МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Обслуговування комп'ютерних систем і мереж»

Група: 4КС-57

Дипломний проект

здобувача освіти денної форми навчання
КС.57.17.000.ДП

ПЛИСКА
ВАДИМ ПЕТРОВИЧ

м. Одеса
2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: **«Обслуговування комп'ютерних систем і мереж»**

Група: **4КС-57**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту на тему:

Розробка системи обліку даних пацієнтів медичного закладу

Проектний матеріал складається з пояснювальної записки на 98 сторінках та графічного (презентаційного) матеріалу на 15 аркушах (слайдах)

Дипломник _____ (Плиска В.П.)

Керівник _____ (Скорняков В.С.)

Консультанти:

з економічного розділу _____ (Іванченков В.С.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Скорнякова О.В.)

Захист «21» 06 2024 р. Протокол ЕК № 5

Оцінка ЕК відмінно / 75 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Обслуговування комп'ютерних систем і мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"15" 01 2024 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Плиска Вадим Петрович*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка системи обліку даних пацієнтів медичного закладу*

затверджена наказом по коледжу від "2" 1 2023 р. № 240-АГ-ОД

2. Термін здачі закінченого проекту *10.06.2024*

3. Вихідні данні до проекту

1. Статистичні дані про роботу з обліку пацієнтів медичних закладів

2. Забезпечити високу швидкість роботи системи та здатність витримувати високі навантаження

3. Клієнт-серверна архітектура додатку

4. Програмні засоби розробки – Embarcadero RAD Studio C++ та Microsoft SQL Server

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Огляд існуючих рішень для електронної реєстратури;

Розробка та опис структури програмного забезпечення;

Програмна реалізація електронної реєстратури;

Реалізація візуального інтерфейсу та тестування розробленого застосунку

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Структура клієнт – серверної архітектури. Модель функціонування.

Схема бази даних електронної реєстратури. Схема взаємодії компонент доступу до БД

Схема алгоритму додавання та редагування даних. Схема ієрархічної структури проекту.

Приклади роботи електронної реєстратури.

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Скорняков В.С.*		
Економічний розділ	*Іванченков В.С.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання _____

Керівник *Скорняков В.С.* _____
(підпис)

Завдання прийняв до виконання *Пліска В.П.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Вступ. Постановка мети та задач проектування*	*20.05.2024*	
2	*Загальний опис реалізації моделі*	*23.05.2024*	
3	*Аналіз структурних елементів системи*	*24.05.2024*	
4	*Розробка структури клієнтської частини*	*26.05.2024*	
5	*Розробка структури серверної частини*	*30.05.2024*	
6	*Реалізація бази даних та опис її структури*	*31.05.2024*	
7	*Розробка діаграми зв'язків таблиць БД*	*1.06.2024*	
8	*Розробка графічного інтерфейсу програми*	*3.06.2024*	
9	*Реалізація програми на мові C++ у RAD Studio*	*6.06.2024*	
10	*Підключення до Microsoft SQL Server*	*8.06.2024*	
11	*Випробування програмного застосунку*	*10.06.2024*	
12	*Аналіз результатів, підготовка слайдів презентації*	*11.06.2024*	
13	*Економічні розрахунки та питання з охорони праці*	*12.06.2024*	
14	*Підготовка графічної частини проекту*	*13.06.2024*	
15	*Підготовка проекту до захисту та тестування ПЗ*	*14.06.2024*	

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КС 57. 17 000. 00 ДП	Дипломний проект		
А4			КС 57. 17 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КС 57. 17 000. 00 ДП		
Розробив		Плиска В.П.		10.06	Розробка системи обліку даних пацієнтів медичного закладу	Літ.	Аркуш 4 / Аркушів
Перевірив		Скорняков В.С.		10.06		Н Д П	
Н. контр.		Петрашова В.І.		10.06	ВСП "ОТФК ОНТУ" гр.4КС-57		
Затверд.		Кривченко Ю.В.					

ЗМІСТ

ВСТУП

Використання реєстру медичної установи неможливо без активного застосування інформаційних технологій, які стають все важливішими у забезпеченні взаємодії між усіма суб'єктами системи охорони здоров'я, підвищенні якості медичної допомоги та поліпшенні стану здоров'я населення.

З кожним роком обсяг медичних даних зростає з кожним днем, що призводить до збільшення часу, необхідного на їхнє введення, обробку та аналіз. У медичній галузі цей тренд також спостерігається. На сьогоднішній день існує багато програм для електронних медичних карт, які відрізняються за потужністю, функціональністю та вартістю. Проте більшість існуючих рішень є або дорогими для державних чи муніципальних медичних закладів, або мають обмежений функціонал.

Розроблена комп'ютерна система призначена для компенсації цих недоліків. Вона є частиною електронної системи охорони здоров'я, яка автоматизує облік медичних послуг та управління медичною інформацією через створення, розміщення, оприлюднення та обмін даними в електронному вигляді.

Ця система складається з двох основних компонентів: центрального реєстру медичних закладів, лікарів, пацієнтів та декларацій, який є інструментом реформи фінансування медичних закладів, і медичних інформаційних систем (МІС), які надають програмне забезпечення для адміністрування медичних закладів.

Основними особливостями розробленої системи є можливість вести облік пацієнтів у відділенні лікарні з необхідними показниками та отримувати медичні записи при необхідності. Метою дослідження є розробка системи обліку даних пацієнтів медичного закладу, яке дозволить користувачу вводити та редагувати дані про пацієнтів, формулювати медичну статистичну інформацію та фільтрувати пацієнтів для звітності.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Огляд існуючих рішень для електронної реєстратури. Основна інформація медичної статистики та звітності

Робота лікарів різних спеціальностей завжди вимагає обробки та аналізу статистичних даних. Навички узагальнення та аналізу інформації, зібраної в медичній практиці, дозволяють ефективно розв'язувати клінічні та організаційні завдання.

Медична статистика - це сукупність методів і знань, що застосовуються для аналізу даних про здоров'я населення та функціонування системи охорони здоров'я. Вона розглядає людину як частину соціальної структури, а її результати виходять за межі лише біологічних аспектів, охоплюючи також вплив соціальних факторів на здоров'я.

Дослідження медичної статистики:

- здоров'я всього населення і окремих його груп шляхом вивчення даних про його чисельність і склад, природному русі, фізичному розвитку та ін.;
- розкриває взаємозв'язок показників здоров'я з різними факторами навколишнього середовища;
- дослідження даних про структуру, діяльність та персоналу лікувально-профілактичних, санітарно-протиепідемічних споруд;
- організація і проведення лабораторно-клінічних досліджень з оцінкою ймовірності результатів спостережень.

Таким чином, питання медичної статистики - обґрунтування нормативних і організаційних потреб в наданні певних видів допомоги, визначення закономірностей рівня допомоги. [1]

Медична інформаційна система (МІС) - комплексний програмний продукт розроблений для автоматизації основних процесів, пов'язаних з роботою медичних установ загальної і вузької спеціалізації. Його основна мета полягає у полегшенні електронного документообігу, гнучкому управлінні пацієнтами, оперативному обліку роботи персоналу та контролі організаційних аспектів.

1.1.1 Основні функціональні компоненти МІС

Медичні інформаційні системи (МІС) складаються з різних модулів, що дозволяє налаштувати їх для потреб різних медичних установ і розширювати функціонал шляхом додавання або вилучення модулів. Вони можуть бути організовані у кілька груп:

1. Аналітичні та управлінські компоненти: інструменти для ведення управлінського обліку, аналізу якості та ефективності медичних послуг. Вони дозволяють проаналізувати роботу медичної організації, виявити проблемні моменти і оптимізувати бізнес-процеси.

2. Медичні компоненти: модулі, що стосуються реєстрації пацієнтів, ведення електронних медичних карток, обліку медичних записів та інше.

3. Фінансово-економічні компоненти: інструменти для обліку медикаментів, управління запасами, розрахунку собівартості лікування, аналізу економічної діяльності та інше.

4. Компоненти обміну даними: системи обміну даними між закладами охорони здоров'я та веденням уніфікованих реєстрів та каталогів.

5. Загальнотехнічні компоненти: контроль доступу та захист баз даних, інтеграція з іншими системами та програмами.

1.1.2 Переваги використання МІС

Впровадження МІС позитивно впливає на роботу медичних закладів, забезпечуючи ефективну автоматизацію документообігу, підвищення якості обслуговування та зручності для пацієнтів. Вона також сприяє телемедицині та узгодженій роботі клінік.

1.1.3 Розробка МІС. Стандартний і кастомний софт

Можна вибрати готове програмне рішення або замовити індивідуальну розробку МІС. Вибір залежить від типу закладу та фінансових можливостей. Підходи можуть бути хмарними або стаціонарними. Кожен варіант має свої переваги і недоліки, які варто врахувати перед вибором.

1.2 Огляд медичних інформаційних систем

1.2.1 Загальні відомості про систему "Doctor Eleks"

Doctor Eleks - це комплексне рішення, розроблене компанією Eleks зі Львова, Україна, яке призначене для оптимізації роботи клінік різних типів та розмірів, включаючи як приватні, так і державні заклади. Серед його основних можливостей - підтримка електронної медичної картки, інструменти для редагування документів, особистий кабінет лікаря, модуль реєстратури, робота зі звітністю, фінансами та персоналом.

Плюси системи включають потужний функціонал, можливість обміну даними у форматі HL7 з іншими інформаційними системами, інтеграцію з різними обладнаннями, можливість роботи з DICOM-зображеннями, підтримку імпорту DICOM-зображень та інше. Doctor Eleks також підключений до системи eHealth та рекомендований МОЗ України для використання.

Рисунок 1.1 Зовнішній вигляд сайту "Doctor Eleks"

Щодо недоліків, то навіть якщо вони існують (наприклад, відсутність підтримки електронних напрямків), вони компенсуються іншими можливостями системи. Оплата за використання Doctor Eleks здійснюється за допомогою постійних та тимчасових ліцензій. Послуги з впровадження, інтеграції та

техпідтримки оплачуються окремо, а також вартість використання мобільного додатка, веб-послуг і роботи зі страховими компаніями.

Система "Doctor Eleks" має значний функціонал та хорошу масштабованість, а також базу даних на платформі Microsoft SQL Server, яка має сертифікат Державної служби спеціального зв'язку та захисту інформації України. Це дозволяє використовувати її для розробки захищених рішень. Однак, основним недоліком є висока вартість, що практично унеможливлює використання цієї системи в державних та комунальних медичних організаціях.

1.2.2 Загальні відомості про систему IRISMED

Щодо системи IRISMED, це компанія, яка надає комплексні рішення у сфері діагностичних радіологічних систем та рентгенології. Вони є офіційним дистриб'ютором FUJIFILM в Україні і надають послуги з монтажу та обслуговування радіологічного обладнання, а також підготовки медичного персоналу для його обслуговування. Їхні інженери сервісу пройшли необхідну підготовку в країнах Європи та забезпечують монтаж, гарантійне та післягарантійне обслуговування радіології.

Рисунок 1.2. Головне вікно компанії «IRISMED»

IRISMED надає комплексні послуги з монтажу та обслуговування радіологічного обладнання, підготовки медичного персоналу для догляду за ним. Інженери сервісу IRISMED пройшли необхідну підготовку в країнах Європи та проводять монтаж, гарантійне та післягарантійне обслуговування радіології.

IRISMED також пропонує інноваційне IT-рішення: радіологічну інформаційну систему IRIS. IRIS наповнений функціоналом, який дозволяє клінікам збирати, структуровано зберігати та обробляти інформацію про пацієнтів, а персонал має доступ до централізованого сховища медичних даних. [4]

1.2.3 Загальні відомості про Радіологічну Інформаційну Систему IRIS

Радіологічна Інформаційна Система IRIS (Radiology Information System – RIS), у поєднанні з PACS (Picture Archiving and Communication System) – дозволяє клінікам збирати та структуровано зберігати інформацію про пацієнтів, а персоналу мати доступ до централізованого медичного сховища даних.

Рисунок 1.3. Радіологічна Інформаційна Система IRIS

IRIS може бути інтегрована для взаємодії з передовою системою DICOM для зберігання та передачі зображень PACS, а також з модулем веб-розподілу.

Ця система розроблена для різних медичних підрозділів, таких як рентгенологічні кабінети, кабінети комп'ютерної томографії (КТ), магнітно-резонансні томографії (МРТ) та радіоізотопні лабораторії, і надає повний контроль над діагностичним процесом. Це технологічне рішення адаптоване для українських умов та забезпечує обіг інформації про пацієнтів, формування діагностичних висновків лікарів та обмін медичними записами відповідно до чинного законодавства. IRIS складається з кількох структурних модулів, які працюють на різних рівнях доступу до інформації.

Основні функції IRIS:

- Попереджує часові неспівпадіння в прийомі пацієнтів.
- Дозволяє сортувати пацієнтів по стану, ділянках та типах обстеження.
- Містить антропометричні дані пацієнта, необхідні для правильного виконання радіологічного обстеження.
- Архівує інформацію пацієнта для використання в наступних обстеженнях.
- Дозволяє налаштовувати звітність та відстежувати результати.
- Організовує хід роботи для продуктивного робочого процесу.
- Покращує логістику прийому пацієнтів.
- Спрощує роботу інтуїтивним рольовим інтерфейсом.
- Дозволяє проводити статистичний аналіз.
- Скорочує час роботи реєстратури закладу охорони здоров’я з пацієнтом.
- Скорочує час контакту рентгенлаборанта із лікарем та іншим середнім медичним персоналом.
- Збільшує кількість пацієнтів, обстежених за певний період часу.

1.2.4 Загальні відомості про медичну інформаційну систему «МедІнфоСервіс»

Медична інформаційна система "Медична інформаційна система "МедІнфоСервіс" - це програмний продукт, який автоматизує лікувальні процеси в амбулаторно-поліклінічних та стаціонарних лікувальних закладах.

Вона має акредитацію від Міністерства охорони здоров'я та інтегрована з електронною системою охорони здоров'я "E-Health". Обираючи МІС "МедІнфоСервіс", ви отримуєте повний набір функціоналу для передачі електронних медичних записів до ЕСОЗ, включаючи імпорт даних з "Медичної карти амбулаторного хворого" та "Медичної карти стаціонарного хворого". Це значить, що лікарям не потрібно виконувати подвійну або потрійну роботу - вони просто вносять дані в електронну документацію з можливістю імпорту та друку документів.

МІС "МедІнфоСервіс" успішно впроваджена в більш ніж 100 медичних закладах України різних рівнів та форм власності, що дозволяє більш ніж 3000 лікарям здійснювати електронну документацію та отримувати фінансування. Заклади, які користуються МІС "МедІнфоСервіс", мають успішний досвід передачі інформації до ЕСОЗ та отримують своєчасну оплату за договорами. Команда МІС "МедІнфоСервіс" використовує цей досвід для налаштування контролів та обмежень, щоб запобігти введенню некоректних даних та уникнути помилок під час звітування.

Рисунок 1.4. Загальний вигляд медичної інформаційної системи «МедІнфоСервіс»

Для уникнення помилок при введенні інформації та спрощення роботи медичного персоналу МІС "МедІнфоСервіс" використовує шаблони для заповнення текстових блоків інформації та набори даних при створенні направлень і заповненні Листка лікарських призначень.

Однією з переваг МІС "МедІнфоСервіс" є можливість імпорту даних з "Медичної карти амбулаторного пацієнта" та "Медичної карти стаціонарного хворого" безпосередньо до електронних медичних записів. Це дозволяє лікарям уникнути подвійної або потрійної роботи - їм просто необхідно внести дані в електронну документацію з можливістю подальшого імпорту та друку документів.

МІС "МедІнфоСервіс" - проста у використанні та зрозуміла для медичного персоналу. Вона забезпечує стабільну та безперебійну роботу в ЕСОЗ, що дозволяє отримувати фінансування за надані послуги в повному обсязі.

1.2.5 Загальні відомості про систему EMCiMED

EMCiMED - передова українська медична інформаційна система для медичних установ, приватних клінік і лабораторій. Складається з модулів, що легко можна збирати в потрібній конфігурації для кожного окремого закладу. Основні підтримувані модулі: реєстратура, управління персоналом, управління організацією, поліклініка, стаціонар, лабораторія, управління партнерськими відносинами.

Можна придбати додаткові модулі: облік послуг, управління запасами, архів медичних зображень PACS та інші. За необхідності вони можуть поставлятися в складі пакетів EMCiMED-Поліклініка і EMCiMED-Стаціонар.

Плюси. Можливість вибирати модулі відповідно до вимог організації, гнучке налаштування, потужна функціональна складова. Система захищена завдяки використанню USB-брелоків та шифрування всієї інформації, підтримує інтеграцію з іншими продуктами, наприклад, 1С. Пройшла перевірку і рекомендована до використання МОЗ України.

Рисунок 1.5. Загальний вигляд EMCiMED

Мінуси. Якщо присутні, то істотно не впливають на роботу з програмою. Оплата та вартість. Ліцензія. Ліцензії безстрокові, припускають одноразову оплату на весь період використання. Більш докладно всі умови обговорюються з замовником при оформленні договору поставки.

1.3 Розробка та опис структури програмного забезпечення

Інформаційна система (ІС) представляє собою набір документів, які стосуються конкретних об'єктів із визначеними засобами пошуку інформації. Основною метою створення ІС є організація роботи з потоками інформації та забезпечення самого процесу управління. У фармації створення ІС сприяє збереженню необхідної інформації та її раціональному використанню в практиці та наукових дослідженнях.

У процесі створення будь-якої ІС беруть участь постановник задачі, який представляє інтереси потенційного користувача, і розробник - програміст, який видає кінцеву продукцію – програмний засіб. ІС можуть бути автоматизованими, що означає використання технічних засобів пошуку та обробки інформації, зокрема комп'ютерних технологій.

Автоматизована ІС складається з упорядкованих у певний спосіб даних і комплексу апаратно-програмних засобів для їх зберігання та маніпулювання. Властивості ІС залежать від структури інформації, а неструктурованість даних може обмежити можливості автоматизованої обробки інформації в ІС.

Структура розробленої системи включає клієнтську програму та серверну частину. (рис. 1.6).

