**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА**

**Кафедра Систем автоматизованого проектування**

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

**До бакалаврської кваліфікаційної роботи на тему**

|  |
| --- |
| Система обліку пацієнтів |
|  |
|  |
|  |

|  |
| --- |
| КН-43, Линда О.П. |

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-43 |

|  |
| --- |
| бакалавр |

на кваліфікаційну роботу (проект) студента групи ОКР

|  |
| --- |
| Линди Олександра Петровича |

(прізвище, ім’я, по батькові)

|  |
| --- |
| Система обліку пацієнтів |

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**

Линда Олександр. Розробка програмного забезпечення для обробки інформації про пацієнтів. В результаті даної бакалаврської кваліфікаційної роботи було розроблено систему обліку пацієнтів, яка оперує даними на локальній або віддаленій базі даних. Після вдалого з’єднання з базою даних користувач може вибрати пацієнта або додати нового. Вибравши пацієнта, користувач може перейти на вкладку «Розклад», вибрати потрібного вільного лікаря та призначити прийом на зручний для пацієнта час. Під час прийому лікар може переглянути інформацію про пацієнта та його історію лікування, також лікарю доступна функція додавання нового лікування та самих процедур. 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**

Lynda Oleksandr. Software development for processing information about patients. As a result of this bachelor qualification system was developed accounting patient system, which operates according to a local or remote database. After a successful database connection user can select or add a new patient. Choosing the patient, the user can switch on the "Schedule", select and appoint the suitable free doctor in a convenient time for the patient. During the appointment, the doctor can view information about the patient's history and its treatment, the doctor also can add a new treatments and procedures. The interface is designed to be easy and not bothersome, and certainly convenient.

After the analysis system and the economic assessment to say that the product is fit its development and to invest the necessary resources.

**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

На даний час більшу частину паперової роботи стараються автоматизувати і комп'ютеризувати для зменшення витрат ресурсів та часу. Правильно розроблена система може замінити цілий набір спеціалістів. Дані системи стараються впроваджувати у всі сфери життя, це також стосується і медичних закладів. Зараз існує гостра проблема вистоювання у чергах, пошук даних пацієнта та компакте збереження та відображення даних загалом.

Для прикладу федеральні і державні органи влади, страхові компанії та інші великі медичні установи у США, в значній мірі сприяють прийняттю електронних медичних записів. Конгрес США стимулює використання електронних систем проти паперових документів в частині охорони здоров'я (до $ 44000 в рамках Medicare, або до $ 65000 протягом шести років в рамках Medicaid) і пенею (тобто зменшення відшкодування для лікарів, які не використовують електронну систему).

Одна з компаній дослідила, що їх електронна система медичної документації може підвищити загальну ефективність на 6% в рік, а щомісячна вартість системи компенсуються вартістю всього лише кілька «неважливих» тестів або госпіталізацій.

Рукописні паперові медичні записи можуть погано читатись, які можуть внести свій вклад в медичні помилки. Попередньо друковані форми, стандартизація скорочень і стандартів почерку мали б підвищити надійність паперових медичних записів. Електронні документи можуть допомогти в стандартизації форм, термінології і введення даних. Оцифровка форм полегшує збір даних по епідеміології і клінічних досліджень.

Також з хороших варіантів це є налаштування обміну записами між різними інформаційними системами для статистичної звітності в таких питаннях, як поліпшення якості, управління ресурсами і суспільної охорони здоров'я епіднагляду за інфекційними хворобами.

# РОЗДІЛ 1. СИСТЕМИ ОБЛІКУ ПАЦІЄНТІВ

# 1.1. Системи обліку пацієнтів. Поняття та види

Системи обліку пацієнтів – це програмне забезпечення, яке дозволяє вести облік інфомації про пацієнтів та розклад для кожного медичного працівника, використовуючи базу даних.

База даних – це сховище даних, які впорядковані за певними правилами.

Бази даних класифікують за різними критеріями. За моделлю організації даних розрізняють такі бази даних:[1]

* Ієрархічна. Ієрархічна база даних може бути представлена як дерево, що складається з об'єктів різних рівнів. Між об'єктами існують зв'язки типу «предок-нащадок». При цьому можлива ситуація, коли об'єкт не має нащадків або має їх декілька, тоді як у об'єкта-нащадка обов'язково тільки один предок.
* Мережна. Така база даних подібна до ієрархічної, за винятком того, що кожен об'єкт може мати більше одного предка.
* Реляційна. Реляційна база даних зберігає дані у вигляді таблиць. Найвживаніші СКБД використовують реляційну модель даних.
* Об'єктно-орієнтована. У базі даних цього виду дані оформляють у вигляді моделей об'єктів.

За розміщенням даних виділяють такі види баз: [1]

* Локальна, або централізована. Така база даних підтримується на одному комп'ютері.
* Розподілена. Частини такої бази даних розміщують на різних комп'ютерах мережі.

Такі системи можуть бути реалізовані для наступних платформ: [1]

* Веб-платформа
* Стаціонарна система
* Мобільний додаток
* Система для влаштованих пристроїв

# 1.2. Приклади існуючих систем обліку пацієнтів.

# 1.2.1. Медична інформаційна система

Медична Інформаційна Система (МІС) – проект безкоштовного «хмарного» програмного забезпечення для будь – якої клініки України.

МІС – це система медичного документообігу для лікувальних закладів та пацієнтів. [2]



*Рис 1.1 Логотип компанії Мольфар та продукту igiMED.*

МІС впроваджує хмарне рішення на безоплатній основі igiMED.

Вони розуміють важливість внесення, збереження та послідовність передачі медичної інформації. Сьогодні, в Україні ні держава, ні приватні структури не пропонують лікувальним закладам зрозумілу та безкоштовну Медичну Інформаційну Систему. Або ці рішення вартують від 100, 00$ за одне робоче місце, або не пропонується нічого. [2]



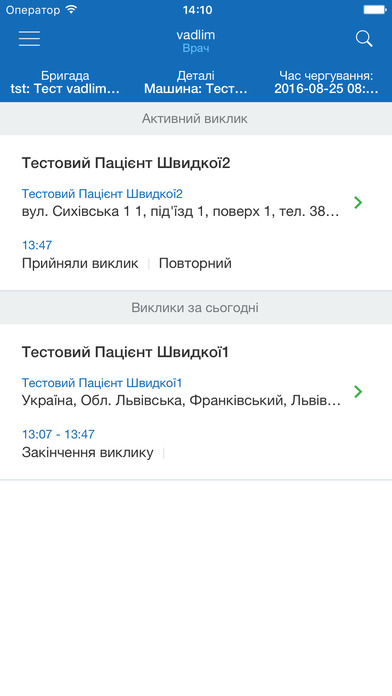
*Рис 1.2 Графічний інтерфейс продукту igiMED на OC Android.*

# 1.2.2 Медична система «Доктор Елекс»

Доктор ELEKS - медична інформаційна система, що дозволяє клінікам накопичувати, впорядковувати та зберігати усю необхідну інформацію про пацієнтів, включно із відео та графічними зображеннями у єдиній картотеці. [3]

Система включає в себе понад 20 підсистем, що дають змогу керувати усіма операційними процесами: від редагування шаблонів різних документів до аналізу ефективності роботи медичного закладу. Серед іншого Doctor ELEKS дозволяє індивідуально планувати робочий графік кожного спеціаліста і клініки загалом на різні проміжки часу. Водночас Doctor ELEKS інтегрований з різноманітними медичними приладами, такими як ультразвукові апарати, рентген-апарати, МРТ та КТ-сканери, ендоскопи, електрокардіографи та електроенцефалографи, що дозволяє "читати" зняті ними зображення. [3]

Наразі програму Doctor ELEKS впровадили та успішно використовують вже більше 60 клінік різного профілю в Україні та за кордоном (у тому числі у Польщі, Білорусі та Молдові). [3]



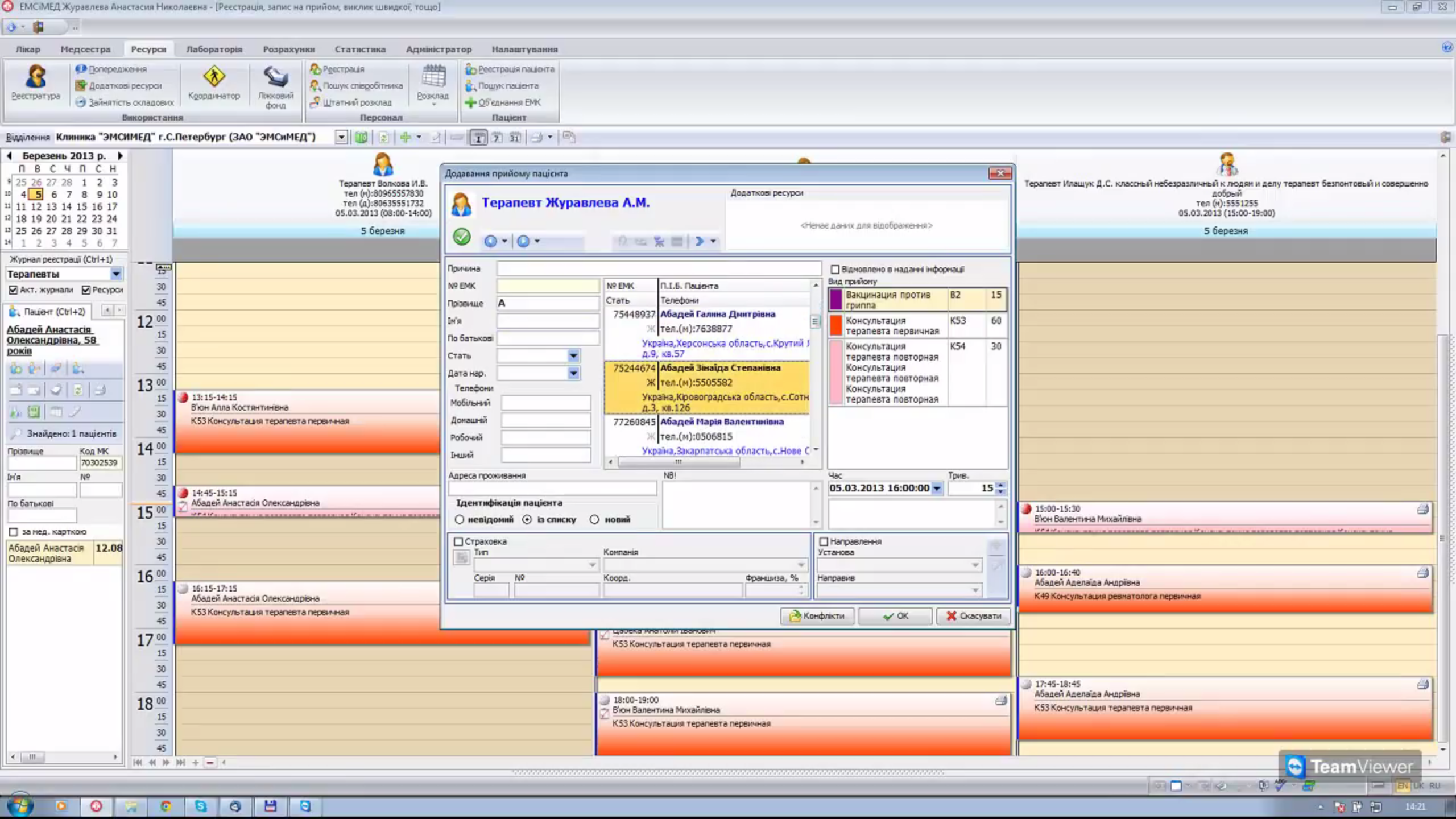
*Рис 1.3 Графічний інтерфейс продукту Доктор Елекс на OC IOS.*

# 1.2.3 МІС «ЕМСіМЕД»

Система «ЕМСіМЕД» – сучасна комплексна медична інформаційна система (далі – МІС), спрямована на максимальну оптимізацію та автоматизацію процесів діяльності лікувально-профілактичних закладів (далі – ЛПЗ) різних видів і форм власності. [4]

Головні завдання:

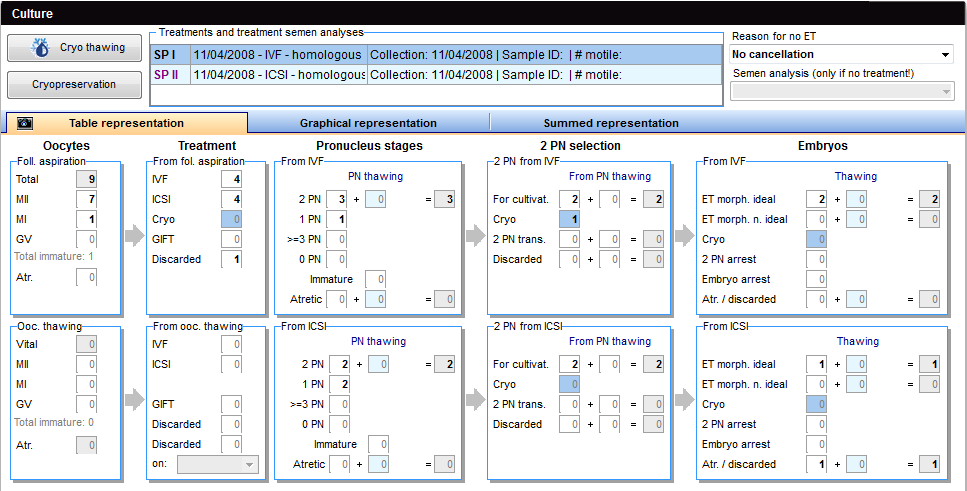
* Планування та оптимізація лікувального процесу для підвищення якості послуг, що надаються
* Інформаційний супровід і підтримка лікувально-діагностичних процесів
* Ефективне управління ресурсами ЛПЗ
* Інформаційне забезпечення роботи медичного персоналу
* Оперативне формування звітної інформації про проведення лікувального процесу
* Скорочення витрат на ведення адміністративно-господарського обліку та обліку фінансового характеру
* Взаємодія з іншими системами документообігу



*Рис 1.4 Графічний інтерфейс продукту ЕМСіМЕД на OC Windows.*

# 1.2.4 Система Meditex

Meditex - медична інформаційна система, яка розроблена спеціально для IVF клінік. Розробником цього програмного забезпечення є німецька компанія Critex. [5]



*Рис 1.5 Графічний інтерфейс продукту Meditex на OC Windows.*

Особливості системи :

* Моніторинг і управління повним циклом лікування безпліддя;
* Чітка структура циклу стимуляції;
* Автоматизовані інтерфейси звітності для різних реєстрів;
* Аналіз якості сперми і тестова лабораторія;
* Щоденне спостереження за кожним ооцитом;
* Контроль процесу кріоконсервації і управління контактами;
* Гнучкі звіти за допомогою статистичного аналізу;
* Миттєвий і легкий доступ до показників ефективності;
* Ідеальна інтеграція і обмін даними з медичним устаткуванням.

