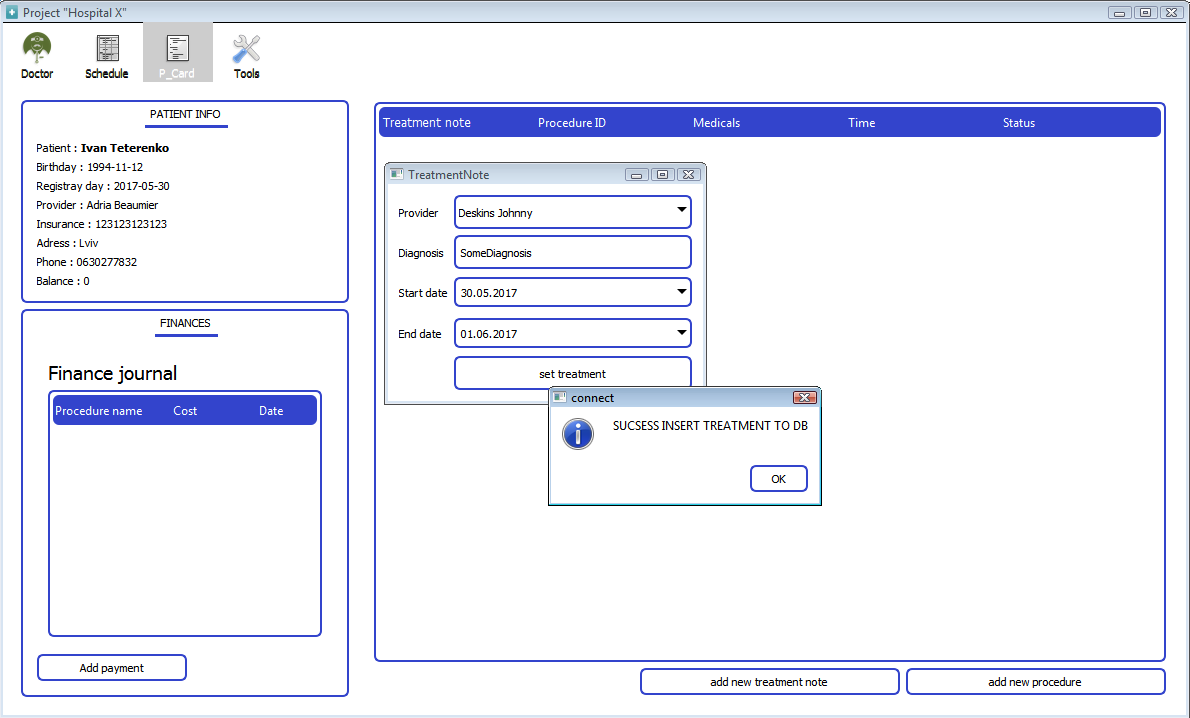
*Рис. 3.22 Призначені прийоми до лікаря*

Після цього під час прийому, користувач вибравши пацієнта, переходить на третю вкладку, де розміщена інформація про пацієнта. Для Івана Тетеренка ця інформація порожня, так як це новий пацієнт