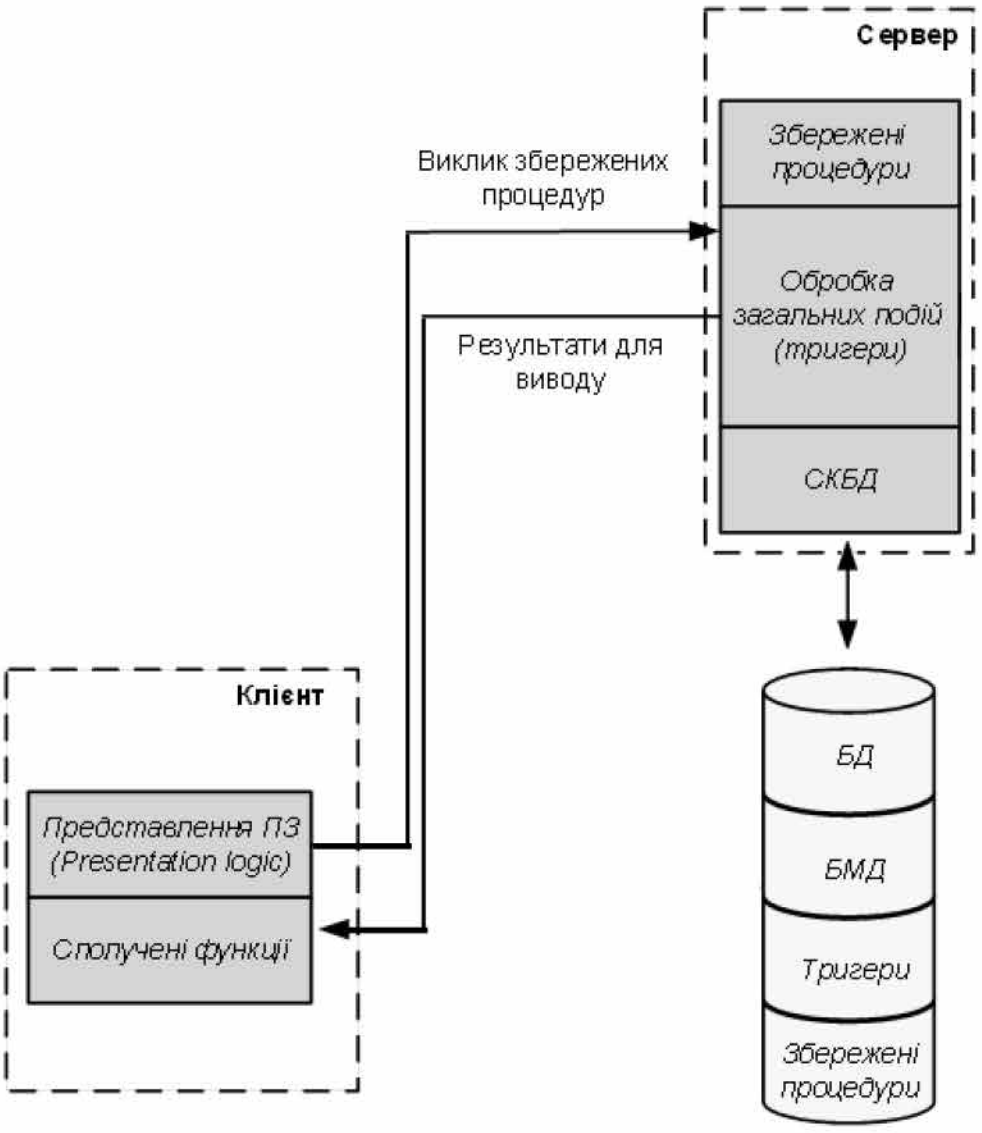


Рисунок 1.6. Модель роботи клієнт-серверної архітектури

Банк даних є різновидом ІС, де реалізовано функції централізованого зберігання та накопичення оброблюваної інформації, організованої в одну або кілька баз даних.

Архітектура мережі визначає основні елементи мережі, її загальну логічну організацію, апаратне та програмне забезпечення, методи кодування, а також принципи роботи та користувацький інтерфейс.

Клієнтська частина системи призначена для взаємодії з користувачем шляхом надання інтерфейсу, який дозволяє переглядати, змінювати та додавати інформацію до бази даних. Це включає в себе різноманітні функції, які дозволяють користувачам взаємодіяти з даними і системою в цілому.

Серверна частина отримує дані та команди від клієнта через мережу, виконує обробку цих даних та команд і відправляє відповіді назад до клієнта. Крім того, важливою функцією сервера є взаємодія з базою даних, що включає створення, читання, редагування та видалення даних з бази даних. Таким чином, серверна частина виконує ключову роль у керуванні даними та забезпеченні їх консистентності і доступності для клієнтської частини системи.

1.3.1 Архітектура клієнт-сервер

Архітектура клієнт-сервер передбачає, що основна частина ресурсів інформаційної мережі сконцентрована у серверах, які обслуговують своїх клієнтів. Ця архітектура визначає два типи компонентів: сервери та клієнти. В системі, про яку йдеться, реалізована саме така архітектура (рис.1.7)

Сервер - це об'єкт, який забезпечує надання послуг іншим об'єктам мережі за їх запитами. Послуга - це процес обслуговування клієнтів.

Сервер виконує завдання клієнтів та керує процесом їх виконання. Після виконання кожного завдання сервер передає результати тому клієнту, який поставив це завдання.

У архітектурі клієнт-сервер сервісна функція описується набором прикладних програм, які забезпечують виконання різних прикладних процесів.

Клієнт - це процес, який викликає службову функцію через певні операції. Це може бути програма або користувач.

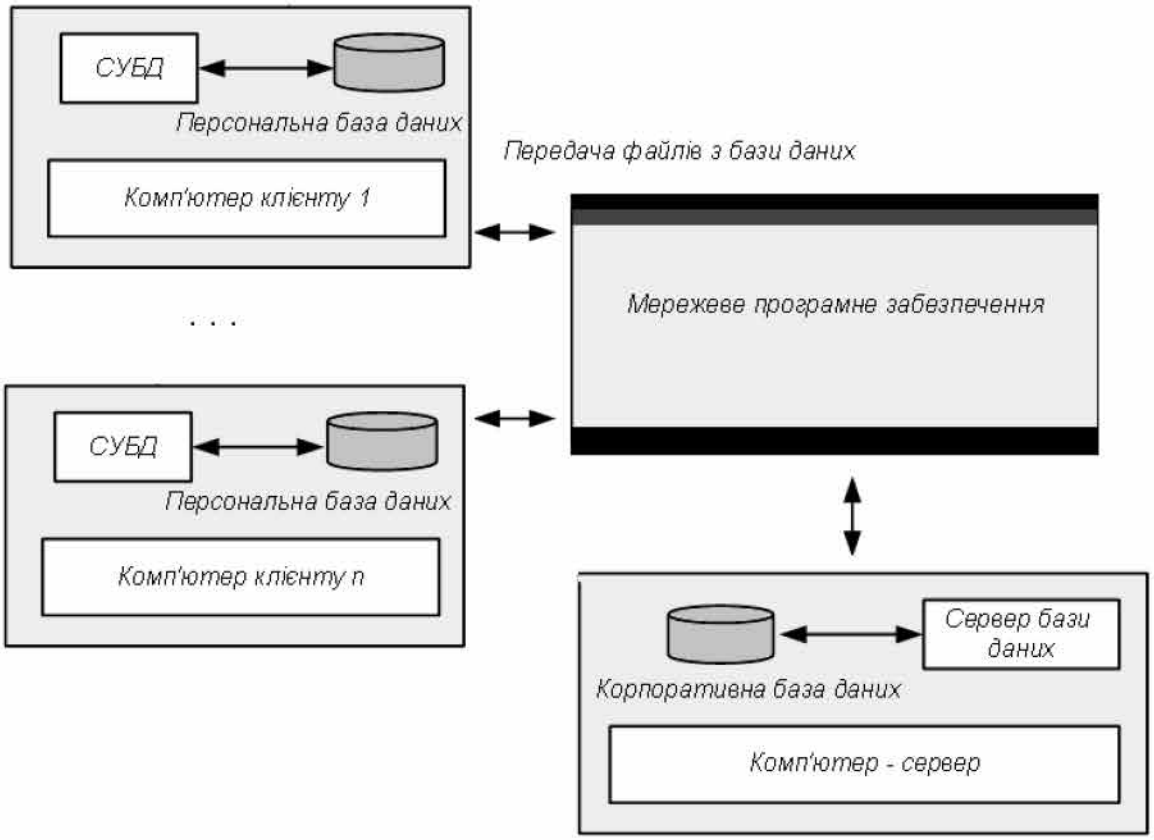


Рисунок 1.7. Структура інформаційной системи клієнт-сервер

Клієнти - це робочі станції, які використовують ресурси сервера та надають зручний інтерфейс користувачам. Призначені для користувача інтерфейси - це процедури взаємодії користувача з системою або мережею.

Клієнт ініціює завдання та використовує доступні йому серверні сервіси. У процесі роботи клієнт здійснює запит на послугу, встановлює сеанс, отримує необхідні результати та повідомляє про завершення роботи.

Сервери та клієнти не залежать один від одного. Клієнти також працюють паралельно та незалежно один від одного. Часто сервер приймає запити від різних клієнтів, а клієнт може звертатися до різних серверів по черзі. Клієнти мають інформацію про сервери, з якими вони можуть взаємодіяти, але не повинні мати інформації про інших клієнтів, які працюють паралельно з серверами.

Клієнтські та серверні процеси можуть розташовуватися на різних комп'ютерах, підключених до мережі, хоча вони можуть бути і на одному

комп'ютері. Сервер може надавати послуги більш ніж одному клієнту, а клієнт може звертатися за послугами до кількох мережевих серверів незалежно від їх розташування. Мережа надає засоби зв'язку для клієнтів та серверів.

Сервери можуть надавати різноманітні послуги, включаючи управління файлами та принтерами, електронну пошту, доступ до Інтернету, а також виступати у ролі веб-серверів або серверів баз даних.

У даній роботі було вирішено використовувати архітектуру клієнт-сервер, оскільки вона оптимально відповідає поставленим вимогам. Цей архітектурний шаблон забезпечує простоту реалізації багатьох режимів роботи, коли до одного сервера звертається кілька користувачів, що допомагає уникнути помилок та заблокувати одночасний доступ декількох користувачів до одного рядка в базі даних.

Клієнтські та серверні процеси не залежать один від одного. Залежно від ступеня розподілу процесів між клієнтом і сервером, їх можна класифікувати як слабкі або сильні. Слабкий клієнт виконує мінімальну обробку на стороні клієнта, тоді як сильний клієнт виконує значну частину обробки даних. Сильний сервер несе основне навантаження по обробці даних, тоді як слабкий сервер має невелике навантаження. Система мейнфрейма є прикладом найсильнішого сервера та найслабшого клієнта.

Системи клієнт-сервер можуть мати два рівні: дворівневу та трирівневу моделі. У дворівневій моделі клієнт напряму запитує послуги від сервера, тоді як у трирівневій моделі запити обробляються проміжними серверами, що координують виконання клієнтських запитів разом із своїми підлеглими серверами.

Під час розробки вибрана дворівнева модель клієнт-серверної архітектури, оскільки, на відміну від трирівневої, вона не потребує додаткового обслуговування. У цій системі реалізується тонка модель клієнта, де вся логіка програми зосереджена на сервері, який представлений системою управління базами даних.

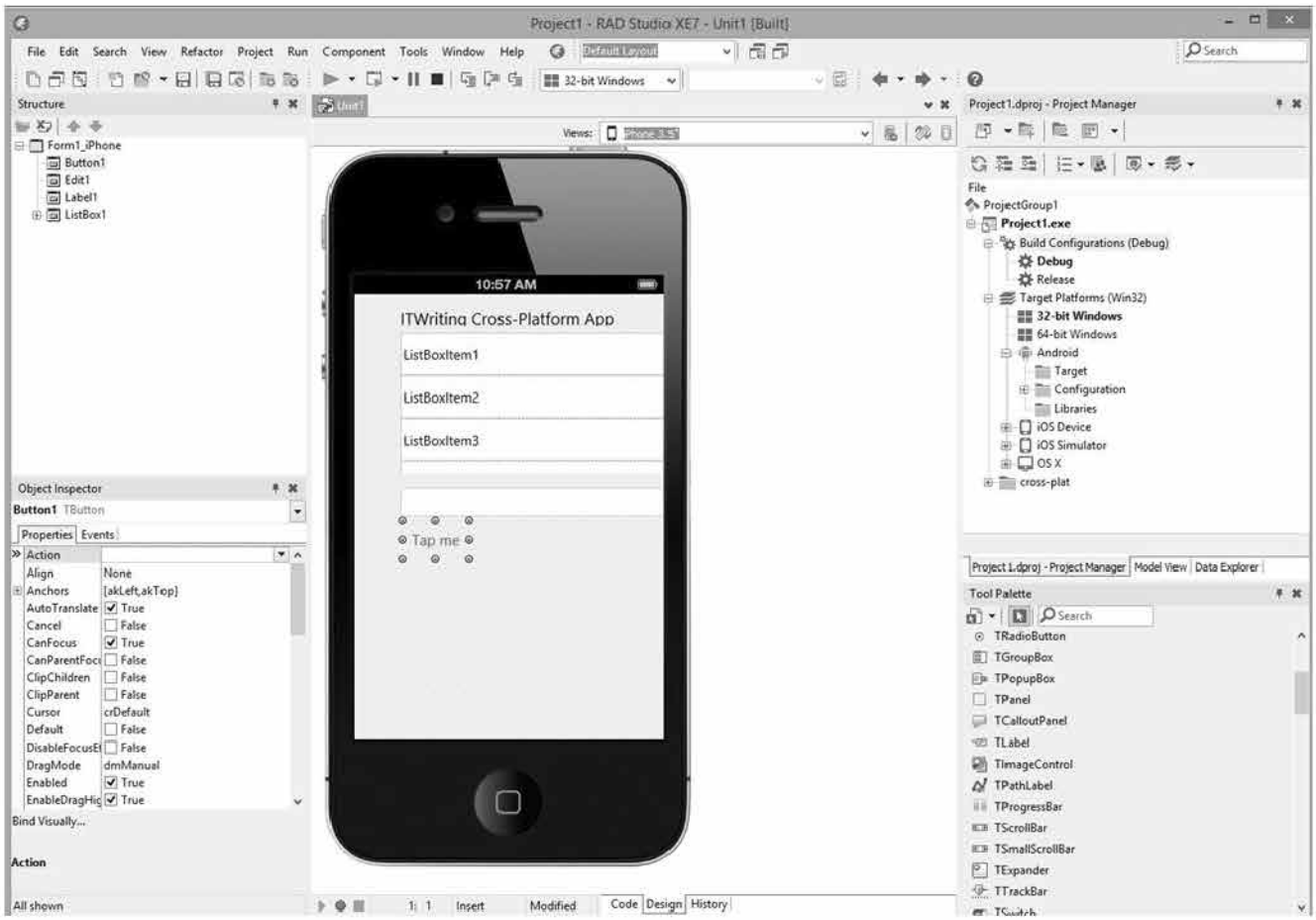


Рисунок 1.8. Зовнішній вигляд вікна проекту Embarcadero C++Builder® XE7

1.3.2 Реалізація клієнтської частини системи

Клієнтська частина призначена для прямої взаємодії з користувачем і якісного відображення всієї інформації, що надходить з сервера. Клієнт зв'язується з сервером через механізм запитів. Клієнтська частина реалізує користувальницький інтерфейс, формує запити до сервера і обробляє відповіді.

Для реалізації клієнтської частини системи були обрані C ++ мови програмування і середовища розробки програмного забезпечення C++Builder® XE7

Embarcadero C++Builder® XE7 — це рішення на основі C ++ для швидкого створення високопродуктивних взаємодіючих додатків для Windows, OS X, iOS, Android, гаджетів і носяться пристроїв. Створюйте високопродуктивні, компілюють нативні додатки для простого підключення до

корпоративних даних, хмарним службам, пристроїв, датчиків і гаджетам, використовуючи всю міць стандартного мови C ++.

C ++ Builder XE7 це - новітня версія популярного рішення для розробки додатків не тільки для Windows, OS X, iOS і Android, а й тепер для "Internet of Things". Даний реліз включає багато нового: нові можливості для VCL, бібліотека паралельних обчислень, поліпшена бібліотека FireMonkey для розробки кроссплатформенних додатків, корпоративні мобільні сервіси (EMS) і засоби роботи з "Internet of Things" завдяки новій можливості взаємодії через Bluetooth.

C ++ Builder - це програмний продукт, інструмент для швидкої розробки надбудов (RAD), інтегроване середовище розробки (IDE), система, яка використовується розробниками для створення [11]

C ++ Builder був спочатку розроблений Borland Software, а потім його підрозділом CodeGear , яке тепер належить Embarcadero Technologies.

C ++ Builder об'єднує набір об'єктних бібліотек (STL, VCL, CLX, MFC і ін.), Компілятор, відладчик (відладчик), редактор коду і багато інших комп'ютери. Цикл розробки аналогічний Delphi. Більшість компонентів, розроблених в Delphi, можна використовувати в C ++ Builder без змін, але протилежне твердження не так.

C ++ Builder містить інструменти, які використовують метод Drag-And-Drop, щоб зробити розробку дійсно наочною і спростити програмування завдяки вбудованому редактору інтерфейсу. [12]

Основні переваги середовища розробки Embarcadero C++Builder® XE7:

- Зручний редактор форми та компонентів.
- Інтерфейс і підтримка редактор Unicode (UTF-8), що дозволяє уникнути помилок з використанням національних символів.
- Введено власний формат управління пакетами.

- Перехід до нової бібліотеці віджетів може легко переключитися на нову бібліотеку віджетів з допомогою функції автоматичної компіляції.

- Поширеність серед програмістів, завдяки чому завжди висока ймовірність знайти готове рішення проблеми, що виникла в процесі розробки.

- C ++ Builder може використовувати не тільки бібліотеку компонентів, а й код, написаний на C ++.

- C ++ Builder може використовувати візуальні та невизуальні компоненти, які були створені у великій кількості за останні кілька років.

- RAD Studio включає в себе сотню компонентів , які забезпечують все необхідне: від створення користувацьких інтерфейсів для підключення до баз даних. Це пришвидшує та спрощує побудову бізнес-додатків для настільних комп'ютерів та мобільних платформ. На додаток до бібліотеки візуальних компонентів VCL та FireMonkey доступний широкий спектр інструментів та компонентів C ++ Builder.

За допомогою GetIt Package Manager розробники можуть значно розширити функціональність своїх додатків, завантажуючи і інтегруючи нові компоненти, не залишаючи студії. Можна запросити спеціальні пакети для конкретного проекту. Це означає, що всі відповідні бібліотеки і елементи управління будуть встановлені при відкритті проекту.