# 1.2.5 Програмне забезпечення Myrian

Програмне забезпечення Myrian - це мультимодальне рішення для перегляду медичних зображень КТ МРТ. Myrian - це робоча станція для апаратів комп'ютерної і магнітно резонансної томографії. Основна її функціональна особливість – забезпечення радіологів і спеціалізованих клініцистів всіма необхідними інструментами для інтерпретації і аналізу медичних зображень, отриманих з будь-якої модальності (комп'ютерна томографія, МРТ, комп'ютерна і цифрова радіографія, ПЕТ, УЗІ та інше). [6]



*Рис 1.6 Програмне забезпечення «Myrian»*

Розробником ПО “Myrian” є компанія Intrasense (Франція) – провідний виробник програмного забезпечення для обробки і діагностики зображень, отриманих в результаті медичних досліджень. Компанія Intrasense заснована в 2004 р. в Монтпелье (Франція), комерційний офіс знаходиться в Парижі. Команда висококваліфікованих фахівців, високий рівень сертифікації (ISO 13485, ISO 9001, CE Iia, FDA, SFDA), сумісність з найвищими промисловими стандартами (MS certified partner, DICOM 3.0), постійний процес промислового розвитку та клінічного контролю. [6]

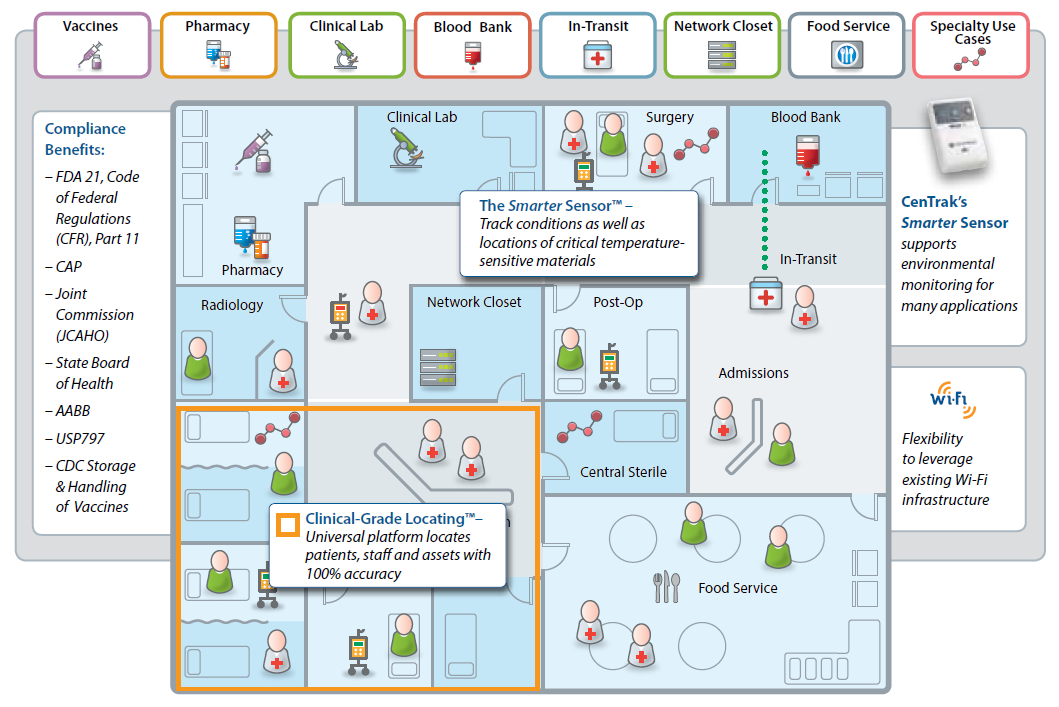
# 1.2.6 Система Centrak

Компанія «МедЕксперт» з 2015 р. є дистрибутором систем Centrak на території країн СНД, окрім Росії.

Співробітники компанії «МедЕксперт» стали сертифікованими спеціалістами з встановлення та налаштування системи. Система «Centrak» – одна з найсучасніших систем з автоматизації діяльності медичного закладу у світі. [8]

Універсальна інфраструктура системи дозволяє за допомогою єдиної платформи керувати різноманітними підсистемами.

Сучасні розробки дозволили створити систему максимально швидкою та гнучкою.



*Рис 1.7 Графічний інтерфейс продукту Centrak.*

# 1.3 Збереження даних

При великих обсягах даних з’являється необхідність в структуризованому збереженні даних. Дані можна зберігати локальному або віддаленому (на сервері) сховищі. Таким сховищем можуть бути різні системи: Block storage, File storage, Object storage і Key-value storage. Перед виборох сховища варто провести тести для перевірки певних параметрів в певних умовах.

Основні характеристики, на які потрібно звернути увагу при виборі системи зберігання:

* Відмовостійкість
* Швидкість відновлення даних
* Продуктивність, відповідна вашим запитам.
* Консистентність даних

# 1.3.1 XML-формат

XML-формат - запропонований консорціумом World Wide Web (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет. Є спрощеною підмножиною мови розмітки SGML. XML-документ складається із текстових знаків, і придатний до читання людиною. [9]

Стандарт XML (англ. Recommendation, перше видання від 10 лютого 1998, останнє, четверте видання 29 вересня 2006) визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Цей формат достатньо гнучкий для того, аби бути придатним для застосування в різних галузях. Іншими словами, запропонований стандарт визначає метамову, на основі якої шляхом запровадження обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних. Ці обмеження описуються мовами схем (англ. Schema), такими як XML Schema (XSD), DTD або RELAX NG. Прикладами мов, заснованих на XML, є: XSLT, XAML, XUL, RSS, MathML, GraphML, XHTML, SVG, а також XML Schema. [9]

# 1.3.2 JSON-формат

JSON формат — це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дозволяє описувати об'єкти та інші структури даних. Цей формат головним чином використовується для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією). Розробив і популяризував формат Дуглас Крокфорд. [10]

JSON знайшов своє головне призначення у написанні веб-програм, а саме при використанні технології AJAX. JSON, що використовується в AJAX, виступає як заміна XML (використовується в AJAX) під час асинхронної передачі структурованої інформації між клієнтом та сервером. При цьому перевагою JSON перед XML є те, що він дозволяє складні структури в атрибутах, займає менше місця і прямо інтерпретується за допомогою JavaScript в об'єкти.

За рахунок своєї лаконічності в порівнянні з XML, формат JSON може бути більш придатним для серіалізації складних структур.

Якщо говорити про веб-застосунки, в такому ключі він доречний в задачах обміну даними як між браузером і сервером (AJAX), так і між самими серверами (програмні HTTP-інтерфейси). Формат JSON так само добре підходить для зберігання складних динамічних структур в реляційних базах даних або файловому кеші.

# 1.3.3 Бінарний формат

Бінарний файл — в широкому сенсі: файл, що містить послідовність довільних байтів. Назва пов'язана з тим, що байти складаються з біт, тобто двійкових цифр.

У вузькому сенсі слова двійкові файли протиставляються текстовим файлам. При цьому, з точки зору технічної реалізації на рівні апаратури, текстові файли є окремим випадком двійкових файлів, і, таким чином, в широкому значенні слова під визначення двійковий файл підходить будь-який файл.

Часто двійковими файлами називають виконувані файли і стиснуті дані, проте так обмежувати це поняття не коректно.

Для візуального перегляду вмісту двійкового файлу він розбивається на шматки рівного розміру, що подаються у вигляді чисел, записуваних, зазвичай, в шістнадцятковій системі, іноді в вісімковій, двійковій або десятковій. Зазначений розмір шматка може дорівнювати одному октету, а також двом або чотирьом (в разі розбиття на шматки по кілька октетів застосовується порядок байтів, прийнятий на використовуваної платформі). [11]

Нерідко крім числових значень байт виводяться також символи у прийнятій в системі або вибраній користувачем кодовій сторінці, наприклад ASCII.

# 1.3.4 База даних

База даних – сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування (за стандартом ISO/IEC 2382:2015). В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки. [1]

База даних (БД) – це структурована сукупність даних, які відображують стан об’єктів певної предметної області та зв’язки між ними. [12]

База даних – це не різновид програмного забепечення, а лише документи, з якими оперують спеціальні прикладні програми, що їх називають системами керування базами даних. Система керування базами даних (СКБД) – це програма, що забезпечує можливість створення БД та виконання різноманітних операцій з даними, які в ній зберігаються. [12]

В загальному випадку базою даних можна вважати будь-який впорядкований набір даних. Наприклад, паперову картотеку з формулярами про працівників підприємства у відділі кадрів. Але дана стаття зосереджена на використанні баз даних в інформаційних системах. На даний час застосунки для роботи з базами даних є одними з найпоширеніших

прикладних програм. [1]

У сучасних інформаційних системах для забезпечення роботи з базами даних використовують системи керування базами даних (СКБД). Система керування базами даних — це система, заснована на програмних та технічних засобах, яка забезпечує визначення, створення, маніпулювання, контроль, керування та використання баз даних (за стандартом ISO/IEC 2382:2015). Застосунки для роботи з базою даних можуть бути частиною СКБД або автономними. Найпопулярнішими СКБД є MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase, Interbase, Firebird та IBM DB2. СКБД дозволяють ефективно працювати з базами даних, обсяг яких робить неможливим їх ручне опрацювання. [1]

Через тісний зв'язок баз даних з СКБД під терміном «база даних» інколи необґрунтовано та неточно мають на увазі систему керування базами даних. Але варто розрізняти базу даних — сховище даних, та СКБД — засоби для роботи з базою даних. СКБД з інформаційної системи може бути видалена, але база даних продовжить існувати. І навпаки: СКБД може функціонувати без жодної бази даних. [1]

В загальному базу даних неможливо просто перемістити з однієї СКБД до іншої. Але СКБД використовують стандарти (SQL, ODBC, JDBC), які уніфікують ряд операцій по роботі з даними і дозволяють різним застосункам працювати з базами даних різних СКБД. СКБД часто класифікують за моделлю організації даних.

Характер зв’язків між елементами бази даних визначає тип організації даних – структуру даних. Розрізняють ієрархічні, реляційні та мережеві бази даних. [13]

Найвживаніші СКБД використовують реляційну модель, у якій дані подають у виді таблиць. Для кінцевого користувача (та прикладних програм) робота з базою даних напряму неможлива. Всі маніпуляції над даними здійснюють через спеціальні запити, які надсилають до СКБД. СКБД опрацьовує їх і повертає результат. Безпосередньо з базою даних працює виключно СКБД. [1]

Сучасні СКБД забезпечують функції щодо керування даними, які можна поділити на такі групи:

* Оголошення даних — створення, зміна та видалення визначень, які описують організацію даних. [1]
* Модифікація даних — додавання даних, їх редагування та видалення. [1]
* Отримання даних — надання даних за запитом застосунку у формі, яка дозволяє їх безпосереднє використання. Дані можуть надаватись або у формі, в якій вони зберігаються у базі даних, або в іншій формі (наприклад, через поєднання різних даних). [1]
* Адміністрування даних — реєстрування та відслідковування дій користувачів, дотримання безпеки роботи з даними, забезпечення надійності та цілісності даних, моніторинг продуктивності, резервне копіювання та відновлення даних тощо. [1]

# Висновок

Для розробки системи обліку пацієнтів 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, найбільший недолік майже усіх систем це дуже висока вартість, лише одна система є безкоштовна, але так як вона призначена для використання в веб-переглядачі суттєво падає продуктивність даної системи.

При реалізації проекту було вирішено базу даних. Я зупинився на сервері Mirosoft Sql Server Express. 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 одними з найбільш універсальних інструментів, які ми маємо в наявності. Вони здатні виконувати вражаючі обчислення, вони дозволяють обмінюватись інформацією легко, незалежно від їх фізичного місцезнаходження, вони спрощують багато завдань на кожен день, також вони дозволяють автоматизувати багато нудних та рутинних процесів. Однак комп'ютери не є «інтелектуальними», як ми. Тому є величезний мовний бар'єр між людиною, яка хоче щоб комп'ютер зробив щось, і комп'ютером, який зазвичай вимагає вказівки на його рідній мові (машинний код), щоб зробити що-небудь. Комп’ютери самі по собі нічого зробити не можуть, аби комп’ютери працювали за призначенням треба надати набори інструкцій, які створені людиною.

# 2.1. С++ мова

С ++ є універсальною об'єктно-орієнтованою (ООП) мовою програмування, розроблена Б'ярном Страуструпом, і є розширенням мови C. Таким чином, можна програмувати на C++ в стилі «C» або в «об'єктно-орієнтованому стилі». [14]



*Рис. 2.1 Логотип С++*

С++ - це мова проміжного рівня, так як вона інкапсулює можливості мов високого і низького рівня. Спочатку мова називалася як «С з класами», так як це була мова з усіма властивостями мови С з додатковим поняттям «класів». Тим не менш, вона була перейменована в C++ в 1983 році. Воно вимовляється як «сі-плюс-плюс».

С ++ є одною з найпопулярніших мов в основному використовуваних для системного / прикладного програмного забезпечення, драйверів, клієнт-серверних додатків і вбудованого мікропрограмного забезпечення.

Основна родзинка C ++ це набір визначених класів, які є типами даних, що можуть бути створені декілька разів. Мова також полегшує оголошення класів, визначених користувачем. Класи можуть додатково пристосувати функції-члени для впровадження певної функціональності. Кілька об'єктів певного класу можуть бути визначені для виконання в функціях-членах класу. Об'єкти можуть бути визначені як екземпляри, створені під час виконання. Ці класи також можуть бути успадковані іншими новими класами, які за замовчуванням приймають публічні і захищені можливості. [14]

C ++, включає в себе кілька операторів, таких як порівняння, арифметичні операції, побітові маніпуляції і логічні оператори. Одним з найбільш привабливих особливостей C ++ є те, що вона дозволяє перевантажувати деякі оператори, для прикладу додавання.

Деякі з основних понять в мові програмування C++ включають поліморфізм, віртуальні і дружні функції, шаблони, простори імен і вказівники.

С++ це:

* ... відкрита ISO-стандартизована мова.

Протягом деякого часу, C++ не мала жодного офіційного стандарту і підтримувалась стандартом де-факто, проте з 1998 року, C++ була стандартизована комітетом ISO.

* ... компільована мова.

C++ компілюється безпосередньо в машинний код комп'ютера, що дозволяє їй бути одною з найшвидших в світі мов, якщо звісно все оптимізовано.

* ... це строго типізована небезпечна мова.

C++ є мовою, де очікується, що програміст знає, що він або вона робить, але допускає неймовірну кількість контролю в результаті.

* ... мова, що підтримує явне і неявне оголошення.

За даними останнього стандарту C++, C++ підтримує як явне і так і неявне оголошення, що дає гнучкість і можливість уникнути багатослів'я, де це необхідно.

* ... мова, що підтримує як статичний так і динамічний контроль типів.

C ++ дозволяє перевіряти перетворення типів або під час компіляції або під час виконання, знову пропонуючи іншу ступінь гнучкості. Проте, більшість перевірок типів в C++ є статичними.

* ... мова, що пропонує безліч варіантів парадигми.

C ++ пропонує чудову підтримку процедурних, загальних і об'єктно-орієнтованих парадигми програмування, також багато інших парадигмам можуть бути застосовані.

* ... мова, що є портативною.

Однією з найбільш часто використовуваних мов в світі, і як відкрита мова, C++ має широкий ряд компіляторів, які працюють на різних платформах, які підтримують його. Код, який виключно використовує стандартну бібліотеку C++'s буде працювати на багатьох платформах без змін або з незначними змінами.

* ... мова сумісна з C

C++, будучи мовою, яка безпосередньо на основі C, сумісна практично з усім кодом C. C++ може використовувати бібліотеки C без будь-яких змін коду (або дуже малою зміною) в бібліотеках.

* ... мова, що має неймовірну підтримку бібліотеки.

Пошуки «бібліотека» на популярному сайті з управління проектами SourceForge дасть більше 3000 результатів для бібліотек C++.

# 2.2. Visual Studio

Microsoft Visual Studio - це набір інструментів для створення програмного забезпечення: від планування до розробки призначеного для користувача інтерфейсу, написання коду, тестування, налагодження, аналізу якості коду і продуктивності, розгортання в середовищах клієнтів і збору даних телеметрії по використанню. Ці інструменти призначені для максимально ефективної спільної роботи; всі вони доступні в інтегрованому середовищі розробки (IDE) Visual Studio. [15]



*Рис. 2.2 Логотип Visual Studio*

Visual Studio можна використовувати для створення різних типів додатків, від простих додатків для магазину та ігор для мобільних клієнтів до великих і складних систем, які обслуговують підприємства та центри обробки даних. Ви можете створювати:

1. додатки та ігри, які виконуються не тільки на платформі Windows, але і на Android і iOS;
2. веб-сайти і веб-служби на основі ASP.NET, JQuery, AngularJS і інших популярних платформ;
3. додатки для різних платформ і пристроїв, включаючи, але не обмежуючись: Office, Sharepoint, Hololens, Kinect і "Інтернету речей";
4. iгри і графічні додатки для різних пристроїв Windows, включаючи Xbox, з підтримкою DirectX.

За замовчуванням Visual Studio забезпечує підтримку C #, C і C ++, JavaScript, F # і Visual Basic. Visual Studio добре працює і інтегрується зі сторонніми додатками, наприклад Unity і Apache Cordova, за допомогою розширень Набір засобів Visual Studio для Unity і інструментів Visual Studio для Apache Cordova відповідно. Ви також можете самостійно розширити Visual Studio, створивши власні інструменти для виконання спеціалізованих завдань.