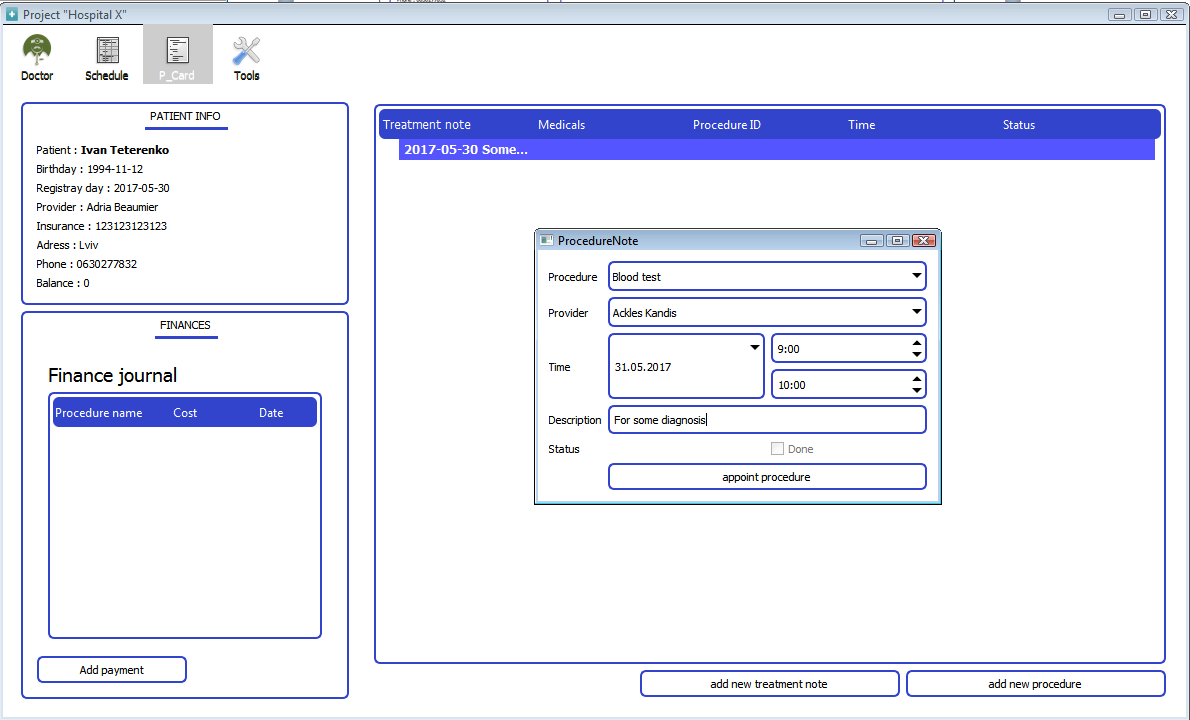
*Рис. 3.23 Сторінка пацієнта Івана Тетеренка*

Далі користувач створює запис лікування, використвуючи кнопку Add new treatment note:

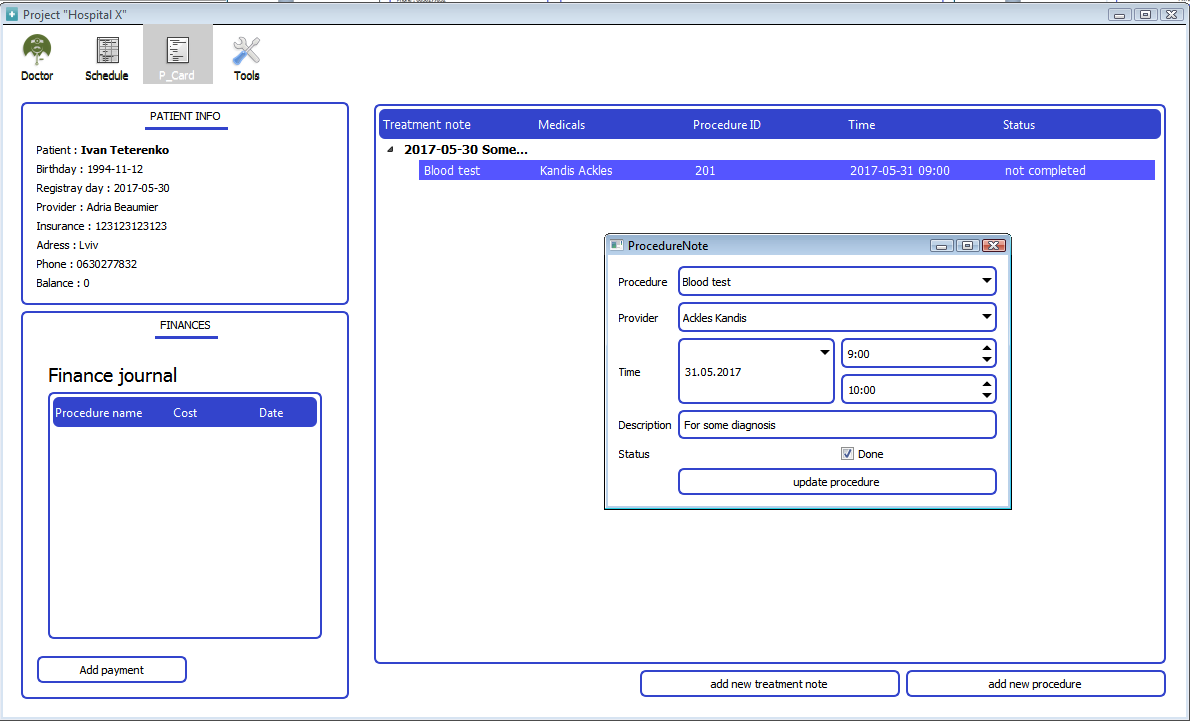


*Рис. 3.24 Створення запису про лікування*

Успішно створивши запис з діагнозом «Якийсь діагноз», користувач додає процедуру «Здача кров на аналіз»:

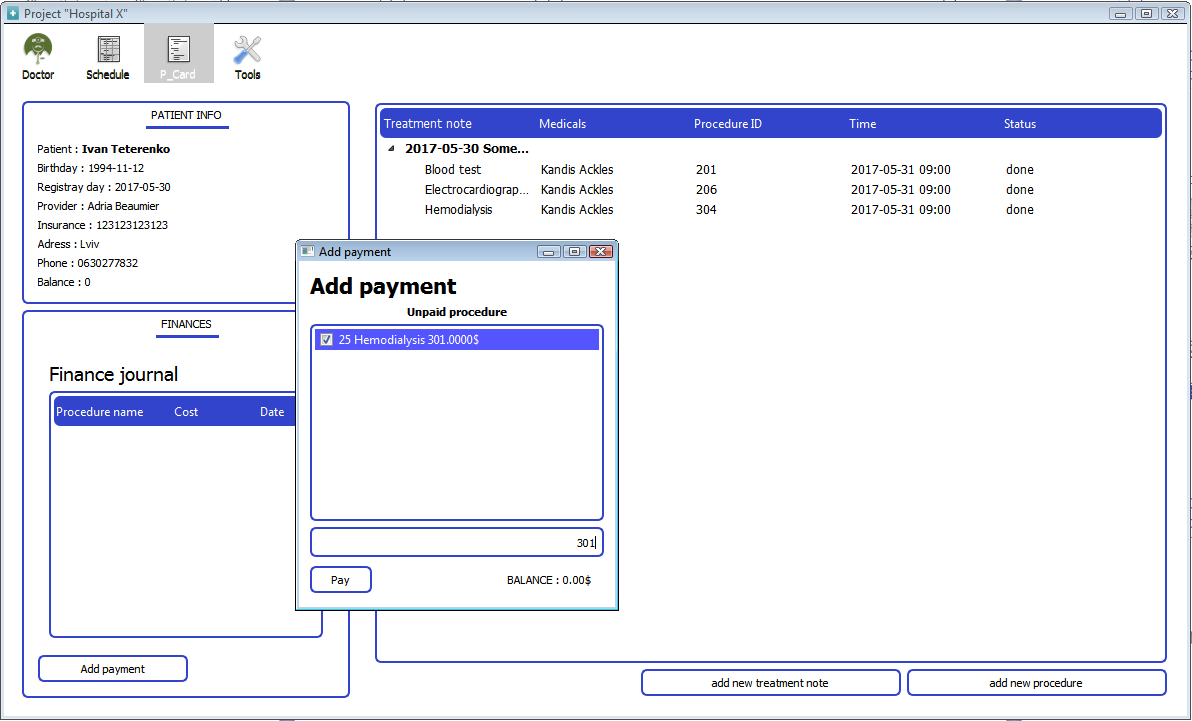


*Рис. 3.25 Створення процедури*

Ми бачимо що процедура доданого до цього лікування, і після її завершення ми можемо перевести її в статус «Виконано»: 