Розширене середовище підтримки сторонніх інструментів та компонентів, що розширює можливості середовища розробки RAD Studio. [13]

Недоліки середовища розробки Embarcadero C++Builder® XE7:

- Щодо великий розмір файлу зі стандартними настройками компілятора.

- Наявність великої кількості властивостей компонентів, більшість з яких можуть не використовуватися при розробці цього проекту.

- Ви можете переглядати значення змінних і полів тільки за допомогою відладчика. Це не дозволяє переглядати властивості об'єктів під час налаштування.

При написанні дипломної роботи і проектуванні архітектури системи ця система була обрана через її переваг і простоти.

1.3.3 Реалізація серверної частини системи

Серверна частина програми відповідає за обробку запитів від клієнтів та передачу результатів обробки назад клієнтській програмі. У цьому програмному продукті серверна частина є базою даних. База даних - це набір даних, організований відповідно до концептуальної структури, яка описує характеристики даних і відносини між ними та один одним.

Інформаційна система (ІС) включає специфічний фрагмент, що відноситься до предметної області, і буде виконувати автоматизовану обробку. Дані про зовнішній світ представлені в ІС у вигляді даних, що обмежує можливості осмисленої інтерпретації інформації та уточнює семантику її подання. Набір цих даних, їх взаємозв'язок та операції з ними створюють інформаційні та функціональні моделі програмного забезпечення, які описують його стан з певною точністю. Ці моделі є вихідними даними для процесу проектування бази даних.

Існує багато типів баз даних, які різняться за різними критеріями, такими як типи зв'язків між компонентами та типи представлення даних.

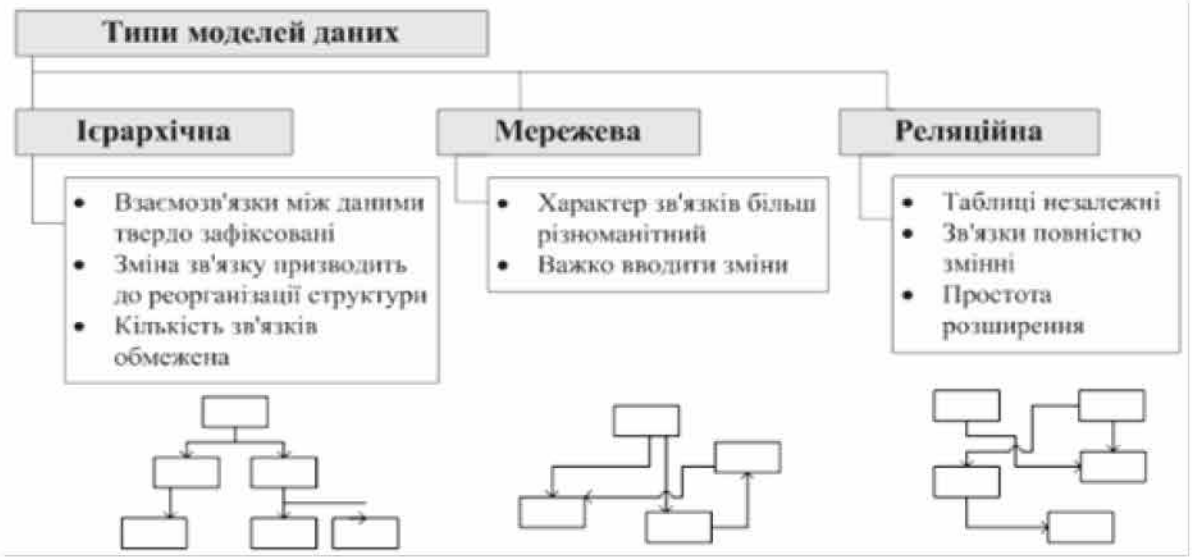


Рисунок 1.9. Типи моделей баз даних

Реляційна модель даних реалізована в програмах. Реляційна модель даних - це логічна модель даних, прикладна теорія побудови баз даних, яка є доповненням до завдань обробки даних для таких розділів математики, як математика. На цій моделі побудовані реляційні бази даних. Модель реляційної бази даних включає наступні компоненти: [14]

- Структурний аспект - дані в базі даних зберігаються як набір відносин.
- Цінності і спектр - все відносини відповідають певним умовам цілісності. Реляційна модель даних підтримує декларативні обмеження на цілісність рівня домену (тип даних), рівень відносин і рівень бази даних.
- Модель обробки і спектральних даних підтримує операторів маніпулювання орієнтацією. Нормалізація також включена в реляційну модель даних.

Термін «реляційний» означає, що ця теорія заснована на математичному понятті «відносини». Слово «стіл» часто використовується як неформальний синонім терміну « відносини ». Важливо пам'ятати, що «таблиця» - це вільне поняття і часто означає не «відносини» як абстрактне поняття, а візуальне уявлення відносин на папері. Неправильне і вільне використання терміна «таблиця» замість терміна «коефіцієнт» часто призводить до нерозуміння. [15]

Реляційна модель відображає об'єкти і відносини між ними у вигляді таблиць. Кожна таблиця повинна відповідати певним вимогам:

- будь-який елемент таблиці є одним елементом даних, тому значення в таблиці повинні бути унікальними;
- відносини представлені як об'єкти;
- таблиця показує один об'єкт і складається із записів (кортежів) і полів (атрибутів);
- кожне поле таблиці має унікальну назву;
- поле повинно містити дані одного типу;
- кожна таблиця повинна містити первинний ключ - поле або кілька полів;
- в таблиці не може бути двох однорагових кортежів.

Переваги реляційної моделі в тому, що вона націлена на обробку документа в цілому, а не його окремих частин. Це не вимагає від користувачів знань програмування. Для використання реляційної моделі досить навичок роботи з комп'ютером і знання основ інформаційних технологій. Отже, реляційна модель проста для розуміння і використання.

Недоліками реляційної моделі є відносно низька швидкість доступу до даних і використання великих обсягів пам'яті на носії. [16]

У цьому програмному продукті база даних написана на мові SQL [17] і управляється за допомогою Microsoft SQL Server 2012. Microsoft SQL Server - це комерційна система управління базами даних. [18]

MS SQL Server - це платформа, спрямована на вирішення важливих завдань на рівні підприємства. Вона відрізняється високою доступністю, підвищеною продуктивністю і безпекою. Ця рішення має гнучкість масштабуватися, є повністю реляційним, високошвидкісним сервером, здатним обробляти великі обсяги даних для додатків клієнт-сервер. Рекордна продуктивність MS SQL Server забезпечується новими технологіями роботи з пам'яттю, що допомагають підприємствам прискорювати бізнес і реалізовувати нові сценарії роботи. SQL Server також підтримує використання нових гібридних хмарних рішень і можливості нових хмарних обчислень. Розширені функції безпеки в поєднанні з вбудованими простими у використанні інструментами і контрольованим доступом до даних дозволяють організаціям відповідати суворим вимогам політик.

Переваги платформи:

- рекордно висока продуктивність;
- швидке отримання результатів аналізу як в локальній системі, так і в хмарному середовищі;
- розширені функції безпеки;
- платформа для гібридного хмари.

Основні особливості платформи:

- Єдина подача даних. Великомасштабна платформа для інтеграції корпоративної інформації включає інструменти інтелектуального аналізу даних , перетворений ия і завантаження (ETL) . MS SQL Server забезпечує доступність і інтеграцію будь-якого різнорідного джерела даних, а також отримання даних з будь-якого стороннього джерела, включаючи Microsoft , DB, Frame, Framework , Oracle, ODBC, Teradata . Платформа дозволяє інтегрувати дані бізнес- процесів в Microsoft BizTalk Server для систем SAP, ERP і CRM , веб сервісів і додатків для мейнфреймів і фіксувати дані, що надходять в реальному житті. ETL - інструменти MS SQL Server гарантують кращу в своєму класі швидкість, а використання високопараллельних пакетів і оптимізованих потокових пулів може підвищити продуктивність.

- Бізнес-аналітика. SQL Server допомагає створити комплексне рішення для аналізу даних на рівні підприємства. Користувачі мають можливість проактивно аналізувати і інтерактивно переглядати агреговані дані з різних точок зору.

- Звітність. MS SQL Server має широкий спектр можливостей для створення високоякісних звітів для друку, що дозволяє використовувати веб-браузери і браузери для перегляду і візуалізації. Рішення спрощує механізми взаємодії та обміну даними, самостійно створює звіти в різних форматах, візуалізує дані. MS SQL Server гарантує високий рівень керованості і масштабованості звітів як локально, так і в хмарі, дозволяє приймати рішення в реальному часі.

- Мова запитів - Transact-SQL, розроблений спільно Microsoft та Sybase. Transact-SQL - це реалізація стандарту ANSI / ISO для мови структурованих запитів SQL з розширеннями. Він використовується як для малих і середніх баз даних, так і для великих баз даних в масштабі підприємства. Microsoft SQL Server вже була успішно конкурувати з іншими системами управління базами даних для багатьох років . [21]

Transact-SQL (скорочено T-SQL) - це реалізація SQL-92 (стандарт ISO для

SQL) з багатьма розширеннями. T-SQL дозволяє використовувати додатковий зберігається синтаксис і забезпечує підтримку транзакцій (взаємодія бази даних з керуючим додатком). MS SQL Server і Sybase ASE для зв'язку з мережею використовують протокол мережевого рівня, званий потоком даних таблиці (TDS).

MS SQL Server також підтримує Open Database Connectivity (ODBC). MS SQL Server забезпечує можливість підключення користувачів, що використовують послуги веб - сервера, які використовують в SOAP - протокол . Це дозволяє клієнтам, відмінним від Windows, підключатися до SQL Server на міжплатформеній основі.

SQL Server підтримує дзеркальне відображення бази даних і кластеризації. Кластер SQL-серверів - це набір однаково налаштованих серверів; ця схема допомагає розподілити навантаження між декількома серверами. Всі сервери мають одне віртуальне ім'я, і дані розподіляються по IP-адресами машин в кластері протягом робочого циклу. SQL Server підтримує дублювання даних з надмірністю в трьох сценаріях:

- відправляється «знімок» бази даних, який сервер відправляє одержувачам.
- Історія змін: все зміни в базі даних безперервно передаються користувачам.
- Синхронізація з іншими серверами: бази даних декількох серверів синхронізовані між собою. Зміни в усіх базах даних мають місце самостійно на кожному сервері, і дані узгоджуються впродовж синхронізації.

На відміну від інших процесів .NET Framework Структура розподіляє оновлення пам'яті і створює клієнтуру SQL Server управляє без використання вбудованих засобів Windows. Воно просуває продуктивність порівняно із звичайними алгоритмами Windows, оскільки алгоритми розподілу ресурсів особливо скоректовані для використання в структурах SQL Server. 2.

1.4 Програмна реалізація електронної реєстратури

1.4.1 Опис структури серверної частини

Серверна частина розроблена на базі бази даних Microsoft SQL Server. Для більш зручної роботи з функціоналом СУБД використовувалася спеціальна програма SQL Server 2012 Management Studio. Схема бази даних буде розглянута більш докладно нижче.

Для реалізації проекту була розроблена і створена база даних SQL.

Схема бази даних електронної реєстратури наведена на рисунок 1.10.

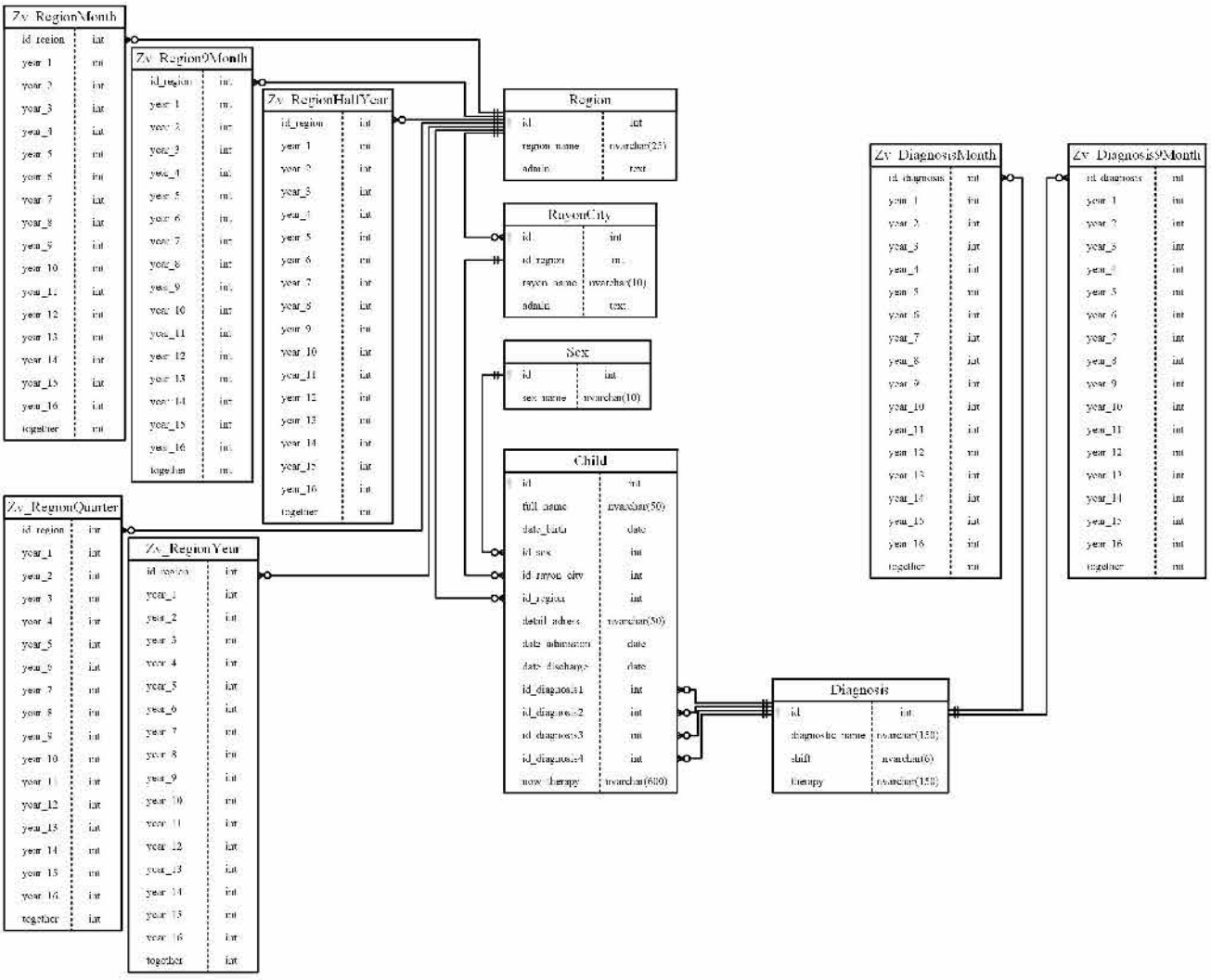


Рисунок 1.10. Схема бази даних електронної реєстратури

Нижче наводиться короткий опис кожної графічної таблиці.

Таблиця 1.1. Перелік основних реляційних таблиць бази даних

Перелік основних реляційних таблиць SQL-бази даних:	
Child:	інформація про пацієнтів відділення;
Diagnosis:	довідник шифрів і назв діагнозів нозологій захворювань;
Region:	довідник адміністративного поділу України;
RayonCity:	довідник адміністративного поділу України;
Sex:	довідник статей;
Zv_IncidenceMonth:	місячний звіт з охоплення лікуванням за віковими категоріями;
Zv_IncidenceQuarter:	квартальний звіт з охоплення лікуванням за віковими категоріями;
Zv_IncidenceHalfYear:	звіт з охоплення лікуванням за віковими категоріями за I півріччя;
Zv_Incidence9Month:	звіт з охоплення лікування за віковими категоріями за 9 місяців;
Zv_IncidenceYear:	річний звіт з охоплення лікуванням за віковими категоріями;
Zv_RegionMonth:	місячний звіт з охоплення лікуванням за регіонами проживання;
Zv_RegionQuarter:	квартальний звіт з охоплення лікуванням за регіонами проживання;
Zv_RegionHalfYear:	звіт з охоплення лікуванням за регіонами проживання за I півріччя;
Zv_Region9Month:	звіт з охоплення лікуванням за регіонами проживання за 9 місяців;
Zv_RegionYear:	річний звіт з охоплення лікуванням за регіонами проживання;
Zv_DiagnosisMonth:	місячний звіт з охоплення лікуванням за шифрами нозологій;
Zv_DiagnosisQuarter:	квартальний звіт з охоплення лікуванням за шифрами нозологій;
Zv_DiagnosisHalfYear:	звіт з охоплення лікуванням за шифрами нозологій за I півріччя;
Zv_Diagnosis9Month:	звіт з охоплення лікуванням за шифрами нозологій за 9 місяців;
Zv_DiagnosisYear:	річний звіт з охоплення лікуванням за шифрами нозологій;
Child_log: Child;	журнал реєстрації внесених змін до таблиці
Diagnosis_log:	журнал реєстрації внесених змін до таблиці Diagnosis;
Region_log:	журнал реєстрації внесених змін до таблиці Region;
RayonCity_log:	журнал реєстрації внесених змін до таблиці RayonCity.

Кожна таблиця має первинний і вторинний ключі, які забезпечують доступ до необхідної інформації в базі даних. Первинний ключ - це атрибут або набір атрибутів, які однозначно ідентифікують кортеж цього відношення. Первинний ключ повинен бути унікальним.

База даних приведена до четвертої нормальної форми. Четверта нормальна форма вимагає, щоб схема бази даних не містила нетривіальних багатих залежностей наборів атрибутів від чого-небудь, крім. Вважається, що таблиця знаходиться в 4NF тоді і тільки тоді, коли вона знаходиться в 4НФ, а багатозначні залежності є функціональними залежностями. Четверта нормальна форма виключає небажані структури даних - істотні залежності.

Таблиця 1.2. Перелік збережених SQL-процедур розробленої бази даних

Перелік збережених SQL-процедур розробленої бази даних:	
Append_patient:	добавлення в базу інформації про нового пацієнта;
Update_patient:	оновлення в базі інформації існуючого пацієнта;
BackUp_curentBase:	створення резервної копії бази даних;
CreateZvit_All:	створення всіх звітів за всі періоди звітності;
CreateZvit_Diagnosis:	формування звіту з охоплення лікування за шифрами нозологій;
CreateZvit_Incidence:	формування звіту з охоплення лікування за віковими категоріями;
CreateZvit_Region:	формування звіту з охоплення лікування за регіонами проживання;
Calc_ChidrenDiagnosis:	обчислення кількості дітей за вказаний період в розрізі шифрів нозологій;
Calc_ChidrenIncidence:	обчислення кількості дітей за вказаний період в розрізі вікових категорій;
Calc_ChidrenRegion:	обчислення кількості дітей за вказаний період в розрізі регіонів проживання;
Calc_SummaDiagnosisOfYear:	обчислення кількості встановлених діагнозів захворювань за вказаний період.