**Створення рішення та проекту**

У всіх випусках Visual C ++ вихідний код і файли, пов'язані з виконуваним файлом (наприклад, EXE, DLL або LIB), упорядковуються в проект. Проект містить файл проекту у форматі XML (з розширенням VCXPROJ), який визначає всі файли і ресурси, необхідні для компіляції програми, а також інші параметри конфігурації, наприклад цільову платформу (x86, x64 або ARM), і визначає побудову остаточної версії або налагоджувальної версії програми. Проект (або багато проектів) знаходяться в рішенні. Наприклад, рішення може містити кілька проектів Win32 DLL і один консольний додаток Win32, що використовує ці бібліотеки DLL. При побудові проекту модуль MSBuild використовує файл проекту і створює виконуваний файл і (або) інші вихідні дані, зазначені розробником.

* Шаблони проектів

До складу Visual C ++ входить кілька шаблонів проекту, які містять початковий код і параметри, необхідні для різних типів простої програми. Зазвичай починають з вибору файл -> Новий проект -> Створення проекту на основі шаблону проекту, потім додайте нові файли вихідного коду на цей проект, або почніть писати код в файли до складу.

* Майстри додатків

Visual C ++ надає майстри для деяких типів проектів. Майстер керує покроковим процесом створення нового проекту. Це зручно для типів проектів, в яких є багато параметрів і налаштувань.

**Побудова (компіляція і компонування)**

Натисніть клавіші Ctrl + Shift + B для компіляції і компоновки проекту. Для створення виконуваного коду в Visual Studio використовується MSBuild. Можна задати параметри побудови користуючись пунктом засоби - > параметри -> Проекти та рішення і можна задати властивості для певних проектів у їх властивостях. Помилки побудови і попередження відображаються в списку помилок (Ctrl + \, E). Додаткові відомості іноді відображаються у вікні виводу (Alt + 2).

Можна також використовувати компілятор Visual C ++ (cl.exe) і багато інших пов'язані з побудовою автономні засоби, такі як NMAKE і LIB, безпосередньо з командного рядка.

**Тестування**

Visual Studio включає платформу модульного тестування для некерованого коду C ++ і C ++ / CLI. Додаткові відомості див. У розділах Перевірка коду за допомогою модульних тестів і Написання модульних тестів для мови C / C ++ з використанням платформи модульного тестування Майкрософт для C ++.

Засоби модульних тестів включають:

* Оглядач тестів. Оглядач тестів дозволяє виконувати модульні тести і переглядати їх результати. Оглядач тестів може використовувати будь-які тестові платформи, в тому числі сторонні платформи, які мають адаптер для оглядача.
* Платформа для тестування Microsoft для керованого коду. Платформа для тестування Microsoft для керованого коду встановлюється з Visual Studio і надає середовище для тестування коду в .NET.
* Платформа для виконання модульних тестів Microsoft для C ++. Платформа для виконання модульних тестів Microsoft для C ++ встановлюється з Visual Studio і надає середовище для тестування машинного коду.
* Засоби покриття коду. Можна визначити обсяг коду продукту, який покривають модульні тести, за допомогою однієї команди в Оглядачі тестів.
* Межі ізоляції Microsoft Fakes. Межі ізоляції Microsoft Fakes можуть створити постановочні класи і методи для робочого коду і систем, які створюють залежно в тестованому коді. Шляхом реалізації підставних делегатів для функції можна контролювати поведінку і повертаються значення об'єкта залежності.

**Налагодження**

Щоб налагодити програму, натисніть клавішу F5, якщо для конфігурації проекту вибрано значення "Налагодження". Під час налагодження можна встановлювати точки зупинки, натиснувши клавішу F9, покроково виконавши код за допомогою клавіші, переглядати значення зазначених змінних або регістрів і навіть в деяких випадках вносити зміни в код і продовжувати налагодження без повторної компіляції.

Під час зупинки виконання, наприклад при досягненні зневаджувачем точки зупинення, можна здійснювати послідовне виконання коду за допомогою трьох команд меню Налагодження (див. табл. *2.1*).

*Таблиця 2.1*

**Команди налагодження**

|  |  |  |
| --- | --- | --- |
| **Команда меню** | **Клавіші сполучення** | **Опис** |
| Крок з входом | F11 | Якщо рядок містить виклик функції, команда Крок з входом виконує тільки сам виклик, а потім зупиняє виконання в першому рядку коду всередині функції. В іншому випадку команда Крок з входом виконує наступний оператор. |
| Крок з обходом | F10 | Якщо рядок містить виклик функції, команда Крок з обходом виконує функцію, що викликається, а потім зупиняє виконання в першому рядку коду всередині виклику функції. В іншому випадку команда Крок з входом виконує наступний оператор. |
| Крок з виходом | Shift+F11 | Команду Крок з виходом відновлює виконання коду до повернення функції, а потім перериває виконання в точці повернення виклику функції. |

# 2.3. Qt Framework

Qt це крос-платформненний фреймворк розробки додатків для настільних комп'ютерів, вбудованих систем і мобільних пристроїв. Підтримувані платформи включають Linux, OS X, Windows, VxWorks, QNX, Android, IOS, BlackBerry, Sailfish OS і інші.Qt не є мовою програмування. Це фреймворк написаний на C++. Препроцесор, МОC (Мета-об'єктний компілятор), використовується для розширення мови C ++ з функціоналом, як сигнали і слоти. Перед стадією компіляції, МОC аналізує вихідні файли, написані на C++ розширеному з Qt і генерує стандартні сумісні C++ коди з них. Таким чином, фреймворк сам по собі і додатки/бібліотеки, які використовують його, можуть бути скомпільовані будь-яким стандартним C++ компілятором як Clang, GCC, ICC, MinGW і MSVC. [16]



*Рис. 2.3 Логотип Qt*

Бібліотека Qt замислювалася і починалася як крос-платформний інструментарій (toolkit) для швидкої розробки графічних інтерфейсів (GUI) додатків на мові C++, з метою спростити життя програмістів, які пишуть на C++ крос-платформні, що переносяться GUI-додатки, які повинні працювати і в середовищі Windows, Symbian, і в середовищі Unix / Linux під X11, і на комп'ютерах Macintosh.

В даний час Qt значно переріс рамки інструментарія для розробки графічних інтерфейсів додатків. Він надає програмісту, що її використовує, цілісний фреймворк (framework), що дозволяє при написанні більшої частини програми використовувати тільки «рідні» класи Qt і практично повністю відмовитися від написання системно-залежного коду, використання системних викликів (будь то Win32 API або Unix system виклики) або від винаходів власних крос-платформних обгорток і «велосипедів». Класи Qt покривають майже всі потреби програміста. В Qt передбачені класи і для роботи з рядками, і для роботи з файлами, мережею, базами даних, XML, і для забезпечення многопоточности в додатку, і багато-багато іншого.

Qt надає програмісту не тільки зручний набір бібліотек класів, а й певну модель розробки додатків, певний каркас їх структури. Дотримання принципів і правил «гарного стилю програмування на C++ / Qt» істотно знижує частоту таких важко відловлюваних помилок в додатках, як витоки пам'яті (memory leaks), необроблені виключення, незакриті файли або не вивільнені дескриптори ресурсних об'єктів, чим нерідко страждають програми, написані «на чистому C++» без використання бібліотеки Qt.

Важливою перевагою Qt є добре продуманий, логічний і стрункий набір класів, що надає програмісту дуже високий рівень абстракції. Завдяки цьому програмістам, які використовують Qt, доводиться писати значно менше коду, ніж це має місце при використанні, наприклад, бібліотеки класів MFC. Сам код виглядає стрункіше і простіше, логічніше і зрозуміліше, ніж аналогічний за функціональністю код MFC або код, написаний з використанням «рідного» для X11 тулкіта Xt. Його легше підтримувати і розвивати.

Крім того, навіть якщо програмісту в даний конкретний момент не потрібна кроссплатформенність для його конкретного додатка (наприклад, планується версія тільки для Windows або тільки для Macintosh), ніхто не може знати, що знадобиться завтра. Бізнес-плани можуть змінитися, і може виявитися необхідним, і вигідним випустити версію для іншої операційної системи або інший апаратної платформи. У разі використання Qt для цього знадобиться всього лише перекомпіляція вихідного коду. У разі ж використання, наприклад, MFC або «рідних» системних API знадобиться багато важкої роботи по перенесенню, адаптації та налагодженні, а то і переписування з нуля існуючого вихідного коду для іншої ОС або апаратної платформи.

Багато компаній-розробників додатків Windows використовують Qt ще з однієї причини: навіть якщо код пишеться і буде писатися тільки для платформи Windows і тестується тільки на ній, можливість скомпілювати один і той же вихідний код на одній і тій же платформі Windows двома різними компиляторами (Microsoft Visual C ++ і GCC / Win32) гарантує кращу якість вихідного коду і кращу його сумісність зі стандартом C++. Що важливо для коду, який планується довго підтримувати і розвивати.

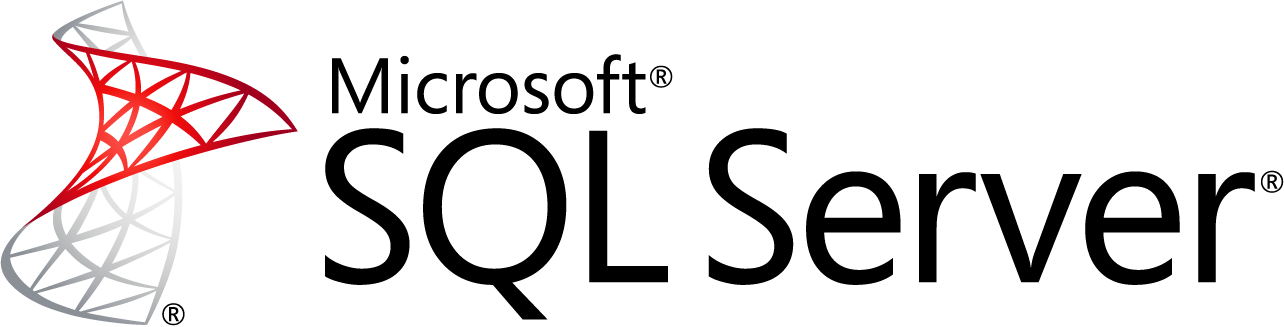
**Qt Designer**

Qt Designer - інструмент для проектування і створення графічних користувацьких інтерфейсів (GUI) з компонентів Qt. Ви можете створити і налаштувати свої віджети або діалоги в режимі "що бачиш, то і отримаєш" (what-you-see-is-what-you-get, WYSIWYG), і перевірити їх використовуючи різні стилі і дозволені дії.

Віджети та форми, створені за допомогою Qt Designer, бесшовно інтегровані з керуючим кодом, який використовує механізм сигналів та слотів Qt, який дозволяє вам легко встановити поведінку до графічних елементів. Всі властивості встановлені в Qt Designer можна змінити динамічно всередині коду. Крім того, такі можливості як просування (promotion) віджетів і підключені користувацькі модулі дозволяє вам використовувати в Qt Designer ваші власні компоненти. [17]

# 2.4. SQL мова

SQL (Структурована мова запитів) - це мова, яка дає вам можливість створювати і працювати в реляційних базах даних, які є наборами зв'язаної інформації, що зберігається в таблицях. [19]



*Рис. 2.4 Microsoft SQL Server*

Інформаційний простір стає більш уніфікованим. Це призвело до необхідності створення стандартної мови, яка могла би використовуватися у великій кількості комп'ютерних середовищ різних видів. Стандартна мова дозволить користувачам, які знають один набір команд, використовувати їх для створення, знаходження, зміни та передачі інформації - незалежно від того, чи працюють вони на персональному комп'ютері, мережевій робочій станції, або на універсальній ЕОМ.

У нашому все більш і більш взаємопов'язаному комп'ютерному світі, користувач що постачається такою мовою, має величезну перевагу у використанні та узагальненні інформації з ряду джерел за допомогою великої кількості способів.

Елегантність і незалежність від специфіки комп'ютерних технологій, а також його підтримка лідерами промисловості в області технології реляційних баз даних, зробило SQL (і, ймовірно, протягом осяжного майбутнього залишить його) основною стандартною мовою.

Стандарт SQL визначається ANSI (Американським Національним Інститутом Стандартів) і в даний час також приймається ISO (Міжнародна Організація по Стандартизації). Однак, більшість комерційних програм баз даних розширюють SQL без повідомлення ANSI, додаючи різні особливості в цю мову, які вони вважають вельми корисними. Іноді вони дещо порушують стандарт мови, хоча хороші ідеї мають тенденцію розвиватися і незабаром ставати стандартами "ринку" самі по собі в силу корисності своїх якостей.

**Microsoft SQL Server**

Рішення Microsoft SQL Server презентує собою високопродуктивну платформу обробки і аналізу даних для бізнесу будь-якого розміру, яка відповідає всім сучасним вимогам по роботі з даними будь-яких типів і підтримки зручної і швидкої розробки додатків. Продукт реалізує концепцію комплексної обробки даних і дозволяє перетворювати цінну інформацію в знання і рішення на основі аналізу зібраних даних. У базах даних SQL Server можна зберігати будь-яку структуровану, напівструктуровану або неструктуровану інформацію, таку наприклад, як зображення і мультимедіа з найрізноманітніших джерел даних. Продукт пропонує великий набір інтегрованих служб, які розширюють можливості обробки, який дозволяє складати запити, виконувати пошук, проводити синхронізацію, формувати звіти і аналізувати дані. SQL Server забезпечує звернення до даних з будь-якої програми, розробленого із застосуванням різних технологій. [20]

**ODBC драйвер**

ODBC - Open Database Connectivity це інтерфейс доступу до баз даних в середовищі Windows. Для доступу використовуються спеціальні драйвера подібно до того як використовуються драйвера принтера. Яку б всередині складно не був влаштований принтер команди звернення до нього та друку однакові і кожна програма може друкувати на принтері. Розробник будь-якої бази даних може створити свій драйвер ODBC.

ODBC реалізує інтерфейс доступу до різних SQL сумісним базам даних.

Ідея полягає в тому, що додаток може отримувати доступ до абсолютно різних баз даних не змінюючи при цьому код. Ось в чому переваги ODBC:

* API функції однакові і не залежать від постачальника
* SQL оператори можуть бути згенеровані на будь-якій стадії при компіляції або виконанні.

Дані приймаються в програму в єдиному форматі.Для роботи з джерелами даних використовують імена DNS - Data Source Name - іменований джерело даних ODBC. Диспетчер використовує інформацію пов'язану з ім'ям для доступу до БД. З іменем пов'язана наступна інформація:

* місцезнаходження
* Тип драйвера
* Інші обов'язкові параметри

Існує три типи імен DNS:

1. Призначений для користувача
2. системний
3. файловий

У першому випадку інформація зберігається в реєстрі Windows і прив'язана до конкретного користувача, у другому випадку до конкретного комп'ютера і кожен користувач має доступ, в останньому випадку інфомація зберігається в файлі, що полегшує перенесення проекту з комп'ютера на комп'ютер.