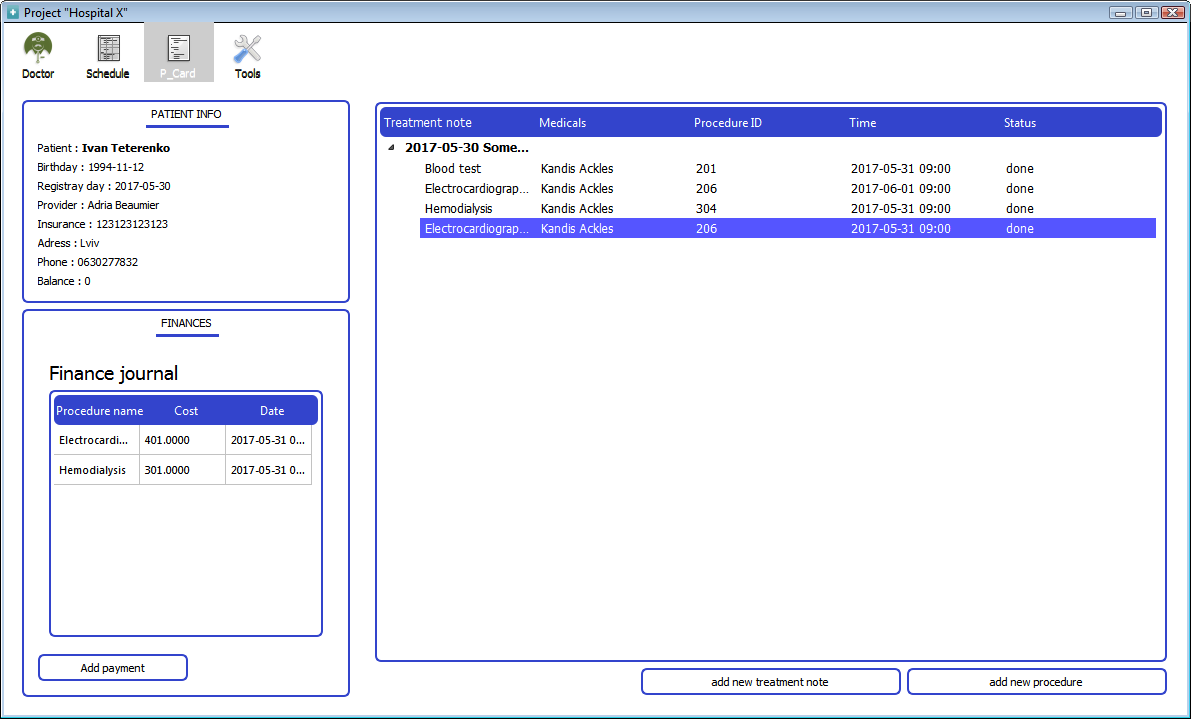
*Рис. 3.26 Налаштування процедури*

Зробивши кілька інших процедур пацієнт оплачує платні послуги, для прикладу йому зробили гемодіаліз і в вікні «Add payment», користувач системи позначає оплачені процедури:



*Рис. 3.27 Оплата процедури*

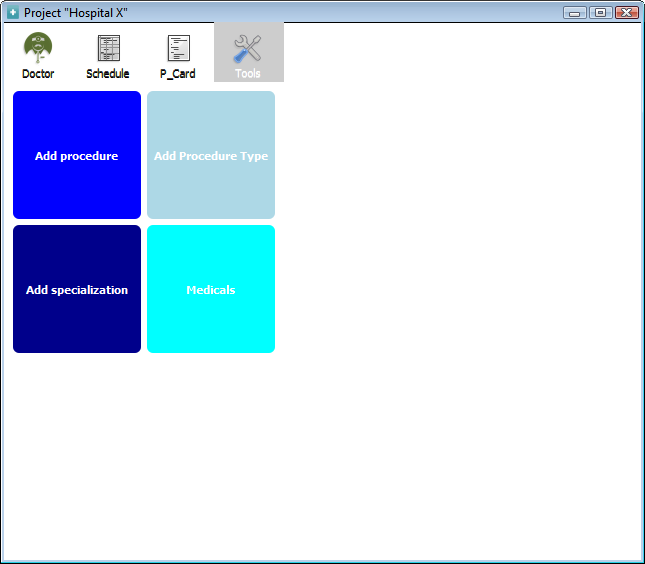
І в результаті в фінансовому журналі ми отримуємо список оплачених процедур:



*Рис. 3.28 Карта пацієнта з фінансовим журналом*

Та остання вкладка дозволяє користувачеві створювати нові:

* Процедури та їх типи
* Спеціалізації
* Медичний персонал



*Рис. 3.29 Вкладка «Засоби»*

# Висновок

При розробці даної системи виникли труднощі з лінкуванням бібліотек, а саме в порядку їх лінкування. Сама система відповідає повністю вимогам та побудована на модульній архітектурі. Усі модулі виконані, як одиниці, що функціонують з мінімальною прив’язкою до інших. Встановлення програми не викликає труднощів та не вимагає встановлення чи наявність будь-яких додаткових пакетів. Інтерфейс програми розроблений простим та не втомлюючим, не навантажуючи користувача великою кількістю функціоналу на виду.