Перелік збережених SQL-процедур, що виконують роль тригерів в розробленій базі даних:

Таблиця 1.3. Перелік збережених SQL-процедур, що виконують роль тригерів в розробленій базі даних

Перелік збережених SQL-процедур, що виконують роль тригерів в розробленій базі даних:	
After_append_Child:	дії після додавання нового пацієнта;
After_append_Diagnosis:	дії після додавання нового шифру нозології;
After_append_RayonCity:	дії після додавання нового району;
After_append_Region:	дії після додавання нового регіону;
After_update_Child:	дії після внесення змін до кортежу пацієнта;
After_update_Diagnosis:	дії після внесення змін до кортежу шифру нозології;
After_update_RayonCity:	дії після внесення змін до кортежу району;
After_update_Region:	дії після внесення змін до кортежу регіону;
After_delete_Child:	дії після вилучення кортежу пацієнта;
After_delete_Diagnosis:	дії після вилучення кортежу шифру нозології;
After_delete_RayonCity:	дії після вилучення кортежу району;
After_delete_Region:	дії після вилучення кортежу регіону.

Впровадження SQL-процедур і тригерів дозволило прибрати з програмного коду всі операції по модифікації даних і операції по формуванню звітів.

Таким чином, в комп'ютерній системі реалізується архітектура клієнт-сервер, яка забезпечує надійний захист і контролює цілісність даних в базі даних серверами SQL

1.4.2 Клієнтська програма

Клієнтська програма розроблена в середовищі розробки Embarcadero C ++ Builder XE7. Вся бізнес-логіка системи була реалізована в серверній частині, тому основним завданням клієнтської програми є формування запитів до бази даних і їх обробка.

Схема взаємодії компонент доступу до БД електронної реєстратури зображена на рисунку 1.11

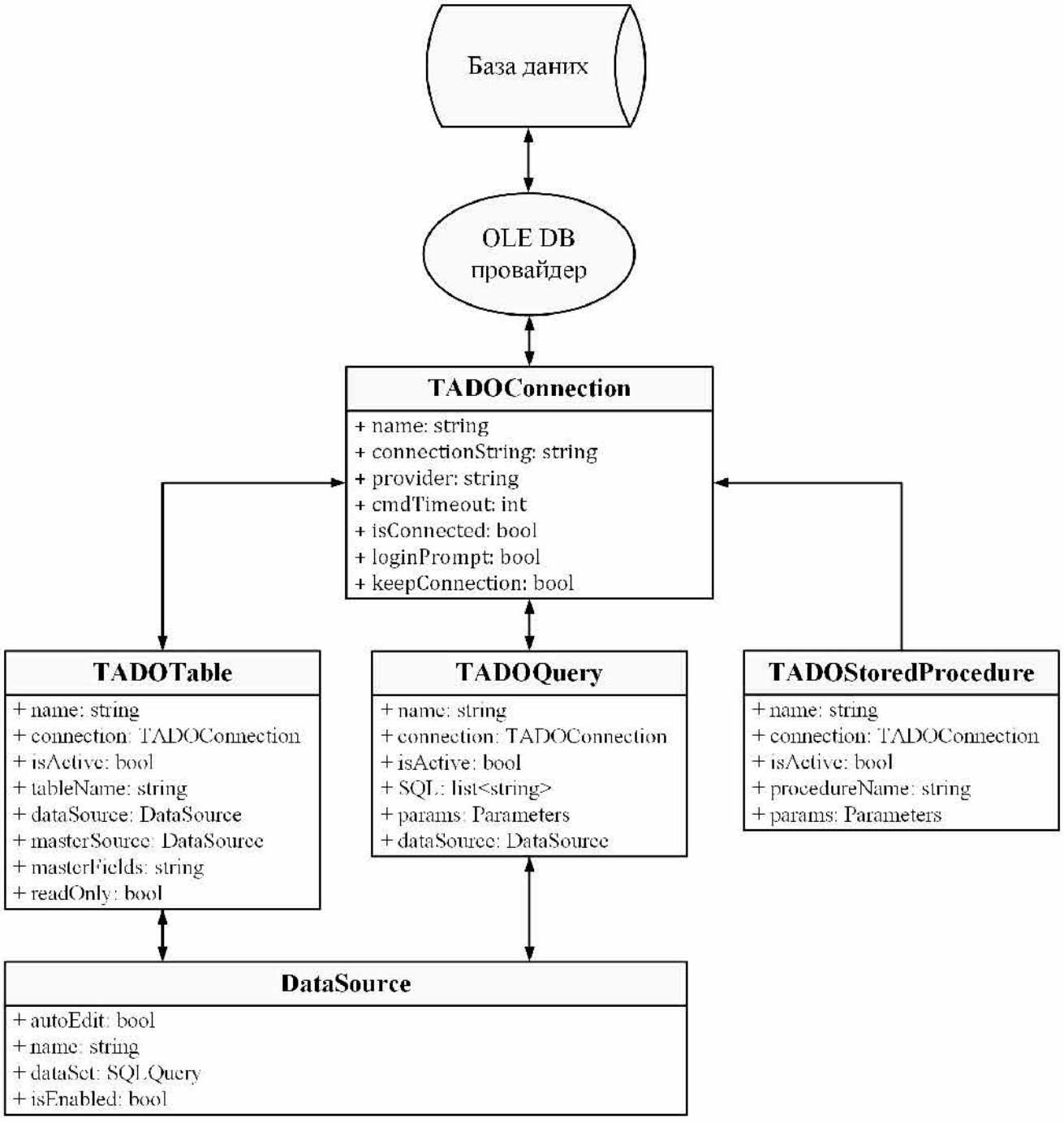


Рисунок 1.11. Схема взаємодії компонент доступу до БД

1.4.3 Доступ до бази даних

Для доступу до бази даних необхідно об'єднати кілька компонентів палітри dbGo, яка заснована на технології ADO. Технологія Microsoft ADO (об'єкти даних ActiveX) - одна із стандартних технологій Microsoft для доступу до джерел даних. Технологія ADO є доповненням до іншої технології доступу до даних - Microsoft OLE DB. [22]

Компонент TADOConnection инкапсулирует об'єкт ADOConnection і підключається до сховищ даних. При роботі з компонентом TADOConnection властивість ConnectionString вказує практично всю інформацію, необхідну для зв'язку зі сховищем даних.

Компонент TADOConnection управляє підключенням до бази даних. Якщо виникає помилка, пов'язана з програмою (помилка на стороні сервера або помилка рівня ADO), цей об'єкт генерує виняток, яке може бути оброблено об'єктом TError.

Компонент TADODataSet реалізує базовий набір даних для технології ADO. Властивість Connection вказує на об'єкт TADOConnection, який підключається до сервера бази даних. Властивість ConnectionString за змістом схоже на однойменну властивість компонента TADOConnection.

Компонент TDataSource діє як посередник між компонентами TADODataSet і компонентами TDataControls, елементами, які забезпечують уявлення даних в фоновому режимі. Компонент TDataSource управляє відносинами між даними в компонентах TDataControls. [23]

Компонент TADOTable використовується для доступу до сховищ даних ADO та подання інформації про них в табличній формі. Цей компонент відбувається з класу TCustomADODataSet, тому він має в своєму розпорядженні набір властивостей і методів, аналогічний набору елементів компонента TADODataSet. Можна сказати, що можливості компонента TADOTable є підмножиною можливостей компонента TADODataSet. Компонент підключається до бази даних за допомогою властивості Connection або ConnectionString. Ім'я таблиці вказується в властивості TableName. Під час

розробки проекту в режимі «Дизайн» властивість TableName перетворюється в список, що розкривається, в якому ви можете вибрати таблицю бази даних. Властивість TableDirect вказує, як набір даних зв'язується з сховищем даних. Якщо для властивості встановлено значення true, компонент використовує фонові оператори SQL для вилучення даних, а якщо для властивості встановлено значення false, для самого компонента встановлюється значення SEL. Властивість ReadOnly надає можливість встановлювати обмеження для таблиці, доступної лише для читання, тим самим забороняючи зміна даних. Властивість MasterSource вказує компонент TDataSource, який використовується для зв'язку з MasterDetail. Метод GetIndexNames повертає список індексів, доступних для компонента, у вигляді списку.

Компонент TADOQuery, який призначений для передачі команд SQL сервера бази даних, також є нащадком класу TCustomADODataSet. Компонент TADOQuery можна розглядати як аналог компонента TSQLQuery для механізму ADO. Компонент підключається до бази даних за допомогою властивості Connection або ConnectionString. Текст запиту записується у властивість SQL. Запити можна параметризрвані, і в цьому випадку параметри запиту повинні міститися в колекції властивостей Parameters. Якщо запит, зазначений у властивості SQL, повертає набір даних, його слід відкрити за допомогою методу Open (або встановивши для властивості Active значення true). Якщо запит не повинен повертати набір даних, його слід виконати з допомогою методу ExecSQL. Метод ExecSQL повертає кількість записів, включених в набір результатів виконаного запиту. Таке ж значення міститься у властивості RowsAffected.

Компонент TADOStoredProc, як і компонент TSQLStoredProc, дозволяє вам отримати доступ до збережених процедур, що містяться в базах даних.[24]

1.4.4 Перелік модулів комп'ютерної системи

Програмний код проекту містить 17 модулів. Ієрархічну структуру комп'ютерної системи, до якої входять зазначені модулі, наведено нижче:

Таблиця 1.4. Перелік модулів проекту

Ієрархічну структуру комп'ютерної системи, до якої входять зазначені модулі	
IP54_Chyzh:	головний модуль клієнтської програми;
Main:	головна форма, що містить меню доступу до всіх режимів;
DataModule:	модуль, що містить невізуальні компоненти доступу до бази даних;
About:	форма, що містить інформацію про комп'ютерну систему та її автора;
Help_Admin:	форма заповнення та редагування адміністративного поділу України;
Help_Diag:	форма заповнення та редагування шифрів нозологій захворювань;
FormEdit:	форма перегляду повного або відфільтрованого за певними критеріями списку пацієнтів відділення;
Append_Patient:	форма заповнення або редагування інформації певного пацієнта;
FilterSet:	форма вибору параметрів для виконання подальшої фільтрації пацієнтів;
Form_ZvitShifr:	форма перегляду звіту з охоплення лікуванням за діагнозами нозологій;
Chart_Diag:	форма перегляду звіту з охоплення лікуванням за діагнозами нозологій у вигляді гістограми або кругової діаграм
Form_ZvitYear:	форма перегляду звіту з охоплення лікуванням за віковими категоріями;
Chart_Year:	форма перегляду звіту з охоплення лікування за віковими категоріями у вигляді гістограми або кругової діаграм
Form_ZvitRegion:	форма перегляду звіту з охоплення лікуванням за регіонами проживання;
Chart_Region:	форма перегляду звіту з охоплення лікування за регіонами проживання у вигляді гістограми або кругової діаграм
Form_Show_Message:	вікно повідомлення;
Form_Message_Dlg:	вікно вибору подальшої дії;

1.4.5 Користувацькі функції, створені для забезпечення інтерфейсу

Для реалізації інтерфейсу користувача створено підпрограми, специфікацію яких наведено нижче:

Модуль IP54_Chyzh (головний модуль):
USEFORM("DataModule.cpp", DM) – модуль даних;
USEFORM("About.cpp", FormAbout) – форма з інформацією про комп'ютерну систему;
USEFORM("Main.cpp", MainForm) – головне вікно;
USEFORM("Help_Admin.cpp", FormHelpsAdmin) – форма адміністративного поділу України;
USEFORM("Help_Diag.cpp", FormHelpsDiagnoz) – форма довідника шифрів нозологій;
USEFORN("FornEdit.cpp", FornEditInfo) – форма перегляду та редагування інформації про пацієнтівідділення;
USEFORN("Append_Patient.cpp", Forn_Append) – форма добавлення та зміни інформації про пацієнтів відділення;
USEFORM("FilterSet.cpp", FormFilertSet) – форма фільтрації списку пацієнтів відділення;
USEFORM("Form_ZvitShifr.cpp", FormZvitShifr) – форма перегляду звіту з охоплення лікування за діагнозами нозологій;
USEFORM("Chart_Diag.cpp", ChartDiag) – форма перегляду звіту з охоплення лікування за діагнозами нозологій у формі діаграми;
USEFORM("Forn_ZvitYear.cpp", FormZvitYear) – форма перегляду звіту з охоплення лікування за віковими категоріями;
USEFORM("Chart_Year.cpp", ChartYear) – форма перегляду звіту з охоплення лікування за віковими категоріями у формі діаграми;
USEFORM("Forn_ZvitRegion.cpp", FormZvitRegion) – форма перегляду звіту з охоплення лікування за регіонами проживання;
USEFORM("Chart_Region.cpp", ChartRegoin) – форма перегляду звіту з охоплення лікування за регіонами проживання у формі діаграми;
USEFORM("Form_Show_Message.cpp", Show_Message) – вікно показу повідомлення;
USEFORM("Form_Message_Dlg.cpp", Message_Dialog) – діалогове вікно вибору подальшої дії.

Модуль DataModule:
void __fastcall DeleteRecord(TDataSet *DataSet, AnsiString id_fld, AnsiString TableName) – метод вилучення запису з таблиці пацієнтів;
void __fastcall TableAfterInsertDelete(TDataSet *DataSet, AnsiString id_fld) – метод встановлення в поточну позицію перегляду вказаного запису таблиці;
bool __fastcall Checked_BackUp(bool write, bool checked) – метод визначення і встановлення параметру автоматичної архівації бази даних при виході із клієнтської програми;

int __fastcall Id_in_ComboBox(TComboBox *ComboBox, AnsiString id_str) – метод визначення індексу заданого рядка у списку, що розкривається;
void __fastcall MyShowMessage(AnsiString caption, AnsiString title, AnsiString text) – метод формування та виводу вікна повідомлення;
int __fastcall MyMessageDlg(AnsiString caption, AnsiString title, AnsiString text, TMsgDlgType dialogType) – метод формування та виводу діалогового вікна вибору подальшої дії;
AnsiString __fastcall Fill_fieldCaptoin(TDBGrid *DBGrid_Zvit, int index) – метод формування заголовку колонки за її іменем;
AnsiString DateToStringForTitle(TDateTime Date, int Tag) – метод формування текстового заголовку періоду звіту за датою його створення.

Модуль About:
void __fastcall BtnExitClick(TObject *Sender) – метод обробки кліку на кнопці "Ок".

Модуль Main:
void __fastcall FornCreate(TObject *Sender) – метод обробки події створення форми;
void __fastcall FornResize(TObject *Sender) – метод обробки події зміни розмірів форми;
void __fastcall FornShow(TObject *Sender) – метод обробки події показу форми;
void __fastcall Enab(bool En) – метод встановлення доступності елементів головного меню;
void __fastcall AboutClick(TObject *Sender) – метод обробки пункту меню "Про автора";
void __fastcall EditInfoClick(TObject *Sender) – метод обробки пункту меню "Редагування відомостей";
void __fastcall ZvitCreateClick(TObject *Sender) – метод обробки пункту меню "Формування звітності";
void __fastcall Zvit_YearClick(TObject *Sender) – метод обробки пункту меню "Охоплення лікуванням за віковими категоріями";
void __fastcall Zvit_ShifrClick(TObject *Sender) – метод обробки пункту меню "Охоплення лікуванням за шифрами нозологій";
void __fastcall Zvit_RegionClick(TObject *Sender) – метод обробки пункту меню "Охоплення лікуванням за регіонами";
void __fastcall ServisHelpsAdminClick(TObject *Sender) – метод обробки пункту меню "Адміністративний поділ України";
void __fastcall ServisHelpsShifrClick(TObject *Sender) – метод обробки пункту меню "Довідник нозологій";
void __fastcall ServisCheckDateClick(TObject *Sender) – метод обробки пункту меню "Зміна дати";

void __fastcall BtnDateCancelClick(TObject *Sender) – метод обробки кліку на кнопці “Відмова” (панель зміни дати);
void __fastcall BackUpClick(TObject *Sender) – метод обробки пункту меню “Створення резервної копії при виході”;
void __fastcall ExitClick(TObject *Sender) – метод обробки пункту меню “Вихід”;
void __fastcall FornCloseQuery(TObject *Sender, bool &CanClose) – метод обробки події перед закриттям форми.

Модуль Help_Admin:
void __fastcall FornCreate(TObject *Sender) – метод обробки події створення форми;
void __fastcall FornResize(TObject *Sender) – метод обробки події зміни розмірів форми;
void __fastcall BitBtn_AppendClick(TObject *Sender) – метод обробки кліку на кнопках “Додати область” та “Додати район”;
void __fastcall BitBtn_DeleteClick(TObject *Sender) – метод обробки кліку на кнопках “Вилучити область” та “Вилучити район”;
void __fastcall DS_RayonDataChange(TObject *Sender, TField *Field) – метод обробки зміни стану таблиці районів;
void __fastcall DS_RegionDataChange(TObject *Sender, TField *Field) – метод обробки зміни стану таблиці регіонів;
void __fastcall FornKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіш на формі;
bool __fastcall PermissionDelete(AnsiString fld, TDataSet *DataSet) – метод визначення дозволу на вилучення запису з таблиці;
void __fastcall Help_Table_RegionBeforeDelete(TDataSet *DataSet) – метод обробки події перед вилученням регіону;
void __fastcall Help_Table_AdminBeforePost(TDataSet *DataSet) – метод обробки події перед записом інформації до бази даних;
void __fastcall Help_Table_RayonBeforeDelete(TDataSet *DataSet) – метод обробки події перед вилученням району;
void __fastcall Table_AfterUpDate(TDataSet *DataSet) – метод встановлення відмітки про запис інформації до бази даних;
void __fastcall BtnExitClick(TObject *Sender) – метод обробки кліку на кнопці “Вихід”;
void __fastcall FornClose(TObject *Sender, TCloseAction &Action) – метод обробки події закриття форми

Модуль FormEdit:
void __fastcall FornCreate(TObject *Sender) – метод обробки події створення форми;
void __fastcall FornResize(TObject *Sender) – метод обробки події зміни розмірів форми;
void __fastcall OpenChild_AllRecords(AnsiString _sql) – метод встановлення властивості SQL-запиту до джерела даних;
void __fastcall BitBtn_EditAppendClick(TObject *Sender) – метод обробки кліку на кнопках “Додати пацієнта” та “Змінити дані”;
void __fastcall Edit_SerchFIOKeyUp(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки уведення символів в поле пошуку за ПІП;
void __fastcall Edit_SerchFIOExit(TObject *Sender) – метод обробки втрати фокусу поля уведення зразка пошуку;
void __fastcall BitBtn_DeleteClick(TObject *Sender) – метод обробки кліку на кнопці “Вилучити пацієнта”;
void __fastcall FornKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіш на формі;
void __fastcall BitBtn_fillterSetClick(TObject *Sender) – метод обробки кліку на кнопках “Новий фільтр” та “Змінити фільтр”;
void __fastcall DBEditKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіші на полях;
void __fastcall BitBtn_PrintClick(TObject *Sender) – метод обробки кліку на кнопці “Друк списку”;
void __fastcall BitBtn_fillterClearClick(TObject *Sender) – метод обробки кліку на кнопці “Зняти фільтр”;
void __fastcall BtnExitClick(TObject *Sender) – метод обробки кліку на кнопці “Вихід”;
void __fastcall FornClose(TObject *Sender, TCloseAction &Action) – метод обробки події закриття форми.