Драйвери можуть підтримувати три рівня граматики SQL мінімальна граматика підтримується всіма драйверами:

* Мінімальна - Create Table, Drop Dtable Select, Insert, Update Dearch, Delete Search Char, VarChar або Long VarChar
* Основна - Мінімальна + Alter Table, Create Index, Drop Index, Create View, DropView, Grant, Revoke Select (повна) Decimal, Numeric, SmallInt, Integer, Real, Float, Double Precision
* Розширена - Основна + зовнішні з'єднання, підтримка позиціювання, Bit, TinyInt, BigInt, Binary, VarVariant, Long, VarBinary, Date, Time, TimeStamp пакетні оператори SQL, виклик процедур

# РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ

# 3.1. Розробка архітектури системи

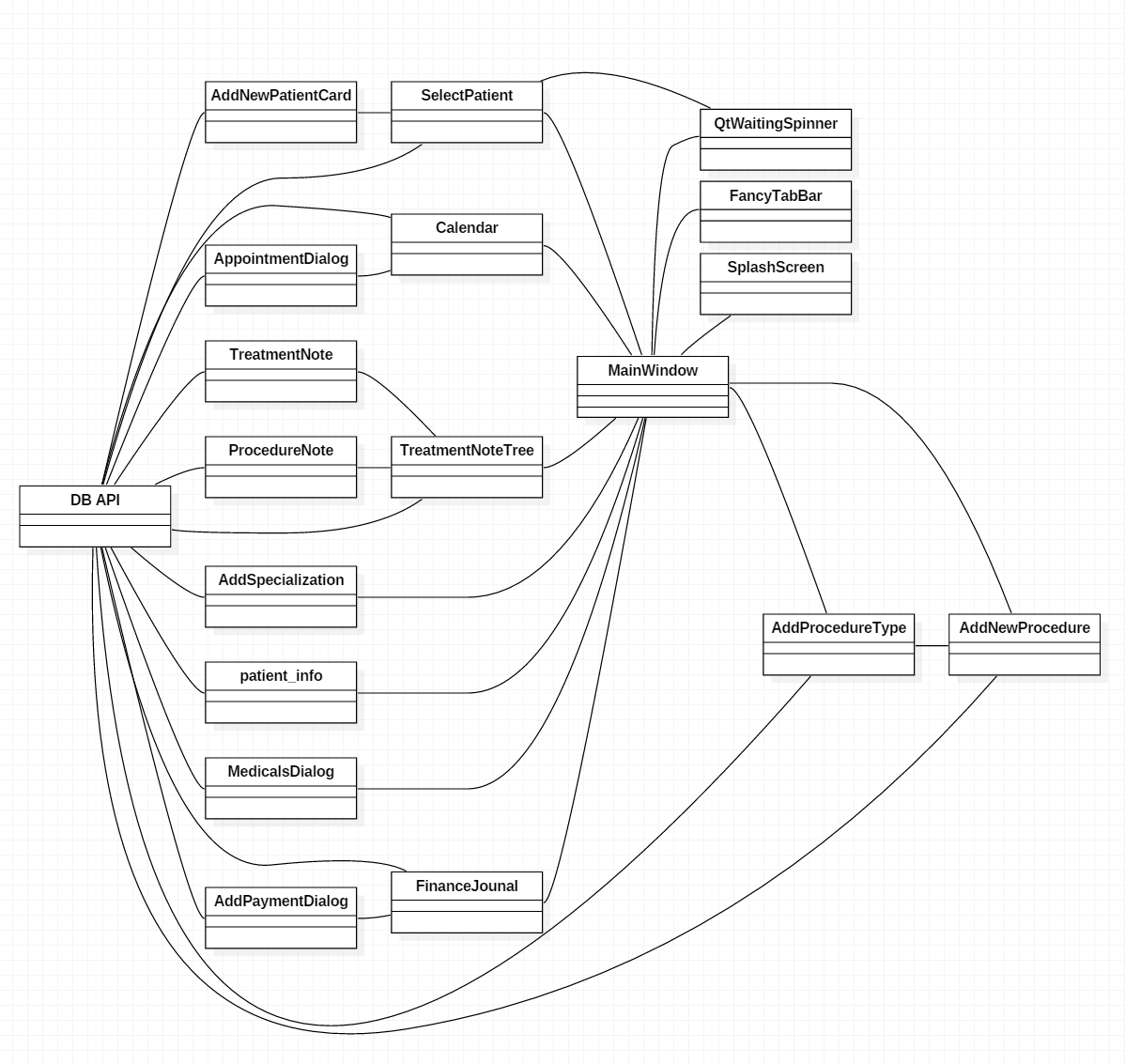
Розробка програми організована таким чином, що система будується з сукупності невеликих незалежних блоків, званих модулями, структура і поведінка яких підкоряються певним правилам. Це дозволяє спростити тестування програми і виявлення помилок. Апаратно-залежні підзадачі можуть бути строго відділені від інших підзадач, що покращує мобільність створюваних програм.

Модуль - функціонально закінчений фрагмент програми, що оформляється у вигляді окремого файлу з вихідним кодом або пойменованої безперервної її частини.

Загалом архітектура виглядає наступним чином:

* Модуль для управління БД
* Основний модуль (модуль, що містить *main()* функцію)
* Інші модулі

Нижче можна побачити краще взаємозв’язок модулів, де головне вікно містить діалоги, графічні компоненти, вікно завантаження, компоненту завантаження та панель закладок. Усі діалоги та графічні компоненти, що маніпулють даними містять компоненту DB API та інші діалоги.



*Рис. 3.1 Діаграма взаємозв’язоку модулів системи*

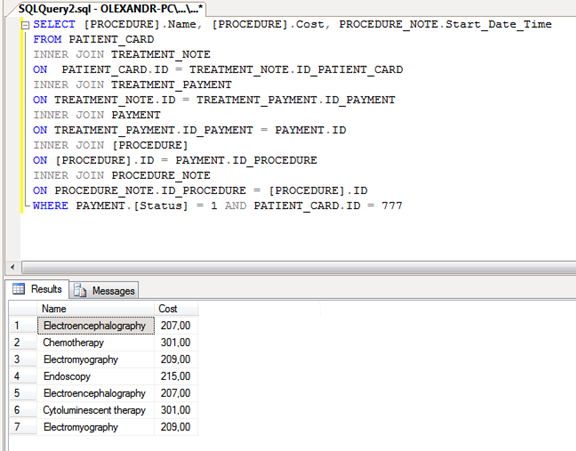
# 3.2. Розробка бази даних

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:

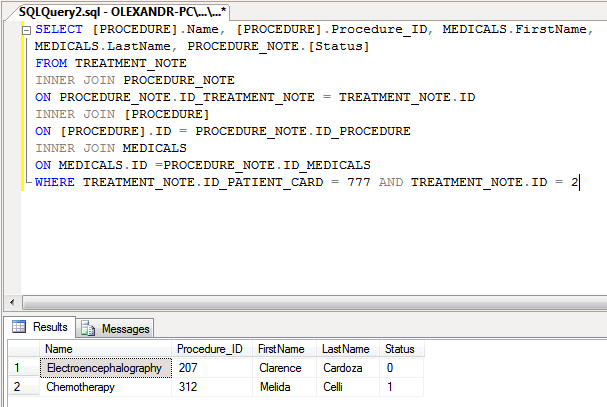
*D:\Temp\SS\OLEXANDR\UML\База даних Hospital-6.pngРис. 3.2 ER-діаграма**бази даних*

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*

# 3.3. Розробка дизайну програми

Кожен раз, проектуючи інтерфейс UI-дизайнер має задавати такі питання: «Яка інформація зараз важлива для кінцевого користувача? Як ми розподілимо його увагу в конкретному випадку? »Для цього в нашому озброєнні є колір і його відтінки, розмір шрифту, його інтенсивність. В сукупності, правильно використовуючи ці інструменти, ми «залишає послання» користувачу, ведемо його по потрібним нам шляхам, концентруючи його увагу на найважливішому.

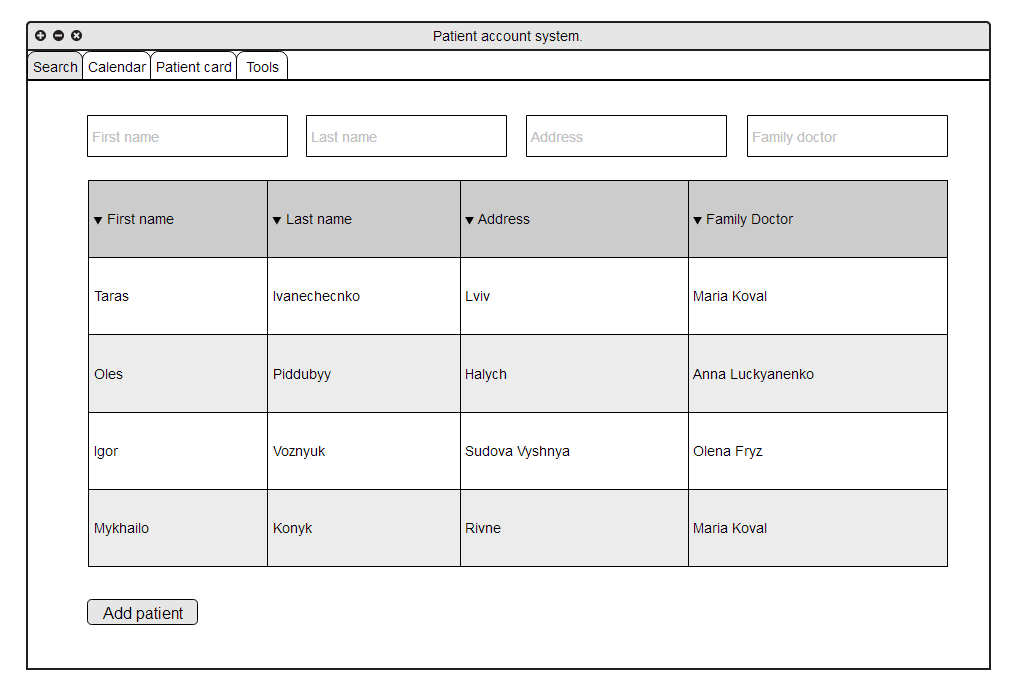
Не забувайте, що саме текст є визначальним фактором того, яке очікування або реакція попередньо сформується у користувача при вигляді кнопки. І лише зображення іконки вторинним чином доповнює зміст. Зображення дзвоника з написом «повідомлення» дасть нам деяке уявлення про призначення цього функцион до того, як ми зробили клік. Аналогічний дзвіночок без підпису в іншому додатку призведе нас до будильника, хоча ми швидше за все будемо очікувати появу екрану з повідомленнями.

Тому дизайн даної системи має бути не втомлюючим, простим але з хорошим функціоналом. Найкращим рішенням для даної системи є розбиття на логічні блоки, цими блоками виступають вкладки. Тому спочатку варто почати з розробки заготовок.

Заготовка - це технічний документ, що побудований на таких елементах як: лінії, блоки, підписи. Також можливе використання пари колірних акцентів.

Сама заготовка вказує лиш на логіку побудови графічного інтерфейсу, а не на специфічні речі. Малювання заготовок може займати як одну годину , так і цілий тиждень.

Для прикладу розглянемо заготовку-дизайн вкладки пошуку:



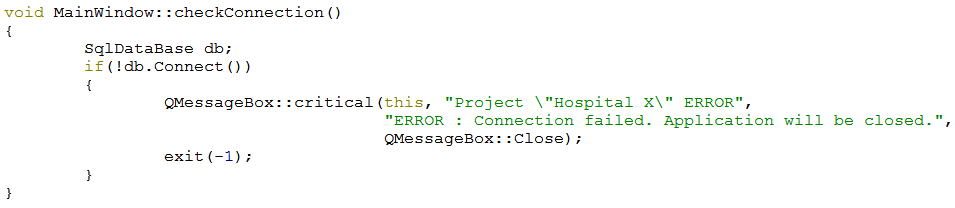
*Рис. 3.5 Дизайн-заготовка вкладки пошуку*

Як бачимо з заготовки вкладки розташовані на місці звичному для пересічного користувача (Те ж розміщення що і в програмах Microsoft Word, Mozilla Firefox, Chrome). Частина, яка для користувача є найважливішою, а це таблиця з даними, займає найбільше місця в просторі системи.

При цьому дизайн це лише концепція, остаточний продукт може набагато відрізнятись від початкового задуму в дизайні.

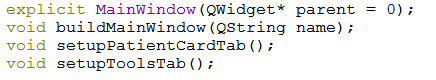
# 3.4. Розробка системи

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;

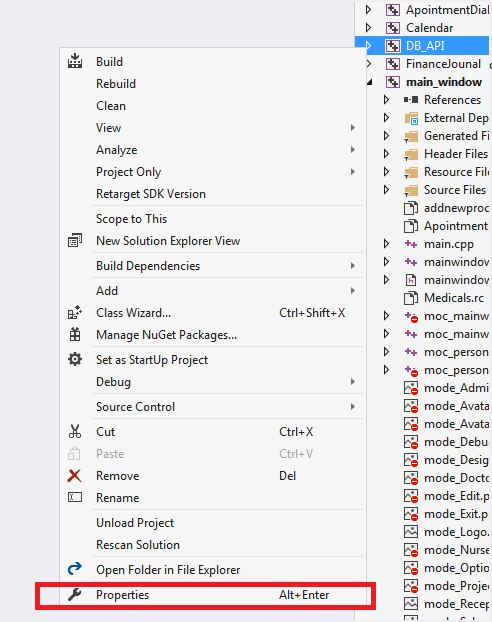
А клас DBSqlQueryвикористовується для виконання SQL запиту до сервера, і має такий набір публічних методів як:

* 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;

# 3.5. 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:

****

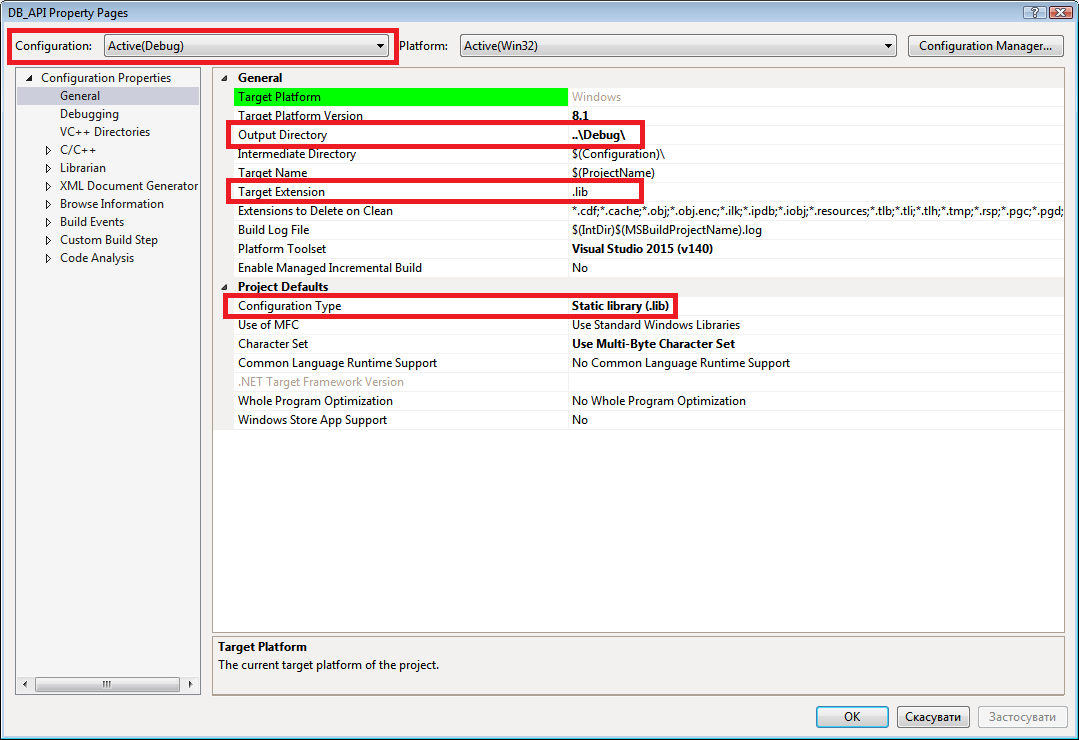
*Рис. 3.9 Налаштування проекту*

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

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

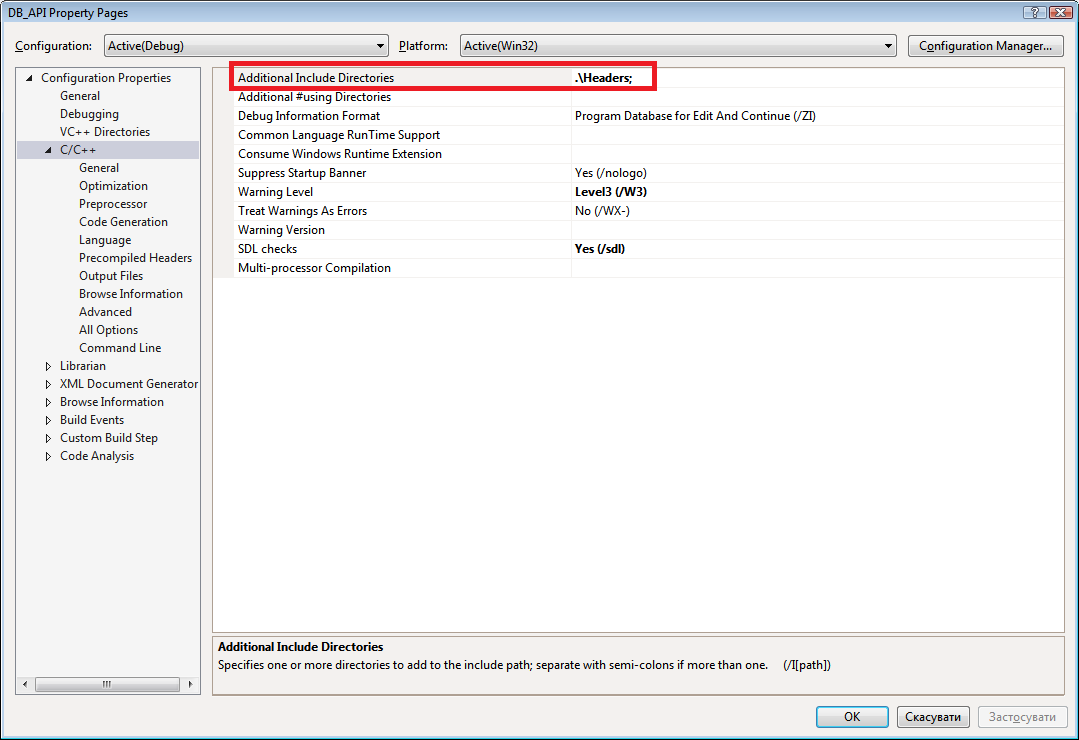
\* Розширення цілі (Target Extension) встановити .lib