# ВИСНОВОКИ

В результаті проектування, розробки та впровадження системи обліку пацієнтів було здобуто практичні навички в проектуванні інформаційних систем, в проектуванні баз даних, в розробці на мові С++ використовуючи Visual Studio i Qt Creator, в розробці SQL скриптів використовуючи SQL Management Studio і у впровадженні системи для кінцевих користвувачів.

Дана система розроблена для використання на стаціонарних комп’ютерах для персоналу в медичних закладах. Вона дозволяє вести облік пацієнтів, їх історії хвороб та лікування, призначати процедури та прийоми у потрібного лікаря. Розміщення візуальних компонентів має бути зручним для персоналу так, як розбите на певні категорії. Дизайн графічного інтерфейсу розроблений ненав’язливим та простим, використовуючи білий та відтінки синього кольори.

Система в порівнянні з конкурентами відрізняється, так як розроблена не як веб-орієнтований додаток, а як повноцінний десктопний. Тому основною перевагою буде зручність та легкість використання.

Сам дипломний проект за оцінками аналізу ринку є перспективним та вартим уваги , тому ресурси вкладені в нього будуть окуплені. Якщо продовжити розробку даної системи, то це спочатку буде підтримка проекту, саме виправлення критичних помилок, а далі розширення функціоналу, такі як перенесення на інші платформи, розробка мобільного додатку на базі ядра даної системи та покращення захисту системи.

# Використана література.

1. База даних — Вікіпедія [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki/База_даних>
2. Медична інформаційна система [Електронний ресурс] – Режим доступу:

<http://molfar.org/nashi-proekty/medychna-informatsiina-systema>

1. ELEKS — Вікіпедія [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki/ELEKS>
2. МІС «ЕМСіМЕД» - АЛТ Украина ЛТД [Електронний ресурс] – Режим доступу: <http://alt.ua/uk/equipment/1234-2>
3. Про продукт Meditex — МедЕксперт [Електронний ресурс] – Режим доступу: <http://www.medexpert.ua/ua/pro-produkte-meditex>
4. Про продукт Myrian — МедЕксперт [Електронний ресурс] – Режим доступу: <http://www.medexpert.ua/ua/pro-prodykt-myrian>
5. Myrian® – Intrasense [Електронний ресурс] – Режим доступу: <http://www.intrasense.fr/myrian/>
6. Про подукт Centrak — МедЕксперт [Електронний ресурс] – Режим доступу: http://medexpert.ua/ua/pro-produkt-centrak
7. XML — Вікіпедія [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki/XML>
8. JSON — Вікіпедія [Електронний ресурс] – Режим доступу: <https://uk.wikipedia.org/wiki/JSON>
9. Двійковий файл — Вікіпедія [Електронний ресурс] – Режим доступу: [https://uk.wikipedia.org/wiki/Двійковий\_файл](https://uk.wikipedia.org/wiki/%D0%94%D0%B2%D1%96%D0%B9%D0%BA%D0%BE%D0%B2%D0%B8%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB)
10. Основи бази даних / Укл. І.О. Завадський. – Львів, 2011. – 193с.
11. Бази даних у навчальному процесі / Укл. Н.В. Морзе. – Київ, 2007.– 120 с.
12. cplusplus.com - The C++ Resources Network [Електронний ресурс] – Режим доступу: <http://www.cplusplus.com>
13. MSDN – сеть разработчиков Microsoft [Електронний ресурс] – Режим доступу: <https://msdn.microsoft.com>
14. Qt | Cross-platform software development for embedded & desktop [Електронний ресурс] – Режим доступу: <https://www.qt.io>
15. Qt Wiki [Електронний ресурс] – Режим доступу: <http://wiki.qt.io>
16. Wikibooks [Електронний ресурс] – Режим доступу: <http://wikibooks.org>
17. SQL Server—Data Platform | Microsoft [Електронний ресурс] – Режим доступу: <https://www.microsoft.com/en-us/sql-server>
18. SQL Tutorial [Електронний ресурс] – Режим доступу: <https://www.w3schools.com/sql/>
19. Методичні рекомендації щодо виконання економічної частини випускної бакалаврської роботидля студентів Інституту комп’ютерних наук та технологій напряму підготовки 6.051501 «Видавничо-поліграфічна справа» / Укл: Н.І. Подольчак, А.І. Ясінська, Н.П. Патрікі, В.Я. Оліховський. – Львів: Національний університет «Львівська політехніка», 2011. – 14 с.