Модуль Help_Diag:
void __fastcall FornCreate(TObject *Sender) – метод обробки події створення форми;
void __fastcall FornResize(TObject *Sender) – метод обробки події зміни розмірів форми;
void __fastcall FornShow(TObject *Sender) – метод обробки події показу форми;
void __fastcall BitBtn_Append_RegionClick(TObject *Sender) – метод обробки кліку на кнопці “Додати нозологію”;

void __fastcall BitBtn_Delete_DiagClick(TObject *Sender) – метод обробки кліку на кнопці “Вилучити нозологію”;
void __fastcall Edit_SerchExit(TObject *Sender) – метод обробки події втрати фокусу поля уведення зразка пошуку;
void __fastcall Edit_SerchKeyUp(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки події уведення символів у полі зразка пошуку;
void __fastcall FornKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіші на формі;
void __fastcall Help_Table_DiagnozBeforeDelete(TDataSet *DataSet) – метод обробки події перед вилученням діагнозу нозології;
void __fastcall Help_Table_DiagnozBeforePost(TDataSet *DataSet) – метод обробки події перед записом нозології до бази даних;
void __fastcall Table_AfterUpDate(TDataSet *DataSet) – метод встановлення відмітки про запис інформації до бази даних;
void __fastcall BtnExitClick(TObject *Sender) – метод обробки кліку на кнопці “Вихід”;
void __fastcall FornClose(TObject *Sender, TCloseAction &Action) – метод обробки події закриття форми.

Модуль FormEdit:
void __fastcall FornCreate(TObject *Sender) – метод обробки події створення форми;
void __fastcall FornResize(TObject *Sender) – метод обробки події зміни розмірів форми;
void __fastcall OpenChild_AllRecords(AnsiString _sql) – метод встановлення властивості SQL-запиту до джерела даних;
void __fastcall BitBtn_EditAppendClick(TObject *Sender) – метод обробки кліку на кнопках “Додати пацієнта” та “Змінити дані”;
void __fastcall Edit_SerchFIOKeyUp(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки уведення символів в поле пошуку за ПІП;
void __fastcall Edit_SerchFIOExit(TObject *Sender) – метод обробки втрати фокусу поля уведення зразка пошуку;
void __fastcall BitBtn_DeleteClick(TObject *Sender) – метод обробки кліку на кнопці “Вилучити пацієнта”;
void __fastcall FornKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіш на формі;

void __fastcall BitBtn_fillterSetClick(TObject *Sender) – метод обробки кліку на кнопках “Новий фільтр” та “Змінити фільтр”;
void __fastcall DBEditKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіші на полях;
void __fastcall BitBtn_PrintClick(TObject *Sender) – метод обробки кліку на кнопці “Друк списку”;
void __fastcall BitBtn_fillterClearClick(TObject *Sender) – метод обробки кліку на кнопці “Зняти фільтр”;
void __fastcall BtnExitClick(TObject *Sender) – метод обробки кліку на кнопці “Вихід”;
void __fastcall FornClose(TObject *Sender, TCloseAction &Action) – метод обробки події закриття форми.

Модуль Append_Patient:
void __fastcall FornCreate(TObject *Sender) – метод обробки події створення форми;
void __fastcall FornShow(TObject *Sender) – метод обробки події показу форми;
void __fastcall FornResize(TObject *Sender) – метод обробки події зміни розмірів форми;
void __fastcall BitBtn_AppendClick(TObject *Sender) – метод обробки кліку на кнопці “Додати інформацію в базу / Внести зміни в базу”;
void __fastcall Clear_PanelAdd() – метод очистки візуальних компонентів форми перед уведенням нових даних;
void __fastcall Full_PanelAdd() – метод заповнення візуальних компонентів форми даними поточного пацієнта;
void __fastcall Table_date_SetText(TField *Sender, const UnicodeString Text) – метод перевірки уведеної інформації на валідність дати;
bool __fastcall Checking_Info(int &err) – метод перевірки уведеної інформації на повноту та коректність;
void __fastcall Full_HelpContext(UnicodeString hlp) – метод встановлення ключа онлайн-довідки в залежності від режиму використання форми;
void __fastcall Edit_newDateChange(TObject *Sender) – метод обробки отримання фокусу полями з датою;
void __fastcall Edit_newKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіш на полях уведення;
void __fastcall DateTimePicker_newExit(TObject *Sender) – метод обробки втрати фокусу компонентами типу “календар”;
void __fastcall DateTimePicker_newDateKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіш на компонентах типу “календар”;
void __fastcall Edit_RayonClick(TObject *Sender) – метод обробки кліку на полях місця проживання пацієнта;
void __fastcall TreeUADblClick(TObject *Sender) – метод обробки подвійного кліку на елементі дерева адміністративного поділу України;
void __fastcall ComboBox_ShifrChange(TObject *Sender) – метод обробки події вибору позиції в списку шифрів діагнозів;
void __fastcall ComboBox_DiagnozChange(TObject *Sender) – метод обробки події вибору позиції в списку назв діагнозів;
void __fastcall ComboBox_ShifrKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіші на компонентах типу “список, що розкривається”;
void __fastcall CheckBoxClick(TObject *Sender) – метод обробки події вибору прапорця;

void __fastcall Fill_Proc_Parameters(TADOStoredProc *ADOProc) – метод заповнення параметрів збереженої SQL-процедури;
void __fastcall BitBtn_AbortClick(TObject *Sender) – метод обробки кліку на кнопці "Скасувати додавання / Скасувати зміни";
void __fastcall FornClose(TObject *Sender, TCloseAction &Action) – метод обробки події закриття форми.

Модуль FilterSet:
void __fastcall FornCreate(TObject *Sender) – метод обробки події створення форми;
void __fastcall FornResize(TObject *Sender) – метод обробки події зміни розмірів форми;
void __fastcall Check_Filter_SexClick(TObject *Sender) – метод обробки події вибору прапорця "Стать";
void __fastcall Check_Filter_RegionClick(TObject *Sender) – метод обробки події вибору прапорця "Регіон проживання";
void __fastcall Check_Filter_Birth_exactlyClick(TObject *Sender) – метод обробки події вибору прапорця "точно" в рядку "Дата народження";
void __fastcall Check_Filter_Birth_inClick(TObject *Sender) – метод обробки події вибору прапорця "з" в рядку "Дата народження";
void __fastcall Check_Filter_Birth_toClick(TObject *Sender) – метод обробки події вибору прапорця "по" в рядку "Дата народження";
void __fastcall Check_Filter_Add_exactlyClick(TObject *Sender) – метод обробки події вибору прапорця "точно" в рядку "Дата поступлення";
void __fastcall Check_Filter_Add_inClick(TObject *Sender) – метод обробки події вибору прапорця "з" в рядку "Дата поступлення";
void __fastcall Check_Filter_Add_toClick(TObject *Sender) – метод обробки події вибору прапорця "по" в рядку "Дата поступлення";
void __fastcall Check_Filter_Dec_exactlyClick(TObject *Sender) – метод обробки події вибору прапорця "точно" в рядку "Дата виписки";
void __fastcall Check_Filter_Dec_inClick(TObject *Sender) – метод обробки події вибору прапорця "з" в рядку "Дата виписки";
void __fastcall Check_Filter_Dec_toClick(TObject *Sender) – метод обробки події вибору прапорця "по" в рядку "Дата виписки";
void __fastcall Check_Filter_ShifrClick(TObject *Sender) – метод обробки події вибору прапорця "Діагноз";
void __fastcall C_Box_ShifrChange(TObject *Sender) – метод обробки події вибору позиції зі списку шифрів діагнозів;
void __fastcall C_Box_DiagnozChange(TObject *Sender) – метод обробки події вибору позиції зі списку назв діагнозів;
void __fastcall BitBtn_Clear_FilterClick(TObject *Sender) – метод обробки кліку на кнопці "Очистити всі фільтри";
void __fastcall BitBtn_Apply_FilterClick(TObject *Sender) – метод обробки кліку на кнопці "Знайти всіх пацієнтів";
void __fastcall FornKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіші на формі;
void __fastcall Filter_Date_Clear() – метод встановлення всім полям з датами значення поточної дати;
void __fastcall BitBtn_Exit_FilterClick(TObject *Sender) – метод обробки кліку на кнопці "Відмова".

Модуль Form_ZvitShifr:
void __fastcall FornShow(TObject *Sender) – метод обробки події показу форми;
void __fastcall FornResize(TObject *Sender) – метод обробки події зміни розмірів форми;
void __fastcall BitBtn_PrintClick(TObject *Sender) – метод обробки кліку на кнопці “Друк звітів”;
void __fastcall FornKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіш на формі;
void __fastcall PrintClick(TObject *Sender) – метод обробки кліку на пункті меню “Звіт в традиційні формі”;
void __fastcall DiagClick(TObject *Sender) – метод обробки кліку на пункті меню “... за діагнозами (Разом)”;
void __fastcall Diag_CurentYearClick(TObject *Sender) – метод обробки кліку на пункті меню “... за діагнозами вказаного віку”;
void __fastcall Year_CurentDiagClick(TObject *Sender) – метод обробки кліку на пункті меню “... за віковими категоріями вказаного діагнозу”;
void __fastcall DBGrid_ZvitTitleClick(TColumn *Column) – метод обробки події подвійного кліку на заголовку колонки сітки;
void __fastcall BitBtn_ExitClick(TObject *Sender) – метод обробки кліку на кнопці “Вихід”;
void __fastcall FornClose(TObject *Sender, TCloseAction &Action) – метод обробки події закриття форми.

Модуль Chart_Diag:
void __fastcall FornResize(TObject *Sender) – метод обробки події зміни розмірів форми;
void __fastcall BitBtn_BarPieClick(TObject *Sender) – метод обробки кліку на кнопках “Гістограма” та “Кругова”;
void __fastcall BitBtn_PrintClick(TObject *Sender) – метод обробки кліку на кнопці “Друк”;
void __fastcall BitBtn_ExportClick(TObject *Sender) – метод обробки кліку на кнопці “Експорт в BMP”;
void __fastcall FornKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіш на формі;
void __fastcall UpDownClick(TObject *Sender, TUDBtnType Btn) – метод обробки події зміни лічильників, які вказують номер поточної сторінки та кількість рядів даних на сторінці гістограми;
void __fastcall BitBtn_ExitClick(TObject *Sender) – метод обробки кліку на кнопці “Вихід”.

Модуль Form_Zvit Year:
void __fastcall FornShow(TObject *Sender) – метод обробки події показу форми;
void __fastcall FornResize(TObject *Sender) – метод обробки події зміни розмірів форми;
void __fastcall BitBtn_PrintClick(TObject *Sender) – метод обробки кліку на кнопці “Друк звітів”;
void __fastcall FornKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіш на формі;
void __fastcall PrintClick(TObject *Sender) – метод обробки кліку на пункті меню “Звіт в традиційні формі”;
void __fastcall ChartBarClick (TObject *Sender) – метод обробки кліку на пункті меню “... за статтю та віком (Разом)”;
void __fastcall YearClick (TObject *Sender) – метод обробки кліку на пункті меню “... за віком (Разом)”;
void __fastcall SexClick (TObject *Sender) – метод обробки кліку на пункті меню “... за статтю (Разом)”;
void __fastcall BitBtn_ExitClick(TObject *Sender) – метод обробки кліку на кнопці “Вихід”;
void __fastcall FornClose(TObject *Sender, TCloseAction &Action) – метод обробки події закриття форми.

Модуль Form_ZvitRegion:
void __fastcall FornShow(TObject *Sender) – метод обробки події показу форми;
void __fastcall FornResize(TObject *Sender) – метод обробки події зміни розмірів форми;
void __fastcall BitBtn_PrintClick(TObject *Sender) – метод обробки кліку на кнопці “Друк звітів”;
void __fastcall FornKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіш на формі;
void __fastcall PrintClick(TObject *Sender) – метод обробки кліку на пункті меню “Звіт в традиційні формі”;
void __fastcall RegionClick(TObject *Sender) – метод обробки кліку на пункті меню “... за регіонами (Разом)”;
void __fastcall Region_CurentYearClick(TObject *Sender) – метод обробки кліку на пункті меню “... за регіонами вказаного віку”;
void __fastcall Year_CurentRegionClick(TObject *Sender) – метод обробки кліку на пункті меню “... за віковими категоріями вказаного регіону”;
void __fastcall DBGrid_ZvitTitleClick(TColumn *Column) – метод обробки подвійного кліку на заголовку колонки сітки;
void __fastcall BitBtn_ExitClick(TObject *Sender) – метод обробки кліку на кнопці “Вихід”;

Модуль Chart_Year:
void __fastcall FornResize(TObject *Sender) – метод обробки події зміни розмірів форми;
void __fastcall BitBtn_BarPieClick(TObject *Sender) – метод обробки події кліку на кнопках “Гістограма” та “Кругова”;
void __fastcall BitBtn_PrintClick(TObject *Sender) – метод обробки кліку на кнопці “Друк”;
void __fastcall BitBtn_ExportClick(TObject *Sender) – метод обробки кліку на кнопці “Експорт в BMP”;
void __fastcall FornKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіш на формі;
void __fastcall BitBtn_ExitClick(TObject *Sender) – метод обробки кліку на кнопці “Вихід”.

fastcall FornResize(TObject *Sender) – метод обробки події зміни розмірів форми;
void __fastcall BitBtn_BarPieClick(TObject *Sender) – метод обробки кліку на кнопках “Гістограма” та “Кругова”;
void __fastcall BitBtn_PrintClick(TObject *Sender) – метод обробки кліку на кнопці “Друк”;
void __fastcall BitBtn_ExportClick(TObject *Sender) – метод обробки кліку на кнопці “Експорт в BMP”;
void __fastcall FornKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) – метод обробки натиску клавіш на формі;
void __fastcall BitBtn_ExitClick(TObject *Sender) – метод обробки кліку на кнопці “Вихід
fastcall FornResize(TObject *Sender) – метод обробки події зміни розмірів форми;

Модуль Form_Show_Message:
void __fastcall FornShow(TObject *Sender) – метод обробки події показу форми;
void __fastcall BitBtn_OkClick(TObject *Sender) – метод обробки кліку на кнопці “Ок”.

Модуль Form_Message_Dlg:
void __fastcall FornCreate(TObject *Sender) – метод обробки події створення форми;
void __fastcall FornShow(TObject *Sender) – метод обробки події показу форми;
void __fastcall BitBtn_YesClick(TObject *Sender) – метод обробки кліку на кнопці “Так”;
void __fastcall BitBtn_NoClick(TObject *Sender) – метод обробки кліку на кнопці “Ні”;
void __fastcall FornCloseQuery(TObject *Sender, bool &CanClose) – метод обробки події перед закриттям форми.

1.4.6 Створення та модифікація даних

Інформація про новий пацієнта вводиться користувачем в клієнтської частини програми. Для зручності цього процесу створюється окрема форма, на якій створюються поля (TEdit), списки (TComboBox), прапори (TCheckBox) і компонент TTreeView. Після заповнення необхідної інформації всіх необхідних об'єктів система виконує перевірку введеної інформації. Запис інформації в таблицю пацієнтів реалізована на стороні сервера шляхом виклику збереженої SQL-процедури Append_patient, яка в якості параметрів передається вся введена інформація.

Модифікація даних була здійснена аналогічно введенню нового пацієнта. Відмінність полягає в заповненні візуальних об'єктів форми значеннями відповідних полів поточного запису з таблиці пацієнтів. Запис змінених даних в таблицю пацієнтів здійснюється на стороні сервера шляхом виклику збереженої SQL-процедури Update-patient з аналогічною SQL-процедурою Append-patient.

1.4.7 Контекстний пошук, фільтрація даних

У клієнтської частини програм реалізований контекстний пошук за прізвищем пацієнта по символам, введеним в компонент TEdit.

Фільтрація здійснюється за допомогою набору прапорів (TCheckBox), списків, що розкриваються (TComboBox) і календарів (TDateTimePicker). За

допомогою TCheckBox встановлюється мітка для позначення елемента, значення якого будуть відфільтровані. За допомогою TComboBox конкретне значення цього елемента вибирається зі списку. Компонент TDateTimePicker дозволяє вибрати потрібну дату в календарі або ввести її з клавіатури. Система забезпечує фільтрацію даних одночасно за кількома критеріями фільтрації.

Завдяки такому підходу до пошуку необхідних критеріїв даних, використання цього програмного забезпечення стає для користувача набагато зручніше.

1.4.8 Медична статистична звітність

Будь яка медична статистична звітність повинна охоплювати інформацію за наступні періоди: місяць, квартал, півріччя, 9 місяців, рік.

Формування звіту здійснюється на стороні сервера шляхом виклику збереженої SQL-процедури CreateZvit__All, яка в якості параметрів передається в зазначену користувачем дату і дату. Результати сформованих звітів заносяться до відповідних таблиць бази даних (їх імена починаються з «Zv__»), які користувач може переглядати і роздруковувати.

1.4.9 Медична статистична

Перегляд статистичних звітів здійснюється візуальним компонентом TDBGrid. Для кожного звіту реалізовані режими друку і перегляду та друку в вигляді гістограм або кругових діаграм.