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



*Рис. 3.10 Загальні налаштування проекту*

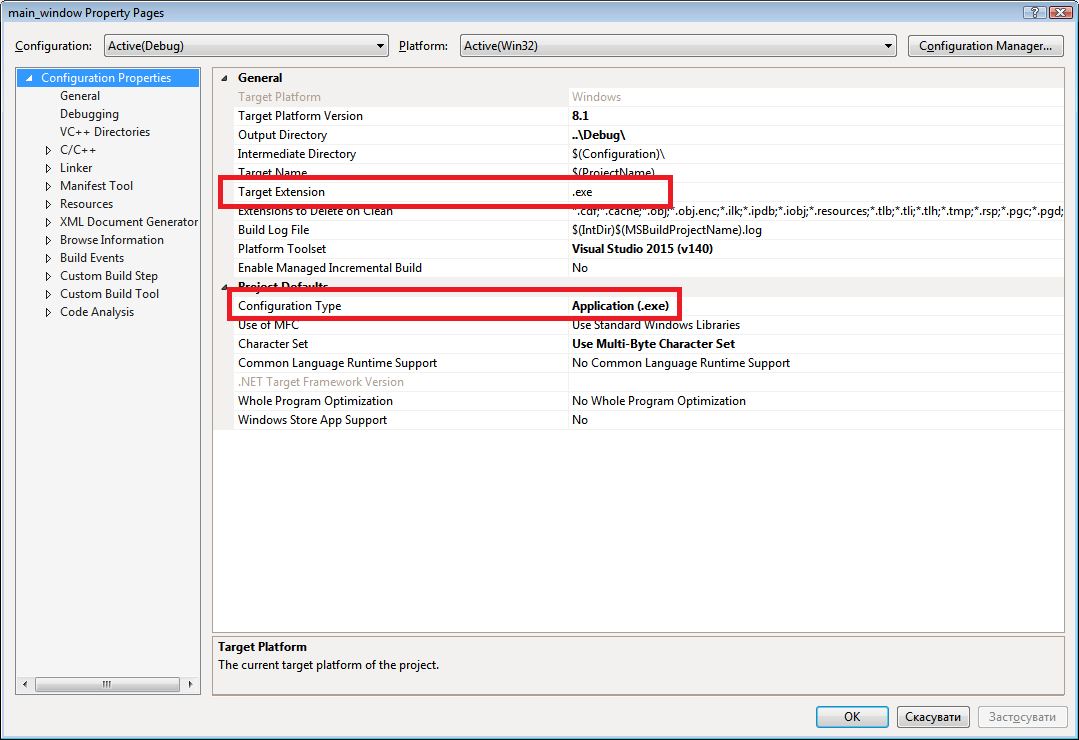
Наступним кроком буде налаштування додаткових включень бібліотек, так як DB\_API не використовує інші модулі системи, тут буде тільки власні бібліотеки:



*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.

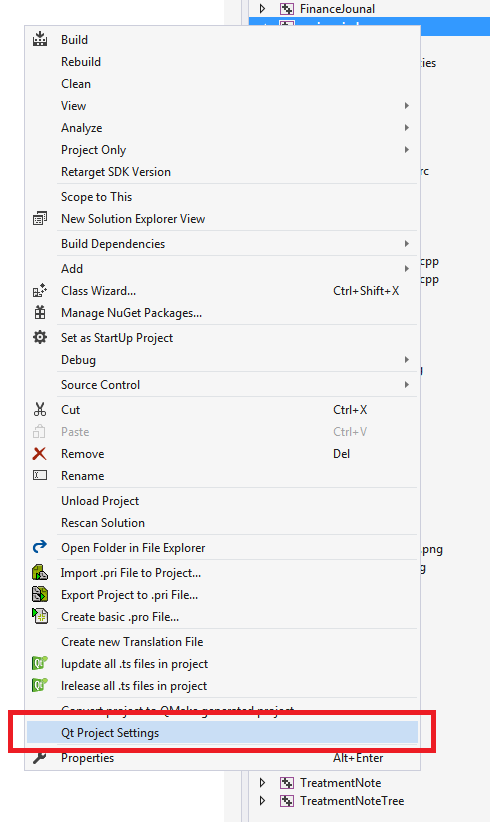
Даний приклад демонструє як зібрати модуль як статичну бібліотеку, також нижче буде розглянуто як зібрати основний модуль, як виконуваний файл. Для цього відкриваємо властивості проекту main\_window, як це було зроблено раніше. Так як даний проект містить функцію входу main(…) вихідний файл збірки буде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

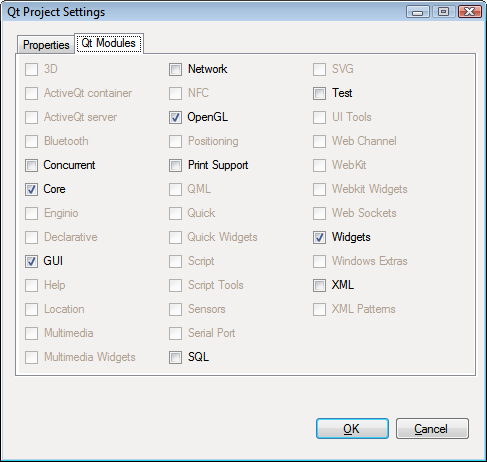
і вибираємо Qt Project Settings.

****

*Рис. 3.13 Загальні налаштування основного проекту*

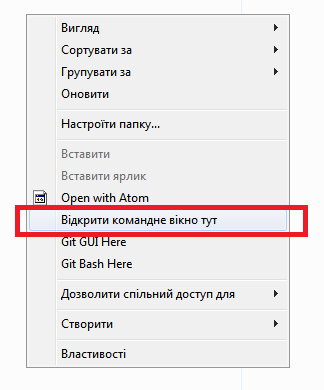
У вікні Qt Project Settings на вкладці Qt Modules вибираємо усі модулі, які були використані в даному проекті, а саме:

* Core
* GUI
* OpenGL
* Widgets



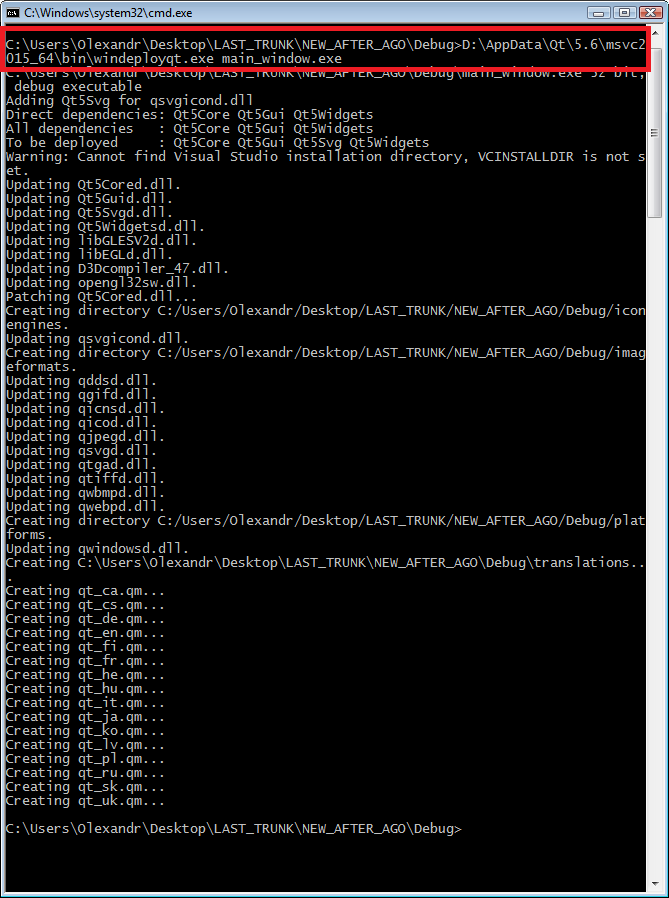
*Рис. 3.14 Вікно налаштувань “Qt Project Settings” для головного проекту*

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:



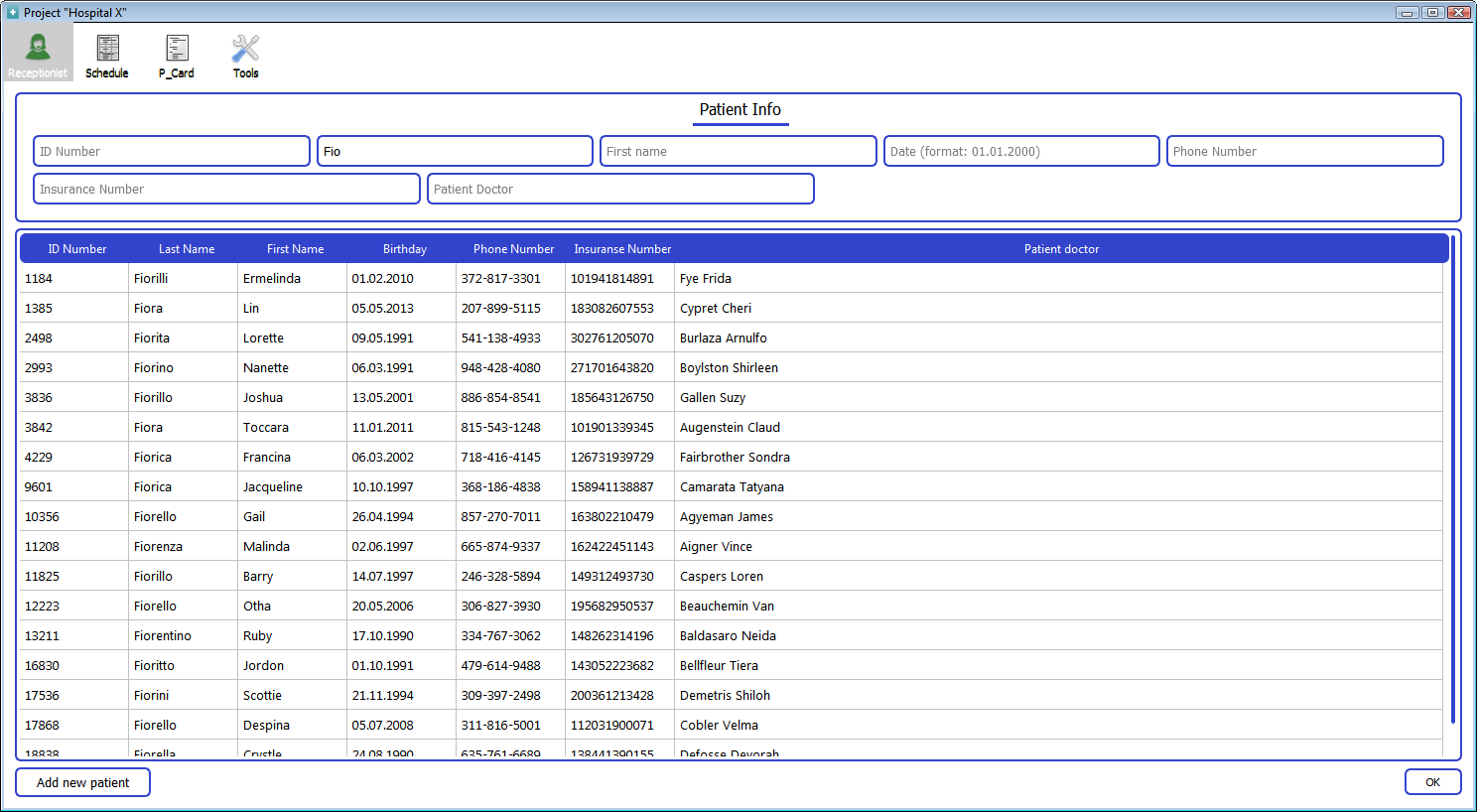
*Рис. 3.15Меню в “Explorer”*

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*

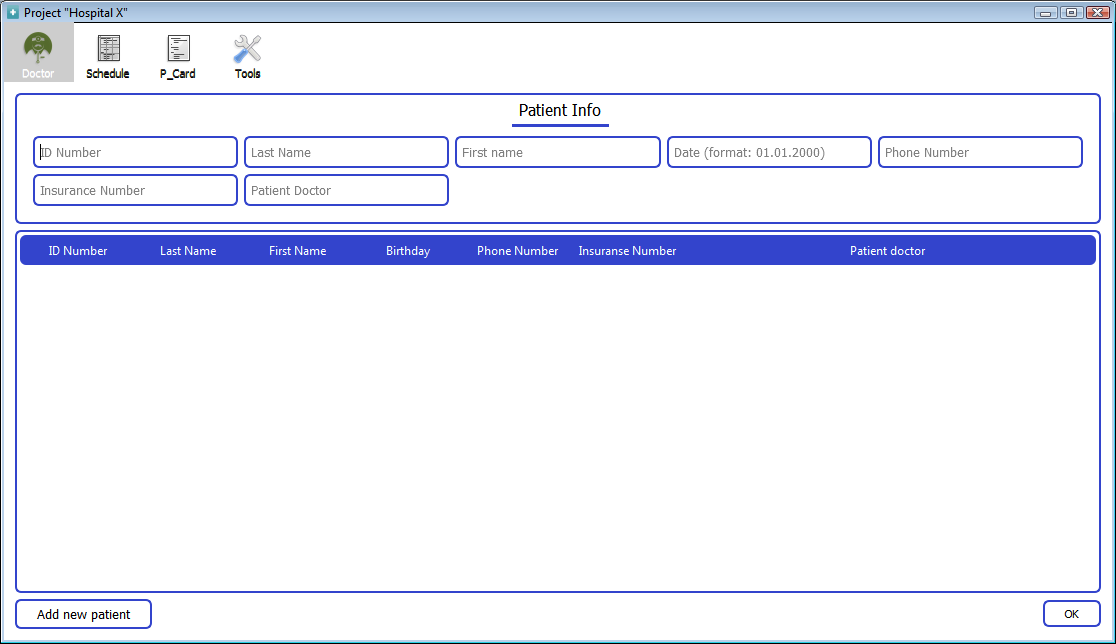
В результаті отримуємо повноцінний виконуваний файл, яким можна ділитись.Запускаємо програму для перевірки робочого стану:



*Рис. 3.17 Запуск системи з виконуваного файлу*

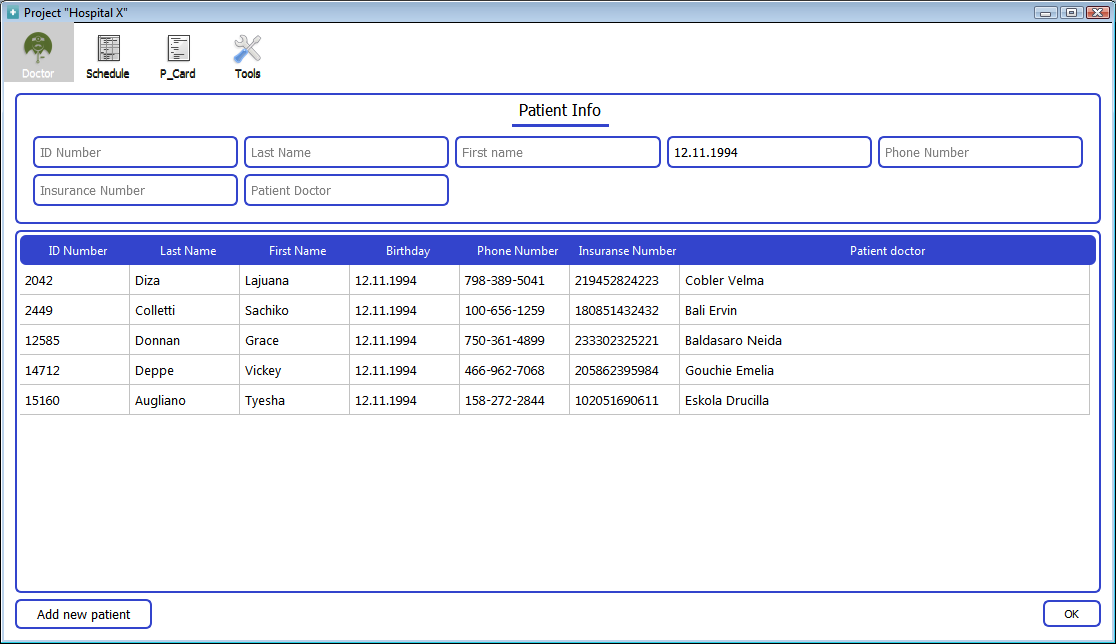
# 3.6. Основний функціонал системи

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



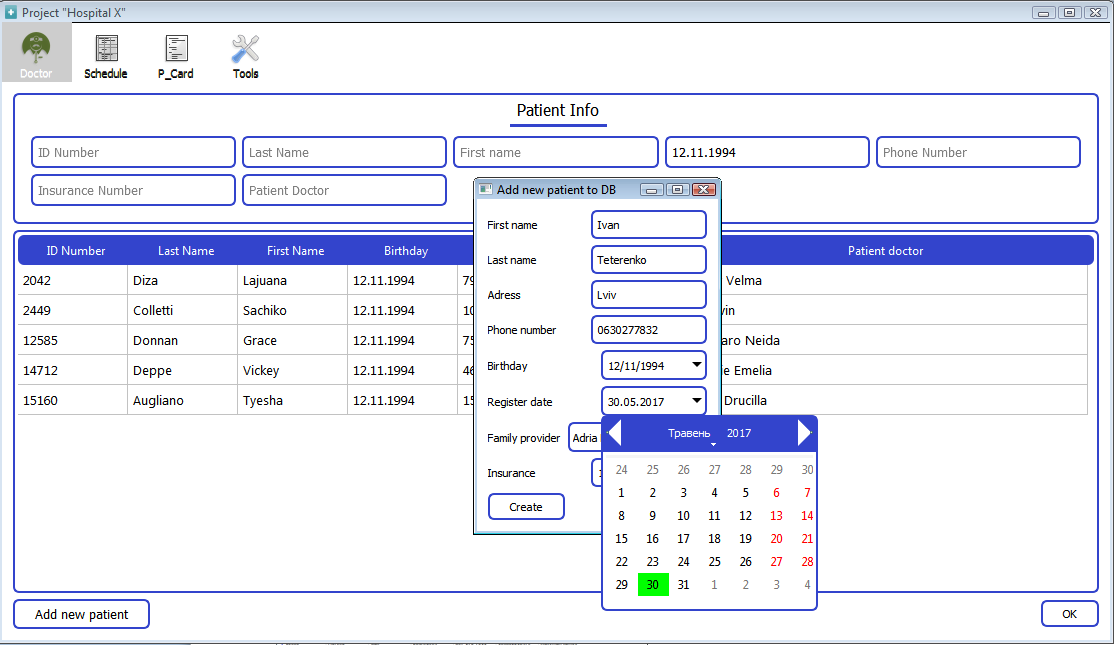
*Рис. 3.18 Перша вкладка системи «Пошук»*

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:



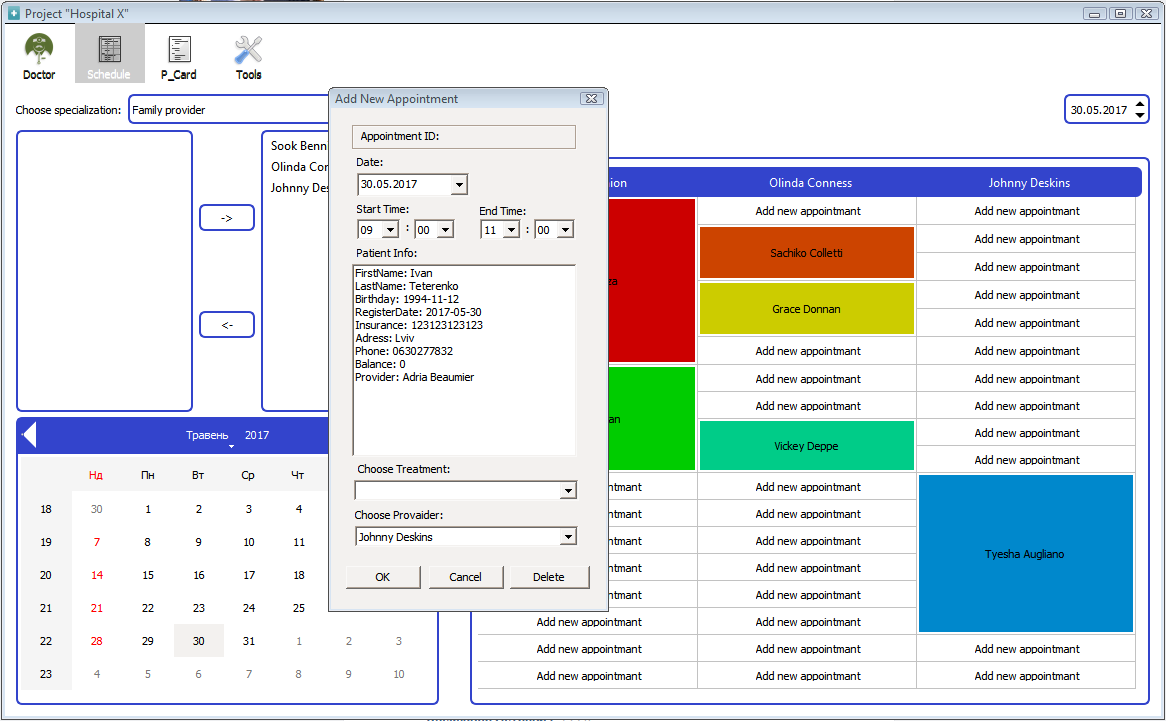
*Рис. 3.19 Результат пошуку по даті*

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:

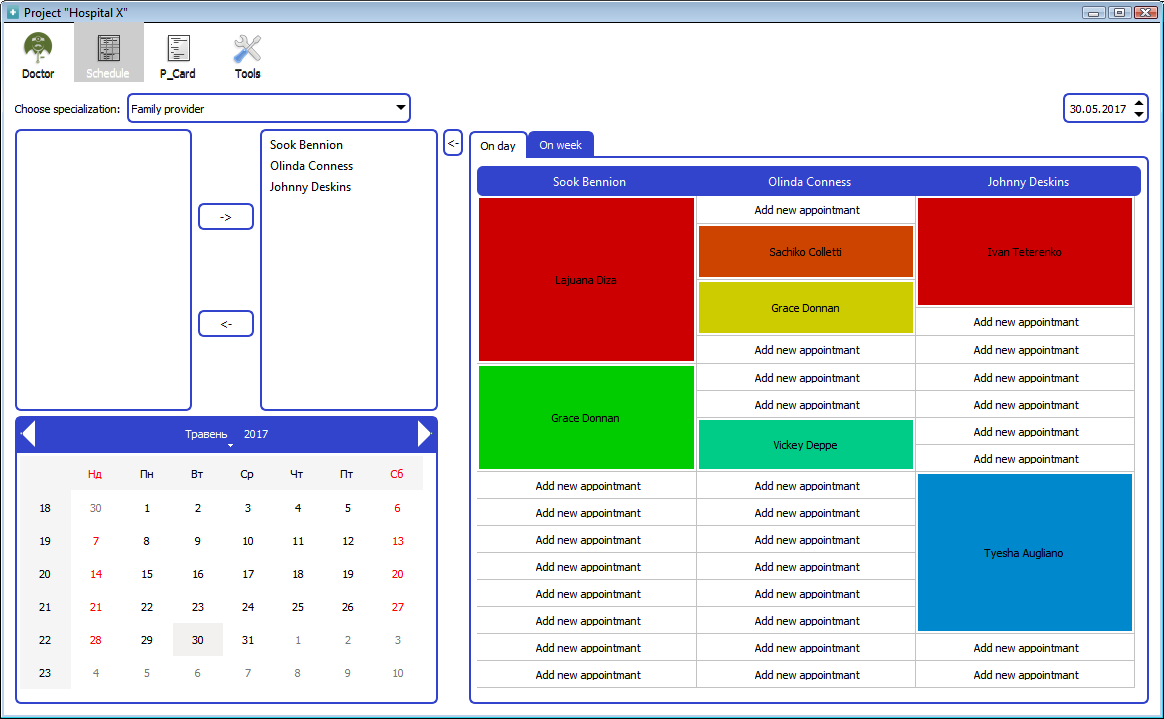


*Рис. 3.20 Створення нового пацієнта в системі*

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

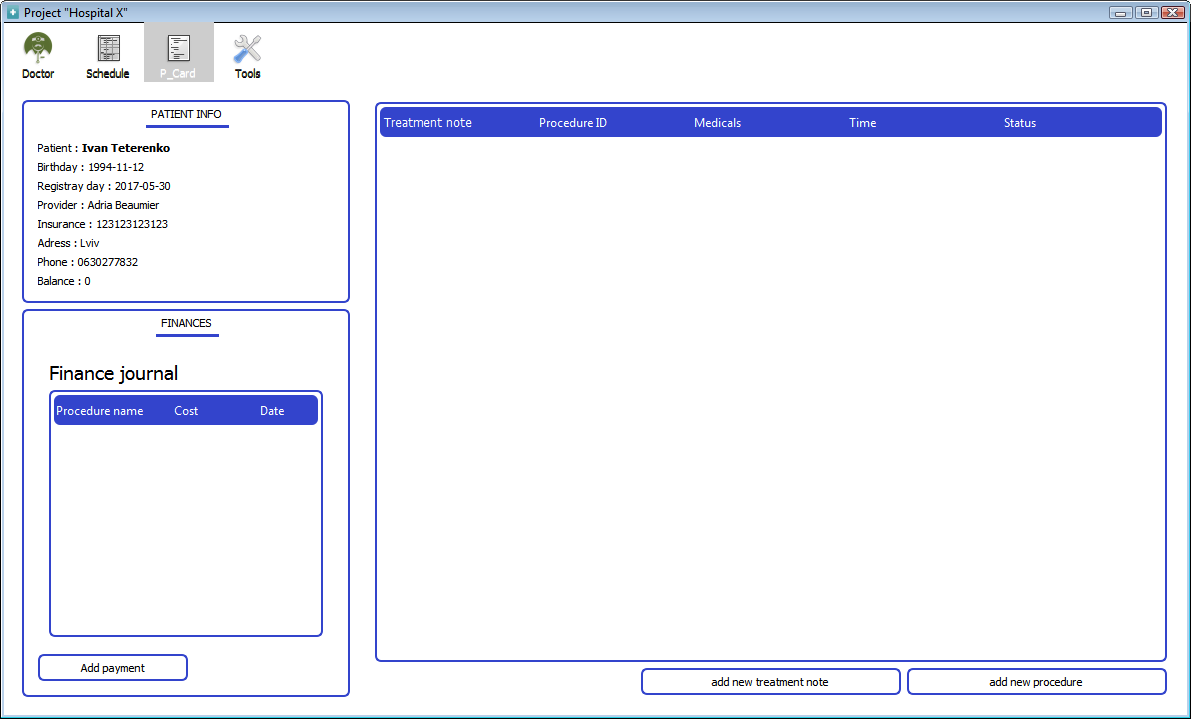


*Рис. 3.21 Призначення прийому*

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. 

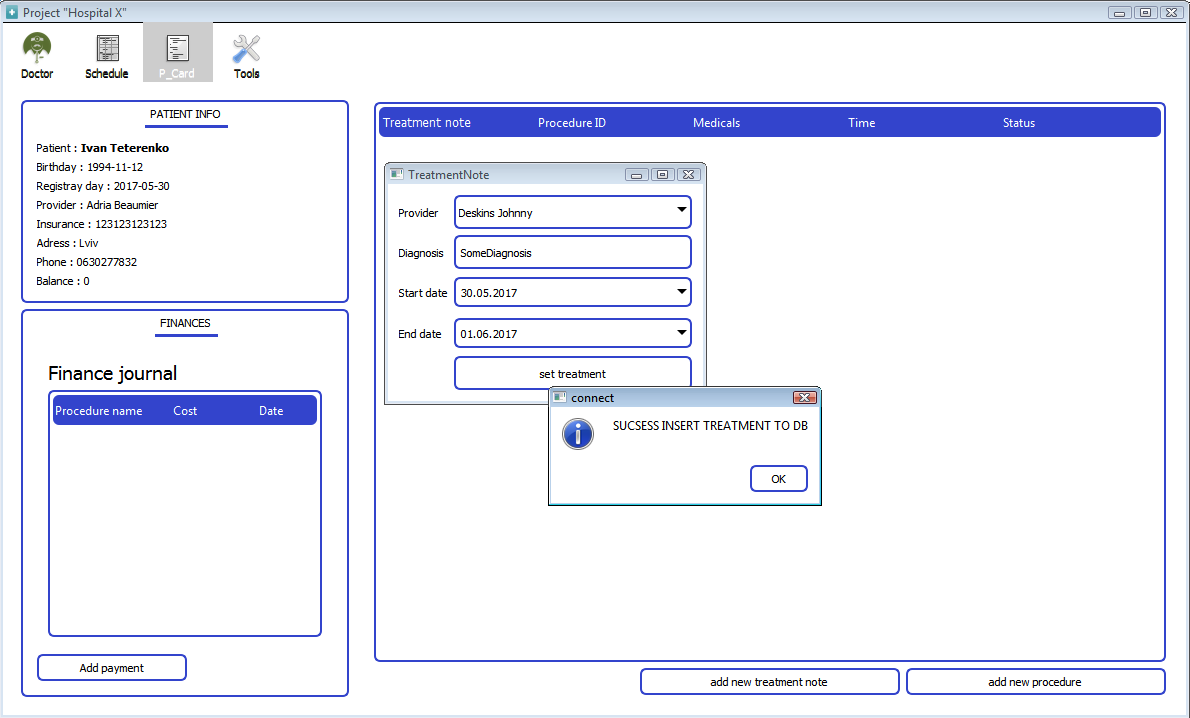
*Рис. 3.22 Призначені прийоми до лікаря*