Для того, щоб роздрукувати звіт, використовується стандартний компонент TfrxReport середовища розробки Embarcadero C ++ Builder XE7, до якого SQL буде прийнятий в якості параметра - це запит на запит.

Схема ієрархічної структури проекту системи для обліку та статистичного аналізу даних пацієнтів наведена на рис.1.12.

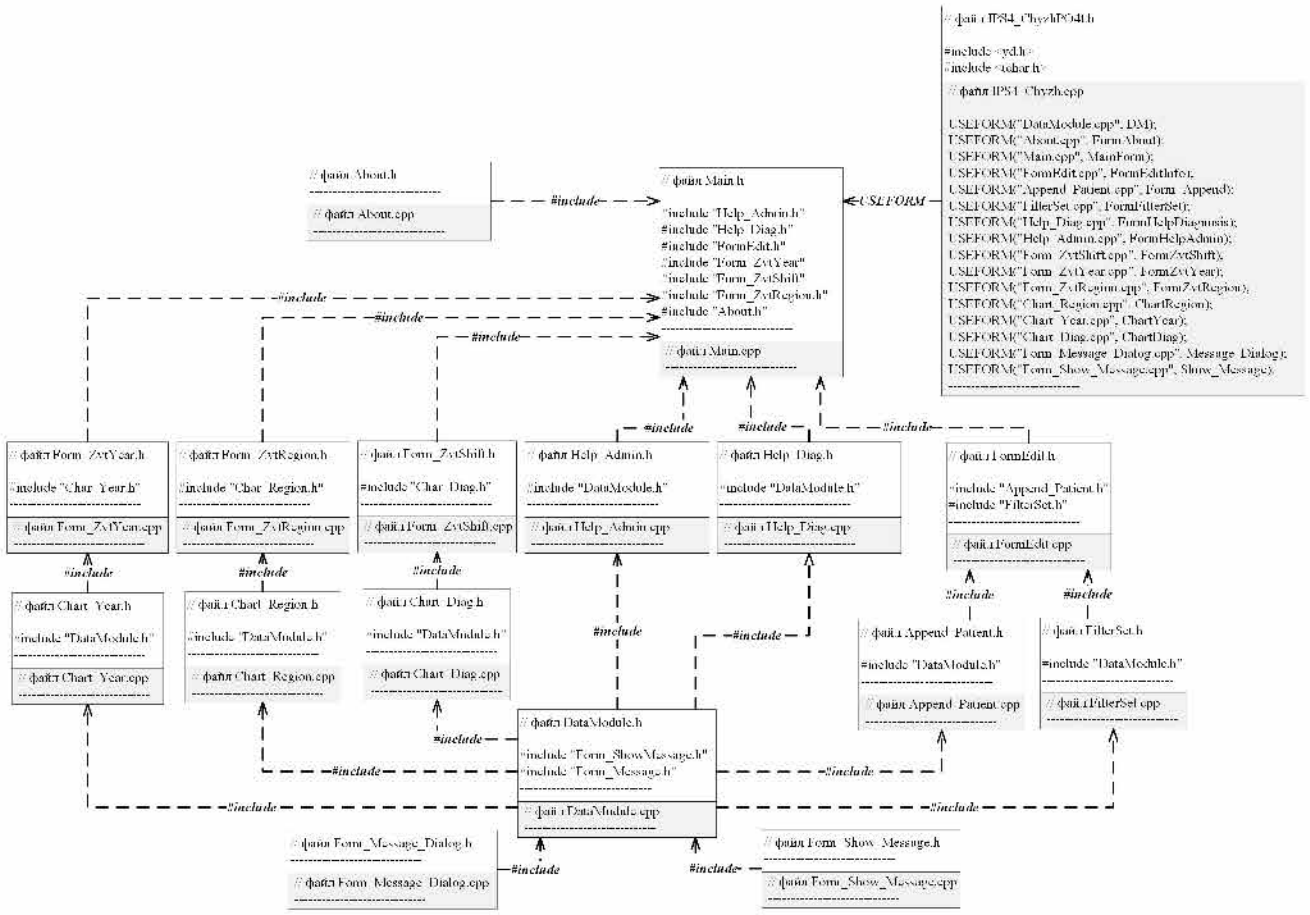


Рисунок 1.12. Схема ієрархічної структури проекту

Перегляд звіту в гістограмі кругової діаграми, реалізованої з використанням стандартного компонента TDBChart з використанням середовища розробки Embarcadero C ++ Builder XE7. Кожен TBarSeries можна налаштувати для створення і виведення діаграм бажаного типу. Для друку діаграм використовується метод PrintLandscape () компонента TDBChart. Запис діаграм в файл у форматі .bmp здійснюється викликом методу SaveToBitmapFile () компонента TDBChart, якому в якості параметра передається ім'я графічного файлу.

1.4.10 Автоматичне створення резервної копії БД

У розробленій системі реалізована можливість автоматичного створення резервної копії бази даних. Для цього потрібно в розділі «Інструменти» активувати пункт меню «Створювати резервну копію при виході» в розділі “Сервіс”. Після активації даного меню резервну копію бази даних буде

створюватис шкірного разу до закриття клієнтської програми (за умови внесення будь-яких змін до всіх - якой таблиці бази даних). Місцезнаходження резервної копії - це каталог BackUp_Base \ PatientOXMADIT на диску D.

Це необхідно для збереженні даних в разі несподіваної відмови комп'ютерної системи (отже, для збереження резервної копії несистемного диска або системи).

1.5 Тестування та налагодження електронної реєстратури

У цьому розділі представлені результати експериментальної експлуатації, інструкція з встановлення і опис роботи клієнтської програми даної системи. Розроблена система орієнтована на неврологів. Він призначений для ведення електронних карт пацієнтів стаціонарного відділення та формування медико-статистичної звітності.

1.5.1 Установка програмного забезпечення

Для забезпечення повноти установки необхідного для роботи системи, був створений окремий програмний модуль установки. Для його запуску необхідно відкрити файл Setup_system.exe, який знаходиться в папці SetupSystem.

Установка складається з трьох етапів:

1) установка бази даних Microsoft SQL Server;

2) установка клієнтської частини системи;

3) створення пустої бази даних на SQL сервері і настройка підключення до неї клієнтської частини системи.

Після запуску установника перевіряється наявність вже встановленого SQL-сервера, визначається його ім'я. У разі відсутності СУБД Microsoft SQL Server або при наявності більш ранньої версії на 2012 рік, програми встановлення про скачати з офіційного сайту і встановити її. Слід зазначити, що до кожного сервера Microsoft SQL Server також пред'являються певні системні вимоги. Тільки після установки SQL-сервера і перевірки його роботи стануть доступні кнопки наступних режимів установки.

Перед установкою клієнтської частини системи необхідно вказати її розташування на комп'ютері і підняти або опустити прапорець поруч із пунктом "створити ярлик на робочому столі». База даних буде створена тільки при піднятому прапорі в пункті «База даних». При цьому необхідно вказати спосіб її подальшої установки: створення пустої бази даних або відновлення з резервної копії. Щоб продовжити режим установки, потрібно натиснути на кнопку «Встановити систему В результаті на SQL сервері з'явиться база з усім розробленим функціоналом, буде налаштовано підключення до неї і збережено у файлі BaseConected.dat. Про успішну установку системи користувача буде повідомлено у відповідному вікні (рис. 1.13).

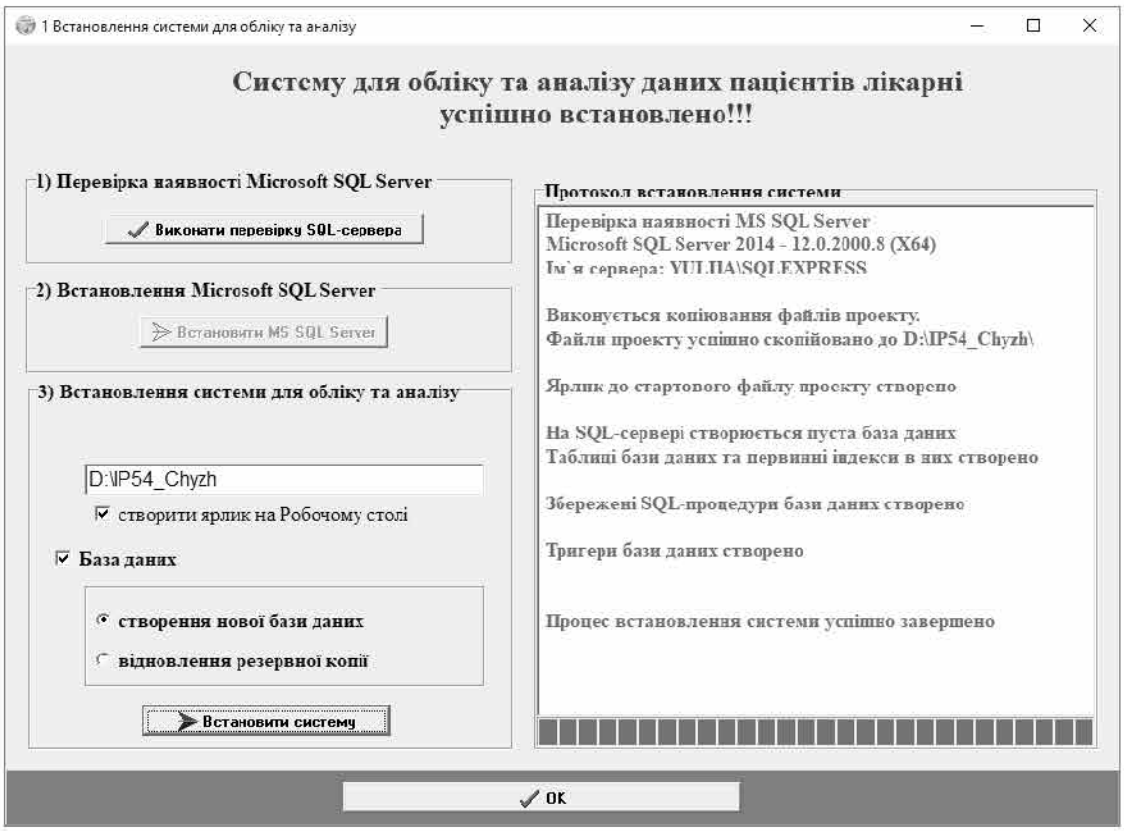


Рисунок 1.13. Вікно інсталятора програми

1.5.2 Клієнтська частина

Клієнтська програма запускається для заповнення бази даних інформацією про пацієнтів і аналізу даних шляхом виконання файлу IP54_Chyzh.exe, розміщеним за шляхом, вказаним при інсталяції. Інтерфейс клієнтської програми наведено на рис. 1.14.

Рисунок 1.14. Головне меню системи

Основні функції системи:

1) Введення, зберігання і редагування інформації про пацієнта відділення неврології.

2) Пошук пацієнта на прізвище (або її частини).

3) Фільтрація пацієнтів відділення за кількома показниками: статтю, регіоном проживання, дату народження, дату надходження на лікування, датою виписки.

4) Формування, перегляд та друк звітів за зазначені періоди.

5) Довідка доступна з будь-якого місця програми (F1)

1.5.3 Робота зі списком пацієнтів

Для входу в режим редагування списку пацієнтів необхідно вибрати команду «Редагувати списки» в Головному меню або натиснути на зображення лікаря з дітьми. Після цього відкриється вікно перегляду інформації, яке показано на рис. 1.15. Основні елементи цієї форми описані нижче.

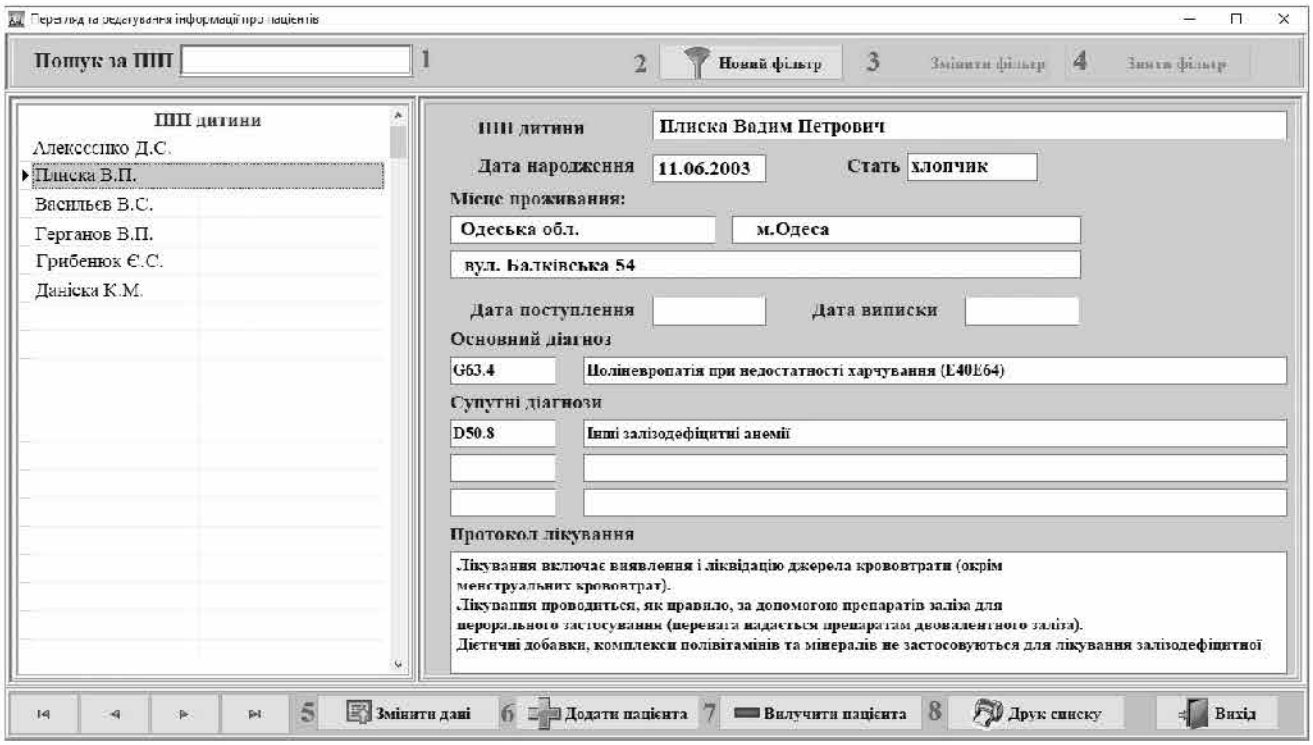


Рисунок 1.15. Форма для роботи із списком пацієнтів

Структура робочого вікна і всі ключові елементи:

1. Уведення частини ПІП для пошуку пацієнта
2. Установка нових параметрів фільтрації
3. Зміна параметрів фільтрації
4. Відключення режиму фільтрації
5. Зміна даних поточного пацієнта
6. Додавання нового пацієнта в список
7. Виведення поточного пацієнта
8. Роздрукувати список пацієнтів

Для добавлення нового пацієнта необхідно натиснути кнопку “Додати пацієнта” або скористатись комбінацією клавіш Ctrl+Ins. Після цього відкриється вікно для добавлення нового пацієнта (рис. 1.16).

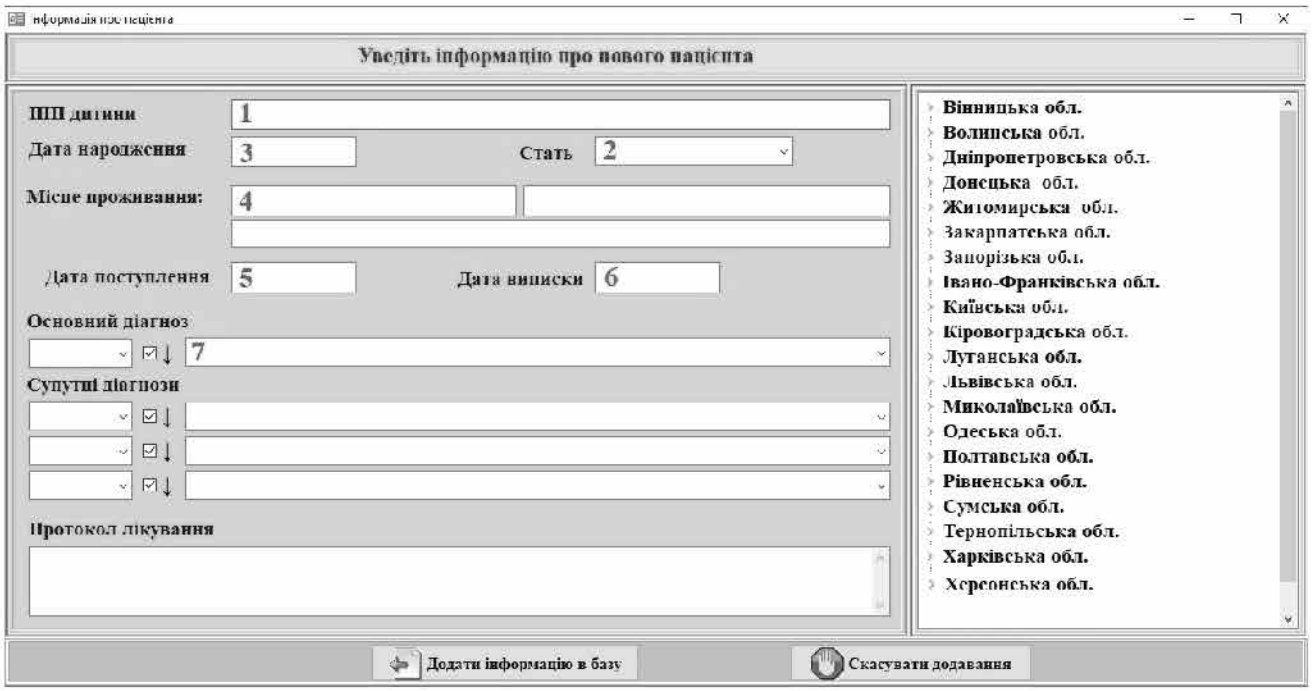


Рисунок 1.16. Форма додавання нового пацієнта

Потім необхідно заповнити такі обов'язкові дані:

1. ПІП дитини
2. Стать
3. Дата народження
4. Регіон проживання
5. Дата поступлення на лікування
6. Дата виписки
7. Основний діагноз

Інші показники (супутні діагнози і протокол лікування) не є обов'язковими.