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



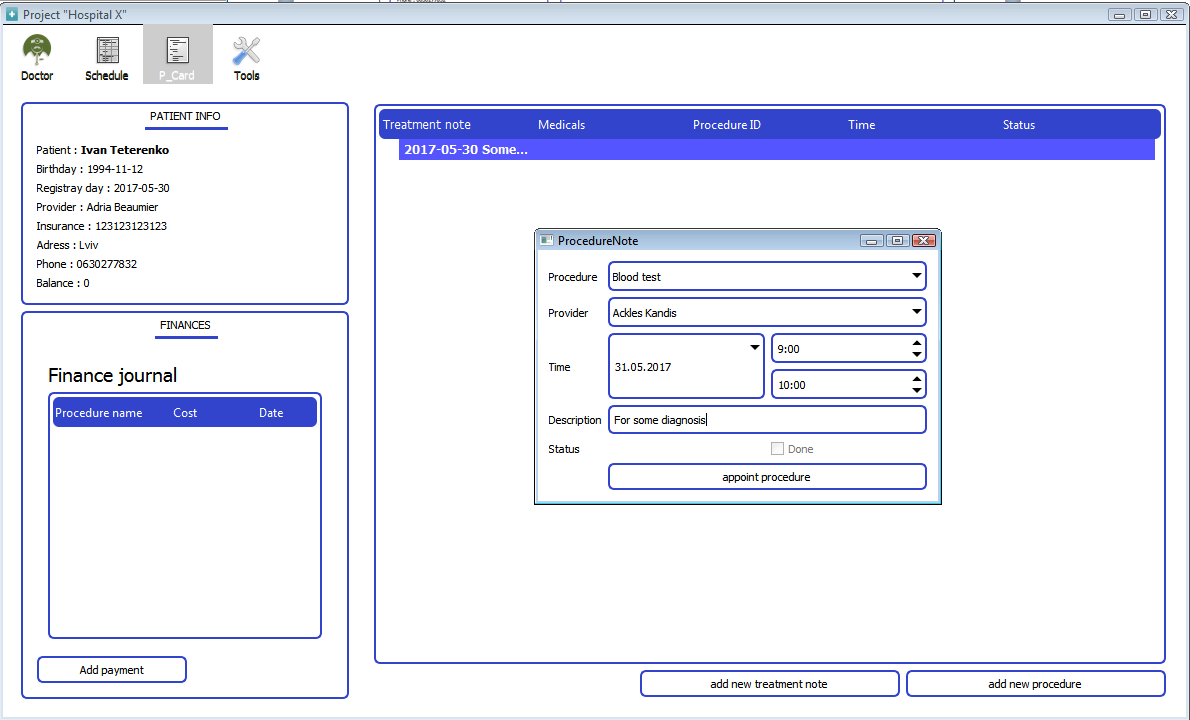
*Рис. 3.23 Сторінка пацієнта Івана Тетеренка*

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

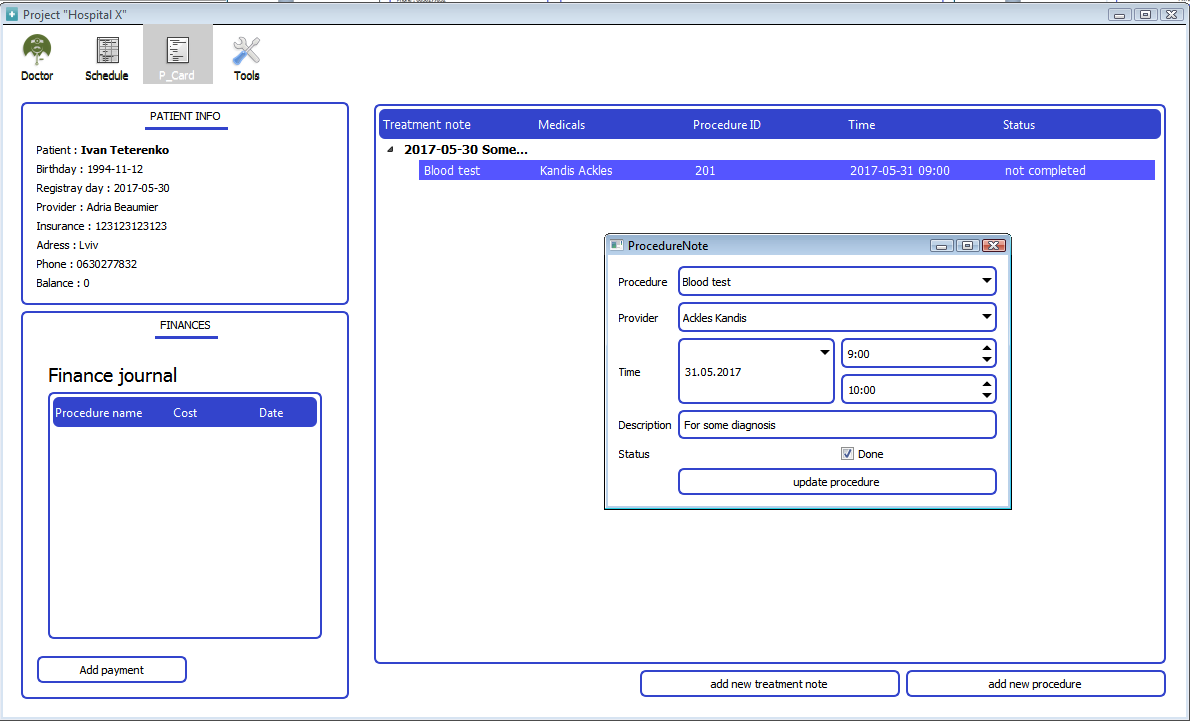


*Рис. 3.24 Створення запису про лікування*

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

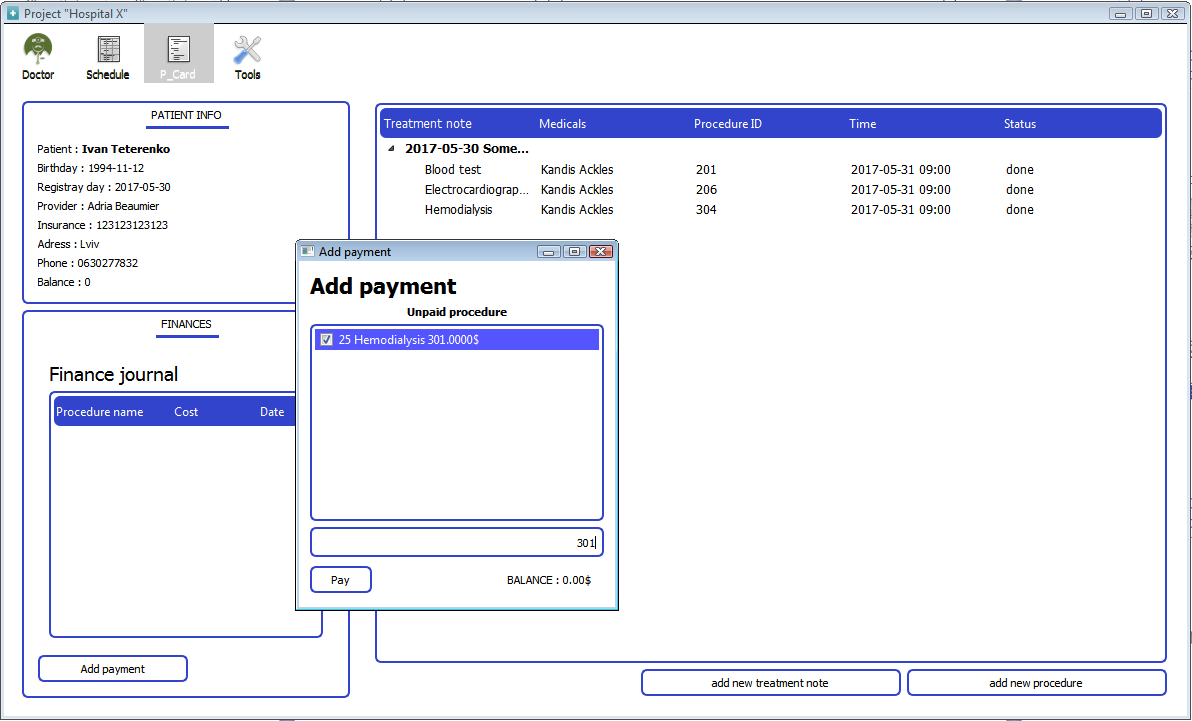


*Рис. 3.25 Створення процедури*

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

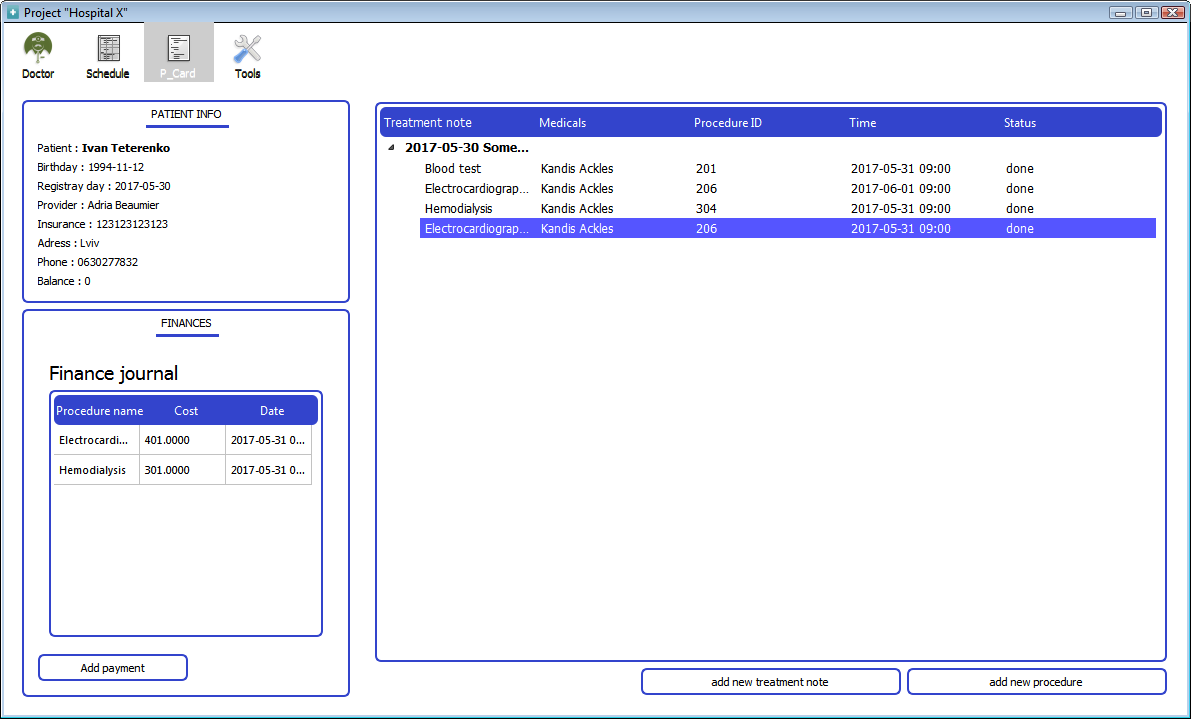
*Рис. 3.26 Налаштування процедури*

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*

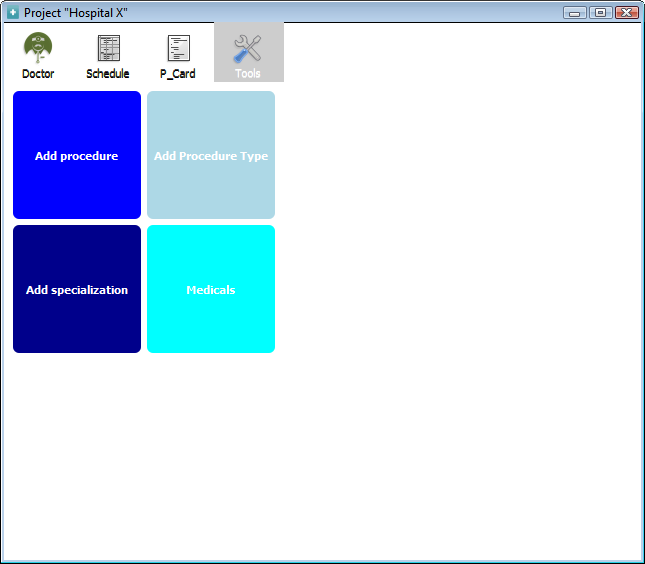
І в результаті в фінансовому журналі ми отримуємо список оплачених процедур:



*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.

# РОЗДІЛ 4. ЕКОНОМІЧНА ЧАСТИНА

# 4.1. Економічна характеристика програмного продукту

У наш час багато рутинної роботи автоматизовується за допомогою комп’ютерних технологій, що дає змогу зосередитись на більш важливих питаннях. Цей процес не минув і такої області, як медицина. Тому метою даної роботи є розробка системи обліку пацієнтів у медичних закладах. Основні завдання, які необхідно досягнути це:

* Простота системи. Так як більшість медичних працівників не мають хороших навиків при роботі з комп’ютером, дана система має бути максимально спрощена без втрати функціональності.
* Зрозумілий та зручний інтерфейс. При першому запуску програми новий користувач має інтуїтивно вже розуміти приблизно як йому працювати з даною системою.
* Швидкість виконання заданих функцій. Також дуже суттєвою проблемою це швидкість обробки даних.
* Мале використання ресурсів системи. Система має працювати на сучасних низькобюджетних комп’ютерах.

# 4.2. Інформаційне забезпечення та формування гіпотез щодо потреби проведення дослідження

Найбільш доцільним пристроєм для використання даної системи є персональний комп’ютер. Доцільно буде використовувати найбільш поширену операційну систему «Windows», так як більшість користувачів працювали саме з нею. Сьогодні на ринку існує безліч хороших систем. Але вони мають як свої переваги, так і свої недоліки.

Очікується, що проект принесе хороший продукт, так як ринок зараз потребує системи, які заощаджують час на рутинних операціях. Дана система орієнтована на вузьке коло спеціалістів – це медичні установи та медичні навчальні інститути. Перевагою даної системи в порівнянні над іншими є її продуктивність та легкість інтерфейсу.

# 4.3. Оцінювання та аналіз факторів зовнішнього та внутрішнього середовищ.

У даному пункті розглядатиметься оцінка та аналіз факторів зовнішнього і внутрішнього середовищ групою експертів.

Фактори зовнішні оцінюються за шкалою [-5;5], при цьому межі шкали відображають максимальний негативний та позитивний вплив факторів на організацію, 0 демонструє, що фактор впливає на організацію нейтрально. [21]

Фактори внутрішні оцінюються за шкалою [0;5], при цьому 0 демонструє нерозвиненість, відсутність чи катастрофічний стан фактору внутрішнього середовища, оцінка 5 демонструє високий рівень розвитку даного фактору.

Сума вагомостей всіх факторів становить одиницю, тобто рівень вагомості для кожного фактору визначається за допомогою коефіцієнтів. Результати експертних оцінок впливу факторів зовнішнього середовища наведено у таблиці 4.1. [21]

*Табл. 4.1*

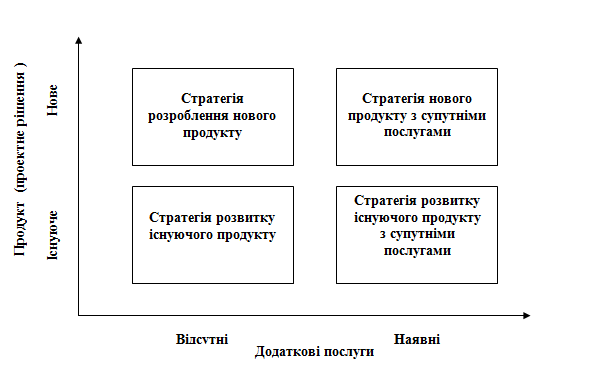
**Результати експертного оцінювання впливу факторів зовнішнього та внутрішнього середовищ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Фактори** | **Середня експертна оцінка, бали** | **Середня вагомість факторів** | **Зважений рівень впливу, бали** |
| 1 | 2 | 3 | 4 |
| *Фактори зовнішнього середовища* |  | | |
| Споживачі | 3 | 0,11 | 0,33 |
| Постачальники | 1 | 0,1 | 0,1 |
| Конкуренти | -4 | 0,1 | -0,4 |
| Державні органи влади | 3 | 0,05 | 0,15 |
| Інфраструктура | 2 | 0,06 | 0,12 |
| Законодавчі акти | 2 | 0,1 | 0,2 |
| Організації-сусіди | -2 | 0,05 | -0,1 |
| Система економічних відносин в державі | 2 | 0,06 | 0,12 |
| Профспілки, партії та інші громадські організації | 1 | 0,01 | 0,01 |
| Міжнародні події | 4 | 0,01 | 0,04 |
| Міжнародне оточення | 2 | 0,03 | 0,06 |
| Науково-технічний прогрес | 1 | 0,07 | 0,07 |
| Політичні обставини | 1 | 0,06 | 0,06 |
| Соціально-культурні обставини | 3 | 0,05 | 0,15 |
| Рівень техніки та технологій | 1 | 0,04 | 0,04 |
| Особливості міжнародних економічних відносин | 3 | 0,02 | 0,06 |
| Стан економіки | 0 | 0,08 | 0 |
| Загальна сума |  | 1 | 1,01 |
| *Фактори внутрішнього середовища* |  | | |
| Цілі | 3 | 0,15 | 0,45 |
| Структура | 2 | 0,13 | 0,26 |
| Завдання | 2 | 0,06 | 0,12 |
| Технологія | 2 | 0,2 | 0,4 |
| Працівники | 1 | 0,22 | 0,22 |
| Ресурси | 1 | 0,24 | 0,24 |
| Загальна сума |  | 1 | 1,69 |

Як видно з табл. 4.1 набільш позитивним фактором зовнішнього середовища є споживачі, так як саме споживачі вказують напрямок розвитку проекту, а найбільш негативним є конкуренти, тому що їм не вигідна поява дешевшого продукту з такою ж якістю, бо їм прийдеться знову ж таки вкладати ресурси для покращення вже існуючого продукту. Внутрішні фактори мають значний вплив на успіх проекту, а це цілі без яких проект не буде розроблений правильно.

# 4.4. Формування стратегічних альтернатив

*Перша група стратегічних альтернатив*. Критеріями поділу альтернативних стратегій розвитку є існуючий продукт (програмне забезпечення) та новий, а також супутні послуги. [21]

****

*Рис.4.1. Стратегічні альтернативи*

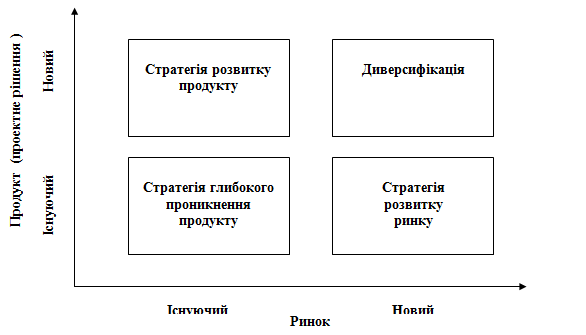
**Стратегія розроблення нового продукту (проектного рішення)** характеризується створенням абсолютно нового програмного забезпечення, яке дає змогу вирішити новоутворені потреби людини, суспільства, економіки тощо.[21]

**Стратегія розвитку існуючого продукту (проектного рішення)** означає модифікацію програмного забезпечення, його якісних характеристик.[21]

**Стратегія розвитку існуючого продукту (проектного рішення) з супутніми послугами** означає пропонування на ринку модифікованого програмного забезпечення із додатковими послугами (встановлення, супроводження, коригування, адаптування до специфіки конкретного підприємства тощо). [21]

**Стратегія нового продукту (проектного рішення) з супутніми послугами** означає розроблення нового програмного забезпечення та пропонування при його експлуатації додаткових послуг.

Друга група стратегічних альтернатив. Критеріями поділу альтернативних стратегій розвитку є існуючий ринок та продукт, новий ринок та продукт.

****

*Рис.4.2. Стратегічні альтернативи*

**Глибше проникнення на ринок** полягає в використанні існуючого продукту (проектного рішення) для збільшення частки на існуючому ринку. Якщо фірма володіє достатніми ресурсами та потужностями для виготовлення існуючого продукту, то ця стратегія є найменш ризикованою. Однак, активно зростання на існуючому ринку призведе до зростання конкуренції. Стратегія буде успішною за умови обмежень у ресурсах та потужностях конкурентів або стрімкому розвитку самого ринку. Слід зазначити, що кожен ринок за обсягом має свій ліміт і якщо підприємство прагнутиме розвиватись, то воно повинно використовувати інші запропоновані стратегії.[21]

**Стратегія розвитку ринку** полягає в використанні існуючого продукту (програмного забезпечення) або незначній його модифікації для виходу на новий сегмент ринку, весь ринок або іноземний ринок. Ця стратегія є з вищим рівнем ризику, оскільки необхідно виходити на новий ринок, де можуть бути інші правила гри, вимоги та смаки споживачів тощо.[21]

**Стратегія розвитку продукту** полягає у створенні нового продукту (програмного забезпечення) для існуючого сегменту ринку. Ця стратегія є досить ризиковою, оскільки вимагає створення нового продукту (програмного

забезпечення) для існуючого сегменту споживачів. Однак, якщо ринок починає зменшувати обсяги та існуючий продукт є на етапі зрілості та падіння, тоді доцільно застосовувати стратегію розвитку продукту.[21]

**Стратегія диверсифікації** реалізується шляхом виходу на нові сфери бізнесу. Тобто розширення номенклатури товарів, послуг тощо. [21]

Розглянувши усі стратегії, найкращим рішенням буде стратегія нового продукту з супутніми послугами. А з другої групи альтернатив обрано стратегію розвитку продукту. Причина цього вибору полягає в тому, що зараз існує гостра потреба в інформаційних системах такого напрямку. Також ця система має усунути проблеми ведення обліку, які присутні у програмних продуктах конкурентів.

# 4.5. Бюджетування

Бюджетування є комплексно обґрунтованою системою розрахунку витрат, пов’язаних з виготовленням та реалізацією продукту, яка дає можливість здійснити аналіз витрат та розробити заходи щодо підвищення рентабельності виробництва. На даному етапі необхідно визначити собівартість продукту, який розробляється та економічно обґрунтувати доцільність вибору однієї із стратегій. [21]

*Табл. 4.2*

**Бюджет витрат матеріалів та комплектуючих виробів**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Назва матеріалів та комплектуючих** | **Марка, тип, модель** | **Фактична кількість, шт.** | **Ціна за одиницю, грн.** | **Разом, грн.** |
| Папір формату А4 | Папір офісний А4 Сompass 500 л | 1000 | 0,15 | 150 |
| Папір для заміток | Папір для заміток, білий 90х90х30 мм, непроклеєний | 25 | 0,4 | 10 |
| Маркер | Маркер текстовий 1-5 мм Economix TEXT чорний | 10 | 99,2 | 99,2 |
| Степлер | Степлер №24 AxentWelle-2 | 1 | 29,62 | 29,62 |
| Разом: |  | 1036 | – | 288,82 |

При розробці даної системи були використані такі людські ресурси:

* Менеджер проекту (М);
* Розробник (Р);
* Тестувальник (Т);
* Дизайнер (Д);
* Аналітик (А);

Витрати на оплату праці складають:

* Місячний оклад менеджера проекту становить 3800 грн./міс.
* Місячний оклад розробника становить 4000 грн./міс.
* Місячний оклад тестувальника проекту становить 2500 грн./міс.
* Місячний оклад дизайнера становить 3000 грн./міс.
* Місячний оклад аналітика становить 3500 грн./міс.

Отримаємо такі значення денної ЗП працівників (з розрахунку на 22 робочі дні у місяць):

* = 3800/22 = 172,72 грн.
* = 4000/22 = 181,81 грн.
* = 2500/22 = 113,63 грн.
* = 3000/22 = 136,36 грн.
* = 3500/22 = 159,09 грн.

Розрахунок витрат на оплату праці кожного розраховується за формулою:

З (1)

де – чисельність працівників і-тої спеціальності, чол.;

– час, витрачений працівником і-тої спеціальності, днів;

– денна ЗП працівника і-тої спеціальності, грн.

Використовуючи дану формулу, обчислимо витрати на оплату праці:

* = 1\* 22 \* 172,72 = 3799,84 грн.
* = 1\* 22 \* 181,81 = 3999,82 грн.
* = 1\* 22 \* 113,63 = 2499,86 грн.
* = 1\* 22 \* 136,36 = 2999,92 грн.
* = 1\* 22 \* 159,09 = 3499,98 грн.

Сумарні витрати на оплату праці:

ВСОП = 3799,84 + 3999,82 + 2499,86 + 2999,92 + 3499,98 = 16799,42 грн.

*Табл. 4.3*

**Бюджет витрат на оплату праці**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Посада, спеціальність | Кількість працівників, осіб | Час роботи, дні | Денна ЗП, грн. | Сума витрати на оплату праці, грн. |
| Основна заробітна плата | | | | |
| Менеджер проекту | 1 | 22 | 172,72 | 3799,84 |
| Розробник | 1 | 22 | 181,81 | 3999,82 |
| Тестувальник | 1 | 22 | 113,63 | 2499,86 |
| Дизайнер | 1 | 22 | 136,36 | 2999,92 |
| Аналітик | 1 | 22 | 159,09 | 3499,98 |
| Разом | 5 |  |  | 16799,42 |
| Додаткова заробітна плата | | | | |
| Разом | – | – | – | – |

*Табл. 4.4*

**Бюджет обов’язкових відрахувань та податків**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Посада, спеціальність | Сума основної заробітної плати, грн. | Сума додаткової заробітної плати, грн. | Разом витрат на оплату праці, грн. | Сума військового збору, грн. 1,5% | Сума податку з доходів фізичних осіб\*, грн. 18% |
| Менеджер проекту | 3799,84 | – | 3799,84 | 56,9976 | 683,971 |
| Розробник | 3999,82 | – | 3999,82 | 59,9973 | 719,968 |
| Тестувальник | 2499,86 | – | 2499,86 | 37,4979 | 449,975 |
| Дизайнер | 2999,92 | – | 2999,92 | 44,9988 | 539,986 |
| Аналітик | 3499,98 |  | 3499,98 | 52,4997 | 629,996 |
| Разом | 16799,42 | – | 16799,42 | 251,991 | 3023,9 |

*Табл. 4.5*

**Бюджет загальновиробничих витрат**

|  |  |
| --- | --- |
| **Статті витрат** | **Сума, грн.** |
| *Змінні загальновиробничі витрати, у т.ч.:* |  |
| - заробітна плата допоміжного персоналу; | 400 |
| - витрати на МШП; | 150 |
| - витрати на електроенергію; | 300 |
| Разом змінних витрат: | 850 |
| *Постійні загальновиробничі витрати, у т.ч.:* |  |
| *-* заробітна плата допоміжного персоналу; | 300 |
| - комунальні послуги; | 200 |
| - витрати на оренду; | 250 |
| - витрати на ремонт; | 0 |
| Разом постійних витрат: | 750 |
| *Разом загальновиробничих витрат:* | 1600 |

*Табл. 4.6*

**Бюджет адміністративних витрат та витрат на збут**

|  |  |
| --- | --- |
| **Статті витрат** | **Сума, грн.** |
| *Адміністративні витрати, у т.ч.:* |  |
| - заробітна плата адміністративного персоналу; | 230 |
| - витрати на МШП; | 100 |
| - витрати на сплату податків і зборів; | 270 |
| - знос адміністративного обладнання; | 200 |
| - інші адміністративні витрати; | 100 |
| Разом адміністративних витрат: | 900 |
| *Витрати на збут, у т.ч.:* |  |
| - заробітна плата менеджерів зі збуту; | 400 |
| - витрати на гарантійний ремонт; | 500 |
| - витрати на гарантійне обслуговування; | 250 |
| - витрати на налагодження і експлуатацію; | 250 |
| - витрати на рекламу; | 300 |
| Разом витрат на збут: | 1700 |

*Табл. 4.7*

**Зведений кошторис витрат на розробку проектного рішення (продукту)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Статті витрат** | **Одиниці виміру** | **Фактична кількість, шт.** | **Ціна одиниці, грн.** | **Разом, грн.** |
| Матеріали та комплектуючі вироби | грн. | 4 | – | 288,82 |
| Додаткові витрати | – | 0 | – | 0 |
| Електроенергія на технологічні цілі (від 100 кВт\*год до 600 кВт\*год | кВт \* год | 135 | 0,99 | 133,65 |
| Основна заробітна плата | грн/день | 1 | – | 16799,42 |
| Додаткова заробітна плата | грн/день | 0 | 0,00 | 0,00 |
| Відрахування на соціальне страхування (22% від доходів) | % | – | – | 3695,87 |
| Витрати на утримання й експлуатацію устаткування | грн. | 3 | 20 | 60 |
| Загальновиробничі витрати, у т.ч.: |  |  |  |  |
| - змінні; | грн. | 1 |  | 850 |
| - постійні; | грн. | 1 |  | 750 |
| *Разом виробничих витрат (собівартість продукту)* | грн. | 1 |  | 22577,76 |
| Адміністративні витрати | грн. | 1 |  | 900 |
| Витрати на збут | грн. | 1 |  | 1700 |
| Інші операційні витрати | – | – | – | – |
| *Разом виробничих і операційних витрат:* | грн. | 1 |  | 25177,76 |

Для визначення фінансових результатів необхідно розрахувати собівартість (ціну) дослідження, яке проводиться. Ціна визначається на основі суми виробних і операційних витрат з урахуванням рентабельності виробництва. [21]

Ц = Сб (1 + Р)

де Ц – ціна проведеного дослідження, грн.

Р – рентабельність, %

Сб – собівартість дослідження, грн.

Отже, при рентабельності 53%, ціна проведеного дослідження:

Ц = 22577,76 \* 1,53 = 34543,97 (грн.);

Цз ПДВ = 34543,97 \* 1,2 = 41452,76 (грн.);

Дохід від реалізації продукції = 41452,76 \* 1 = 41452,76 (грн.);

ПДВ = 41452,76/6 = 6908,79 (грн.);

Чистий дохід від реалізації продукції = 41452,76 - 6908,79= 34543,97 (грн.);

Собівартість реалізованої продукції = 22577,76 \* 1 = 22577,76 (грн.);

Валовий прибуток = 34543,97 - 22577,76 = 11966,21 (грн.);

Фінансовий результат від операційної діяльності = 11966,21 – 900 – 1700= 9366,21 (грн.);

Податок на прибуток = 9366,21 \* 0.18 = 1685,92 (грн.);

Чистий прибуток (збиток) = 9366,21 - 1685,92 = 7680,29 (грн.);

*Табл. 4.8*

**Бюджет фінансових результатів**

|  |  |
| --- | --- |
| **Показники** | **Сума, грн.** |
| Дохід від реалізації продукції | 41452,76 |
| Податок на додану вартість | 6908,79 |
| Чистий дохід від реалізації продукції | 34543,97 |
| Собівартість реалізованої продукції | 22577,76 |
| Валовий прибуток | 11966,21 |
| Операційні витрати: |  |
| - адміністративні витрати: | 900 |
| - витрати на збут; | 1700 |
| - інші операційні витрати; | 0 |
| Фінансовий результат від операційної діяльності | 9366,21 |
| Податок на прибуток | 1685,92 |
| Чистий прибуток (збиток) | 7680,29 |

# 4.6. Вибір стратегії

Розрахунки вище показують, що система варта розробки, якщо не враховувати обставини непереборної сили. Попит на цей програмний продукт є високим, тому що зараз існує в них гостра необхідність. Тому вибрана стратегія, це розробка нового програмного продукту з додатковими послугами при його використанні та стратегія розвитку продукту для існуючого сегменту ринку.

# Висновок

Найкраща стратегія з існуючих для даного продукту це розробка нового

програмного продукту з супутніми послугами при експлуатації. Так як системи, що вже існують мають ряд недоліків, такі як: довге по часу завантаження, потреба в великій кількості ресурсів, важкий для сприйняття інтерфейс та інші, тому продукт спроектований та розроблений повністю з нуля. Тому для виходу на існуючий сегмент ринку програмного забезпечення для медичних установ обрано стратегію розвитку продукту.

Як бачимо розрахований прибуток вказує на те, що система досить швидко окупиться:

Дохід від реалізації (1 шт.) = 41452.76 (грн.);

Чистий прибуток = 7680.29 (грн.).

Проаналізувавши фактори зовнішнього та внутрішнього середовища можна вказати що набільш позитивний вплив чинять споживачі, а негативний вплив найбільше завдають конкуренти.

Успішна реалізація проекту потребує ретельного вивчення програмного забезпечення конкурентів, визначення їх недоліків, та створення програмного продукту набагато кращого аніж його аналоги.

# 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.