Щоб зберегти введену інформацію в базі даних, необхідно натиснути кнопку «Додати інформацію в базу даних». Вибір кнопки «Скасувати доповнення» скасовує збереження введеної інформації і повертає в режим перегляду списку пацієнтів.

Для редагування інформації про обраний пацієнта необхідно натиснути кнопку «Змінити дані» або скористатися комбінацією клавіш Ctrl + E. Після цього відкриється вікно редагування даних пацієнта (рис. 1.17).

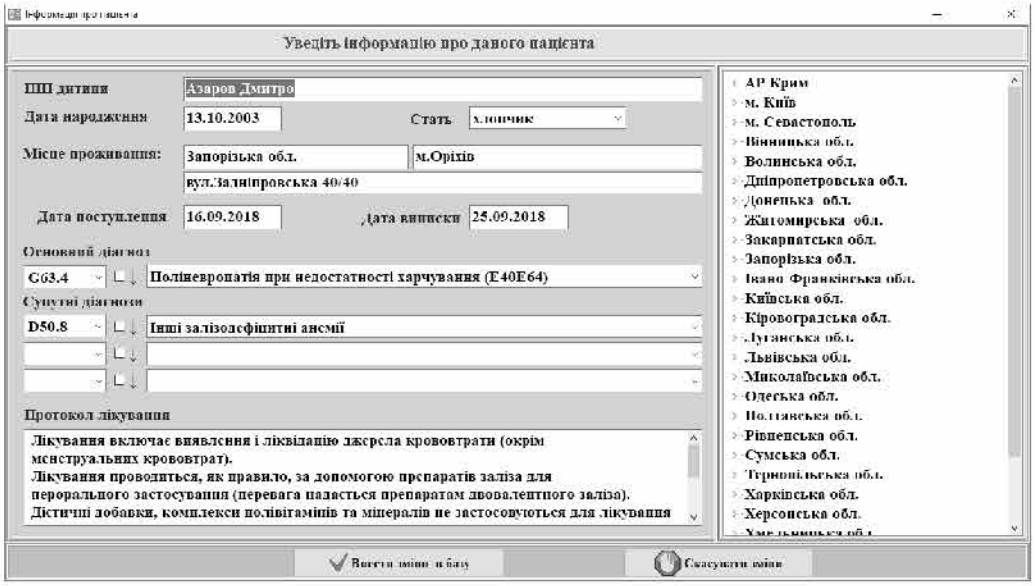


Рисунок 1.17. Форма редагування даних пацієнта

Схема алгоритму додавання та редагування даних у електронній реєстратурі (рис.1.18.)

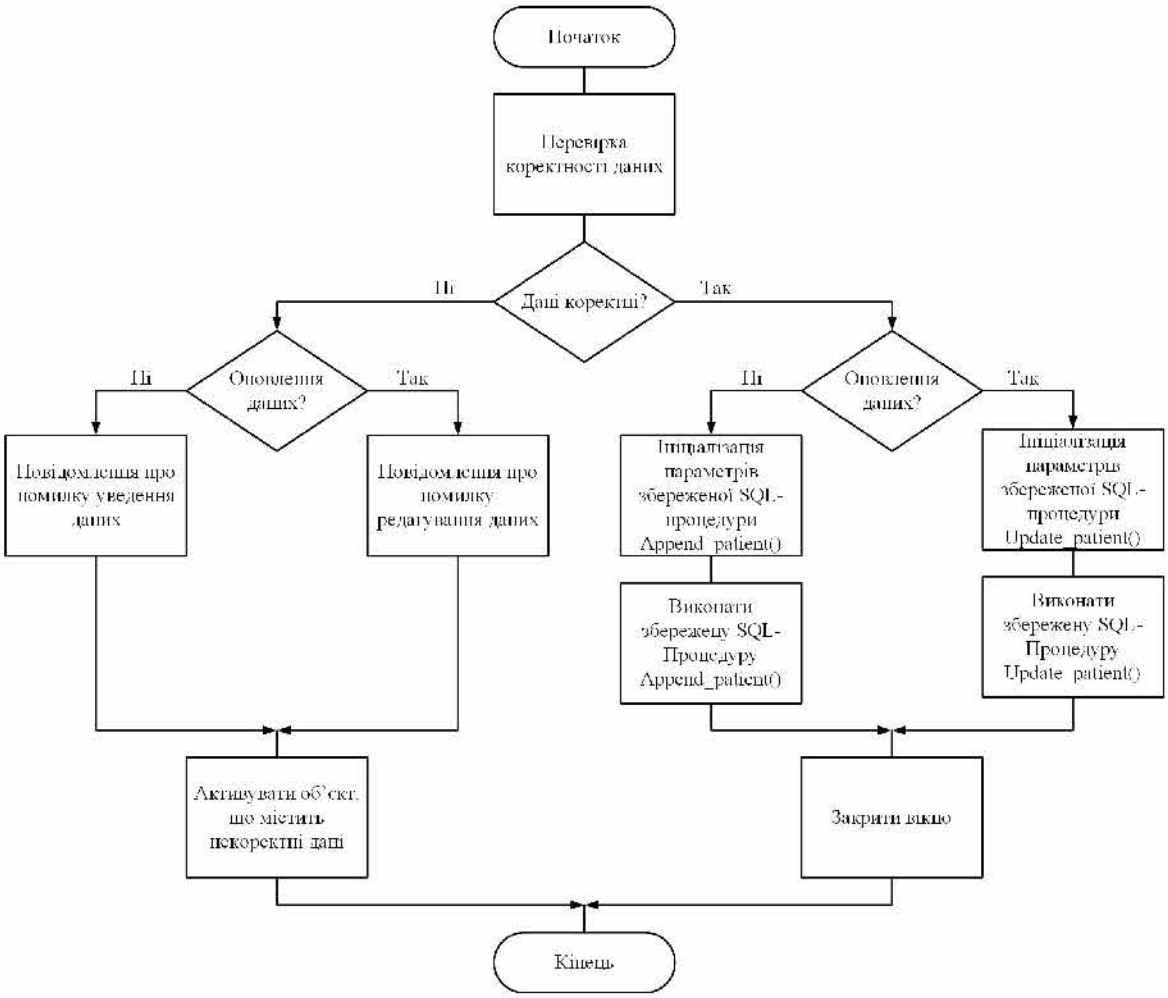


Рисунок 1.18. Алгоритм додавання та редагування даних

Зміна інформації припускає повне або часткове зміна даних за всіма показниками даного пацієнта. Щоб зберегти введену інформацію в базі даних, необхідно натиснути кнопку «Внести зміни в базу даних». Вибір кнопки «Скасувати зміни» скасовує збереження введеної інформації і повертає в режим перегляду списку пацієнтів.

Щоб видалити поточного пацієнта з бази даних, натисніть кнопку «Видалити пацієнта» або натисніть комбінацію клавіш Ctrl + Del і підтвердіть дію (рис. 1.19).

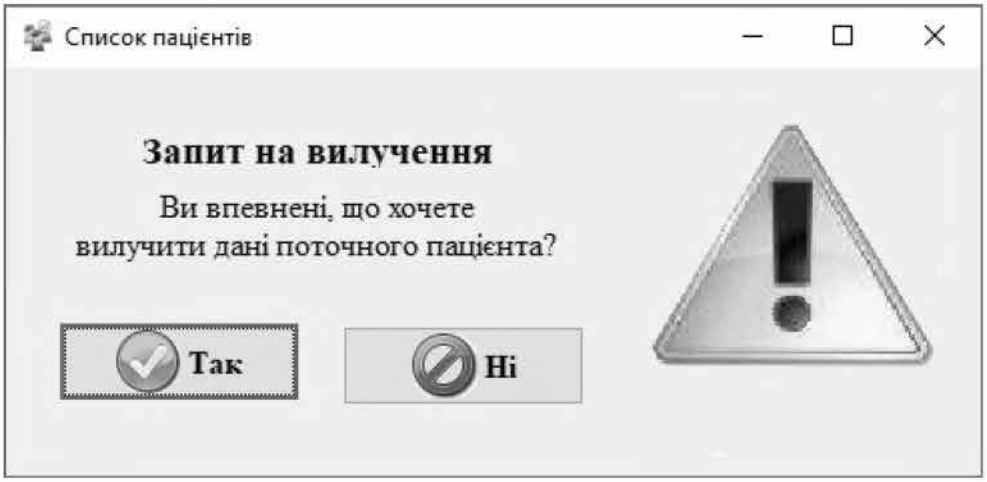


Рисунок 1.19. Форма підтвердження видалення вибраного пацієнта

Для пошуку пацієнта достатньо увести частину його ПІП у поле для контекстного пошуку. Якщо у списку пацієнтів присутній запис, ПІП якої містить задану послідовність символів, курсор буде поміщений в позицію знайденої запису. Якщо збігів немає, курсор буде поміщений в початок списку.

Фільтрація списку забезпечує вибір записів з усієї бази даних пацієнтів, які відповідають заданим критеріям. Щоб створити новий фільтр, вам потрібно натиснути кнопку «Новий фільтр». Після цього відкриється вікно налаштування параметрів фільтрів відбору пацієнтів, яке показано на рис. 1.20.

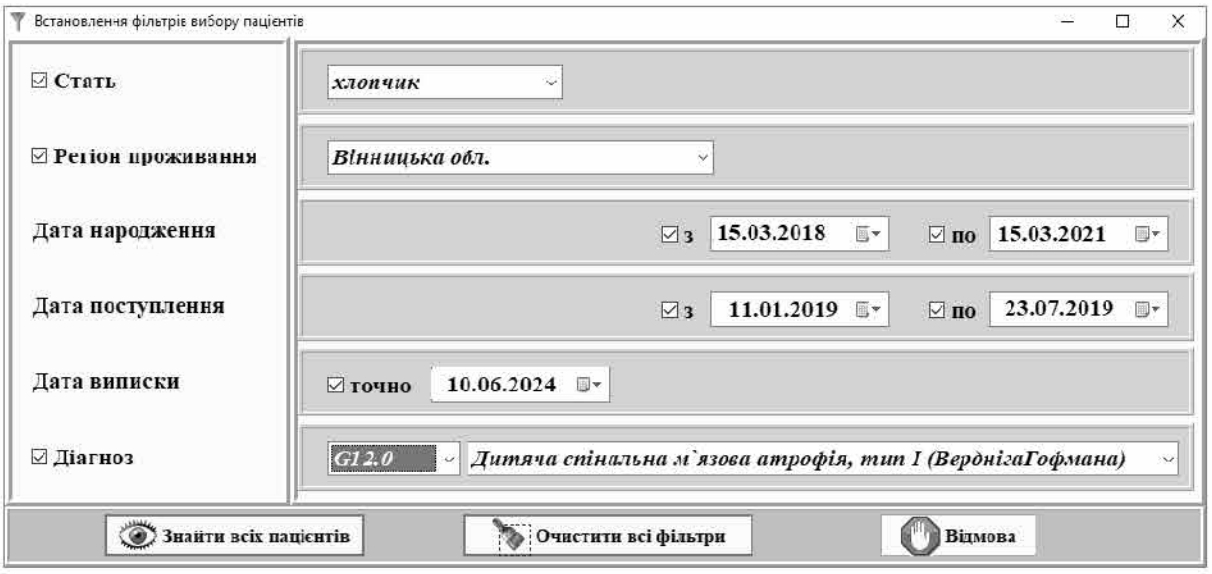


Рисунок 1.20. Форма встановлення параметру фільтрів

У разі відсутності пацієнтів, які відповідають критеріям відбору, з'явиться відповідне повідомлення. Список, отриманий в результаті фільтрації, можна роздрукувати. Зміна параметрів фільтрів здійснюється кнопкою «Змінити фільтр».

Роздрукувати список всіх пацієнтів або список пацієнтів, отриманих в результаті фільтрації. Для цього натисніть кнопку «Роздрукувати список». Приклад фрагмента списку пацієнтів, відфільтрованих за статтю і діагнозу (хлопчики з діагнозом E03.0 (природжений гіпотиреоз з дифузним зобом)) наведено на рис. 1.21.

Список пацієнтів

Результат фільтрації по: 'хлопчик'; 'Природжений гіпотиреоз з дифузним зобом';

Ковалів Р.С. 2004 р.н. поступлення: 11.03.2021 виписка: 27.05.2021	Житомирська обл.	м.Овруч	м.Житомир
	G56.4	Каузалгія	МРТ
	E03.0	Природжений гіпотиреоз з дифузним зобом	
	E45.0	Затримка розвитку внаслідок білковоенергетичної ка	
Купрій І.А. 2003 р.н. поступлення: 16.01.2021 виписка: 21.05.2021	Харківська обл.	Харківський район	Харківська обл.
	G56.4	Каузалгія	МРТ
	E03.0	Природжений гіпотиреоз з дифузним зобом	

Рисунок 1.21. Фрагмент відсортованого списку пацієнтів для друку

1.5.4 Робота з довідниками

У програмі два довідника: адміністративний поділ України і довідник нозологій. Для перегляду їх змісту необхідно вибрати в головному меню пункт “Сервіс” → “Адміністративний поділ України” або “Сервіс” → “Довідник нозологій” відповідно (рис. 1.22).

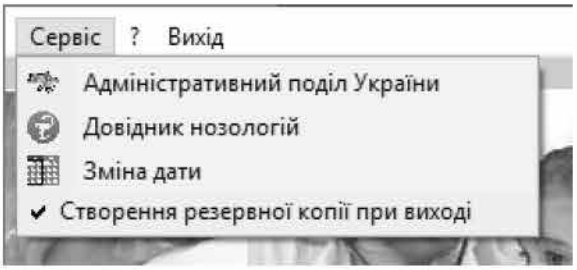


Рисунок 1.22. Фрагмент головного меню системи для вибору довідника

Для довідника адміністративного поділу України можна:

- додати нову адміністративну одиницю;
- редагувати існуючі адміністративні одиниці;
- вилучати адміністративні одиниці, що вже існують на момент розробки, заборонено. Видаляти можна лише користувацькі адміністративні одиниці.

Для довідника нозологій можна:

- динамічно керувати наявністю необхідного діагнозу з шифром згідно

1.6 Результати статистичної обробки даних

Основним результатом статистичної обробки даних є звіти. Система формує такі звіти по обхвату лікуванням:

- за віковими категоріями;
- за шифрами нозологій;
- за регіонами проживання.

Звітні періоди:

- місяць;
- квартал;
- півріччя;

– 9 місяців;

– рік.

За типами звітів і їх періодів розроблена система надає користувачеві гнучкий механізм формування медичної статистичної звітності. За датою, зазначеної користувачем, система визначає повноту звітів за наступним принципом:

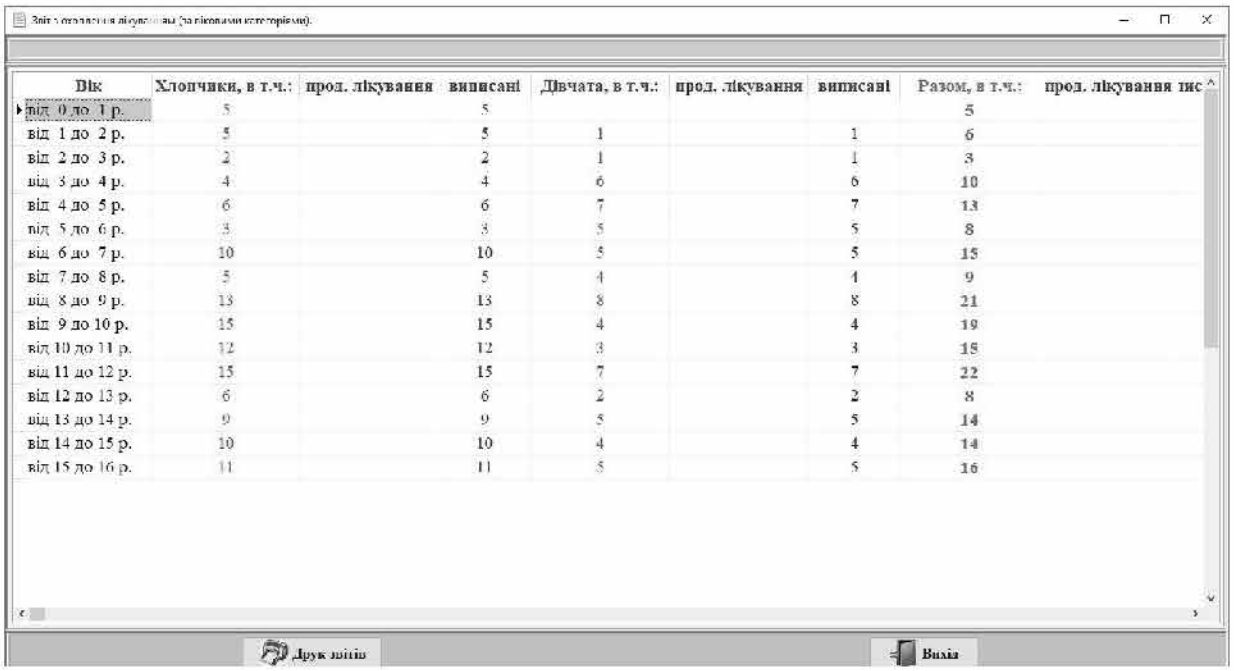
– місячні звіти формуються завжди;
– звіти за поточний квартал, півріччя та 9 місяців формується лише в тому випадку, коли користувач вказав дату, що є останнім місяцем кварталу, півріччя чи 9 місяців відповідно;
– річний звіт формується при виборі користувачем будь-якого дня грудня місяця.

Після формування звітів їх можна переглянути та надрукувати.

Формування звітів відбувається після вибору відповідного зображення у головному вікні або після виконання команди “Звітність” → “Сформувати звітність”.

1.6.1 Звіт за віковими категоріями

Звіт з охоплення лікуванням за віковими категоріями, зображений на рис.1.23, містить інформацію про кількість дітей віком до 16-ти років (з кроком в 1 рік), які в даному звітному періоді поступили на лікування.



Вік	Хлопчики, в т.ч.:	прод. лікування	виписані	Дівчата, в т.ч.:	прод. лікування	виписані	Разом, в т.ч.:	прод. лікування	вис
від 0 до 1 р.	5		5				5		
від 1 до 2 р.	5		5	1		1	6		
від 2 до 3 р.	2		2	1		1	3		
від 3 до 4 р.	4		4	6		6	10		
від 4 до 5 р.	6		6	7		7	13		
від 5 до 6 р.	3		3	5		5	8		
від 6 до 7 р.	10		10	5		5	15		
від 7 до 8 р.	5		5	4		4	9		
від 8 до 9 р.	13		13	8		8	21		
від 9 до 10 р.	15		15	4		4	19		
від 10 до 11 р.	12		12	3		3	15		
від 11 до 12 р.	15		15	7		7	22		
від 12 до 13 р.	6		6	2		2	8		
від 13 до 14 р.	9		9	5		5	14		
від 14 до 15 р.	10		10	4		4	14		
від 15 до 16 р.	11		11	5		5	16		

Друк звіту | Вихід

Рисунок 1.23. Фрагмент звіту з охоплення лікуванням за віковими категоріями

Хлопчики та дівчатка фіксуються окремо. На рис.1.24 зображено приклад діаграми охоплення лікуванням за віковими категоріями та статтю.

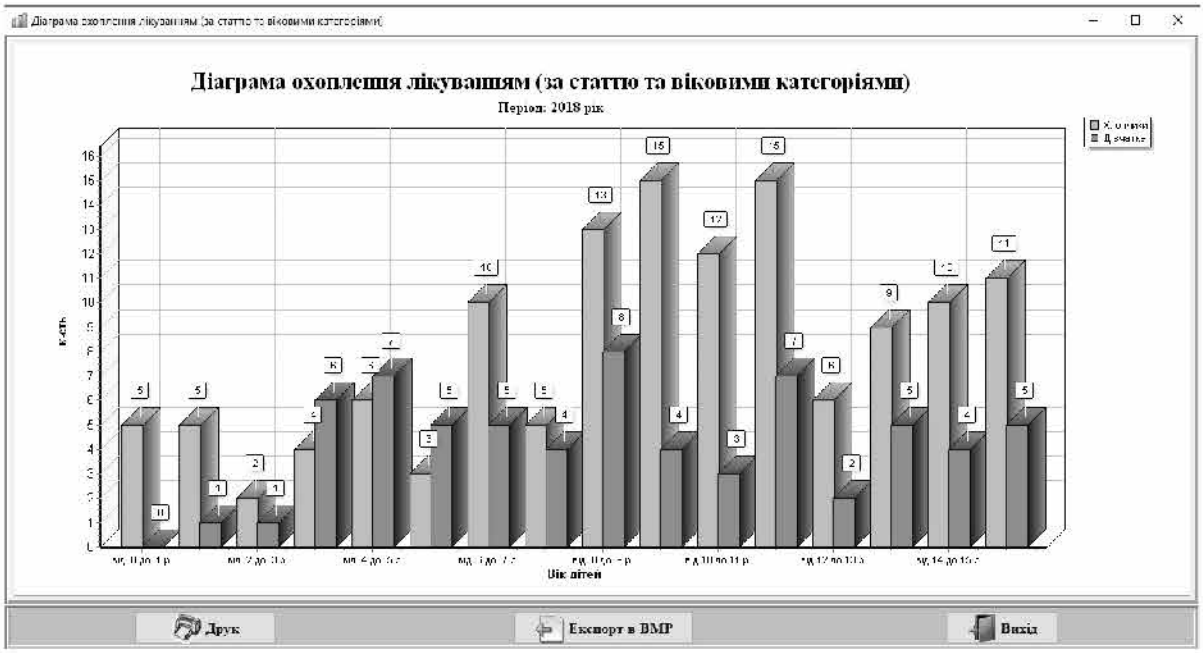


Рисунок 1.24. Діаграма охоплення лікуванням за віковими категоріями та статтю

1.6.2 Звіт за шифрами нозологій

Звіт з охоплення лікуванням за шифрами нозологій, зображений на рис.1.25, містить інформацію про кількість дітей віком до 16-ти років (з кроком в 1 рік), які в даному звітному періоді поступили на лікування.

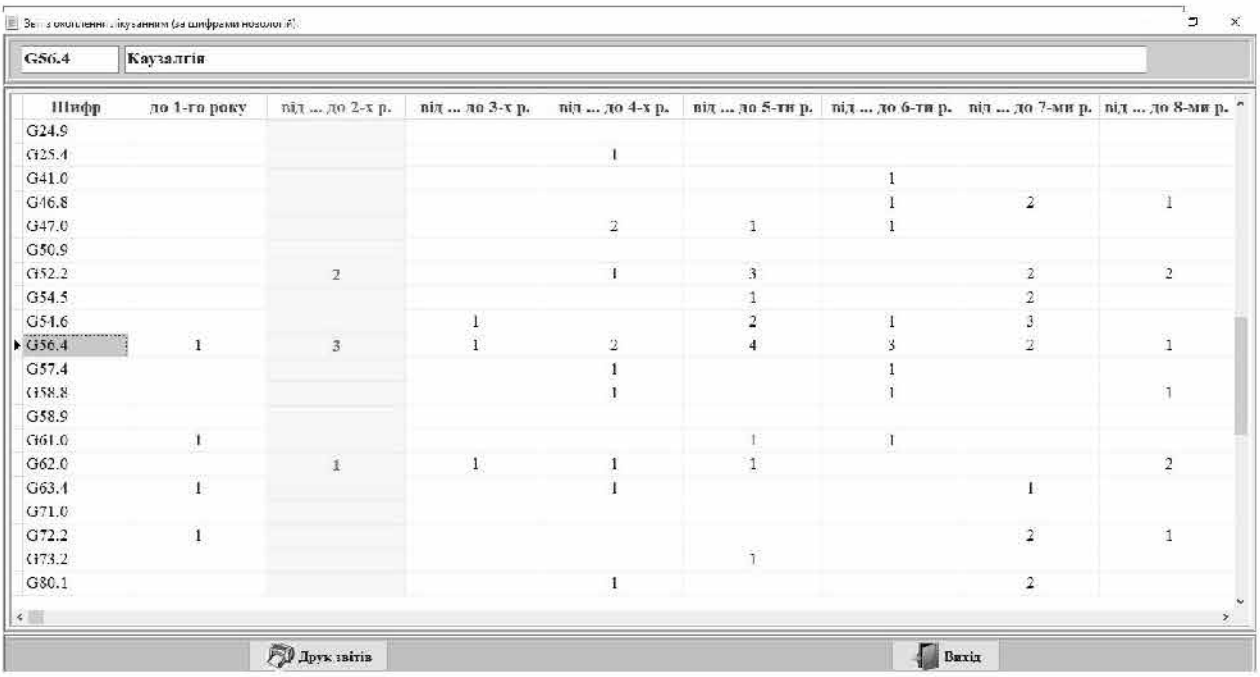


Рисунок 1.25. Фрагмент звіту з охоплення лікуванням за шифрами нозологій

Вибір віку або діагнозу для побудови діаграми виконується кліком по заголовку потрібної колонки або рядка. На рис.1.26 зображено приклад діаграми охоплення лікуванням пацієнтів вказаного діагнозу (G56.4) за віковими категоріями.

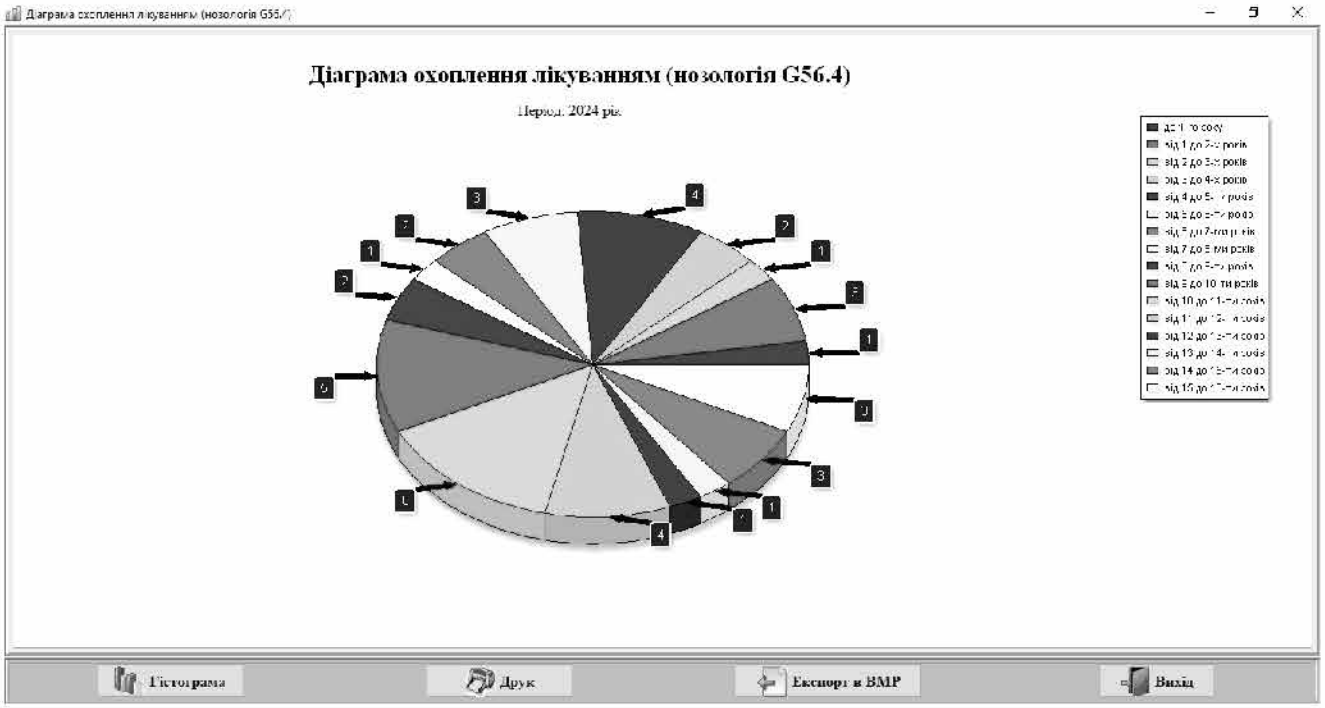


Рисунок 1.26. Діаграма охоплення лікуванням пацієнтів вказаного діагнозу (G56.4) за віковими категоріями

1.6.3 Звіт за регіонами проживання

Звіт з охоплення лікуванням за регіонами проживання, зображений на рис.1.27, містить інформацію про кількість дітей віком до 16-ти років (з кроком в 1 рік), які в даному звітному періоді поступили на лікування.

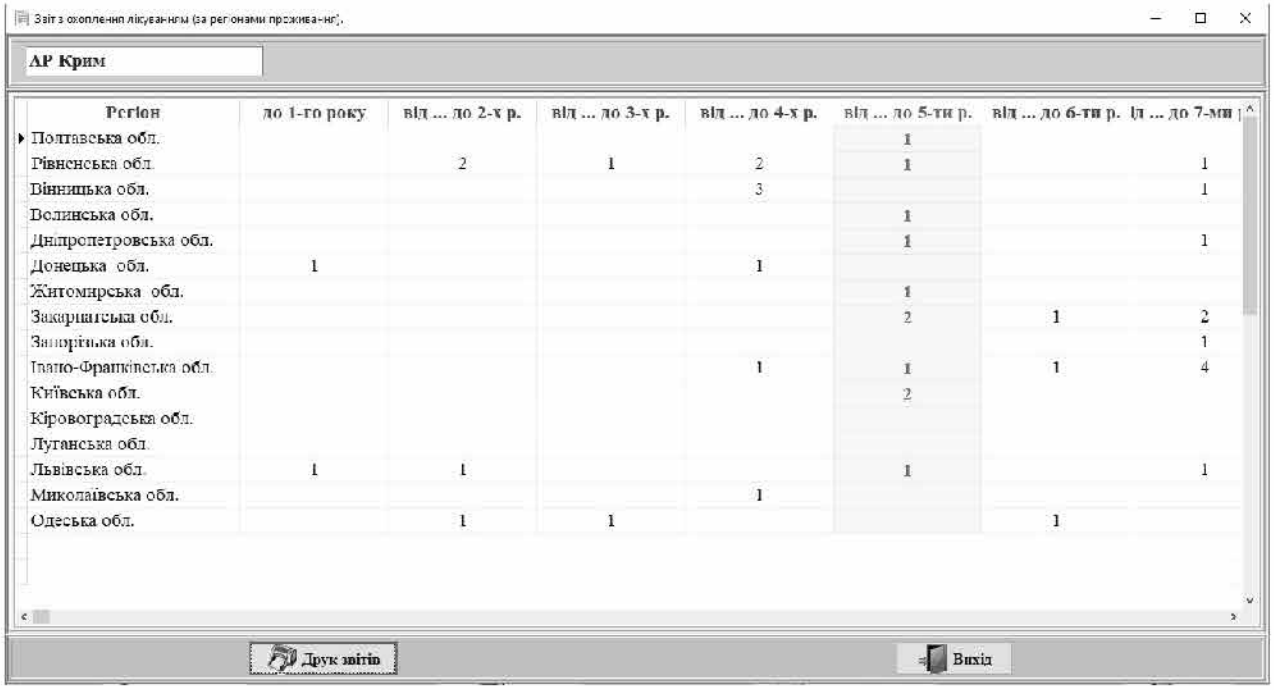


Рисунок 1.27. Фрагмент звіту з охоплення лікуванням за регіонами проживання

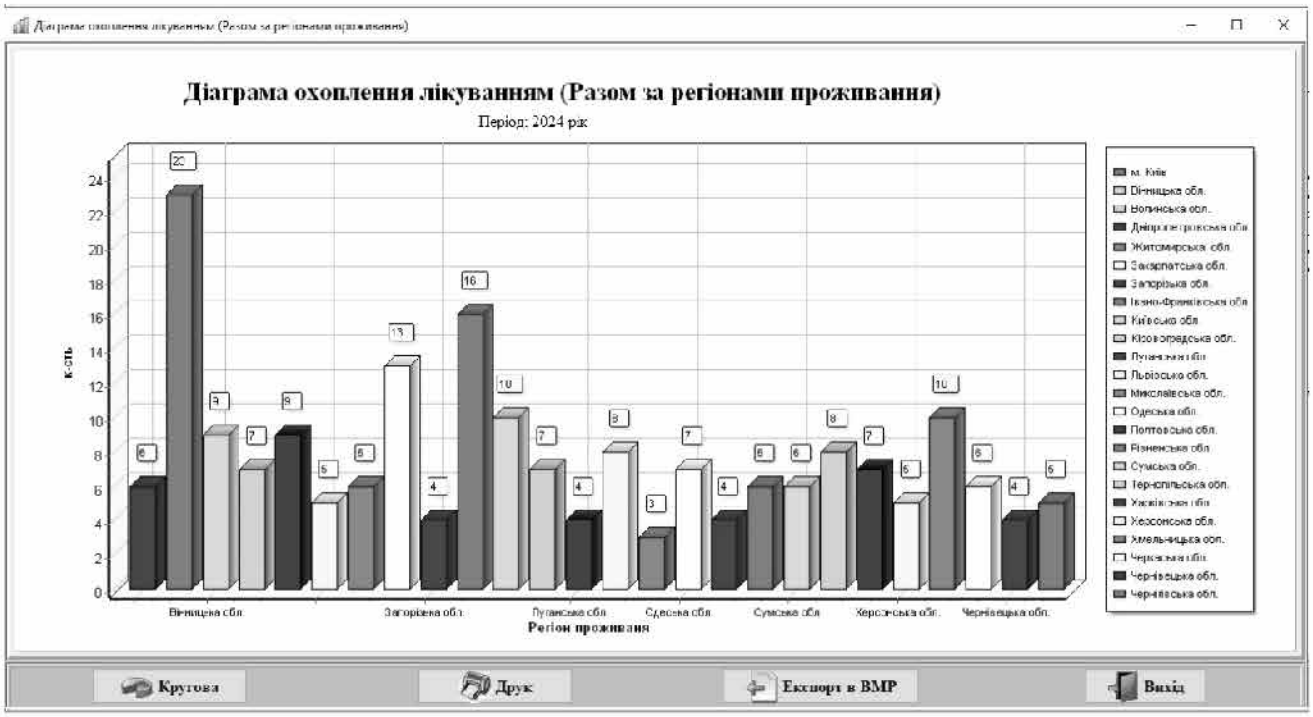


Рисунок 1.28. Діаграма охоплення лікуванням пацієнтів за регіонами проживання

Вибір віку для діаграми "... за регіонами вказаного віку" виконується кліком по заголовку потрібної колонки. На рис. 1.28 зображено діаграма з охоплення лікуванням за регіонами.

Всі сформовані звіти можна переглядати та друкувати.

Для друку (Ctrl+P) звітів в традиційній формі потрібно вибрати команду "Друк звітів" → "Звіт в традиційній формі" (кнопка "Друк звітів").

На рисунку 1.29 наведено фрагмент звіту в традиційній формі, призначеного для друку.

Вік	Хлопчики			Дівчатка			Всього		
	продовжують лікування	закінчили лікування	Разом	продовжують лікування	закінчили лікування	Разом	продовжують лікування	закінчили лікування	Разом
від 0 до 1 р.		5	5					5	5
від 1 до 2 р.		5	5		1	1		6	6
від 2 до 3 р.		2	2		1	1		3	3
від 3 до 4 р.		4	4		6	6		10	10
від 4 до 5 р.		6	6		7	7		13	13
від 5 до 6 р.		3	3		5	5		8	8
від 6 до 7 р.		10	10		5	5		15	15
від 7 до 8 р.		5	5		4	4		9	9
від 8 до 9 р.		13	13		8	8		21	21
від 9 до 10 р.		15	15		4	4		19	19
від 10 до 11 р.		12	12		3	3		15	15
від 11 до 12 р.		15	15		7	7		22	22
від 12 до 13 р.		6	6		2	2		8	8
від 13 до 14 р.		9	9		5	5		14	14
від 14 до 15 р.		10	10		4	4		14	14
від 15 до 16 р.		11	11		5	5		16	16
РАЗОМ		131	131		67	67		198	198

Рисунок 1.29. Фрагмент звіту, призначеного для друку

Для перегляду звітів у вигляді діаграми потрібно вибрати команду "Друк звітів" → "Звіти у формі діаграми" або натиснути кнопку "Друк звітів" і уточнити тип діаграми та данні для її побудови (кнопки "Кругова" або "Гістограма").

При натисненні кнопки "Експортувати у файл" або за допомогою комбінації клавіш Ctrl+E поточну діаграму можна експортувати у файл.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

Програма створена на мові програмування C++ у інтегрованому середовищі візуальної розробки Embarcadero RAD Studio. Ефективність кожного програмного продукту залежить від якості та ефективності процесу розробки. Якість програмного продукту визначається з кількох поглядів: перш за все, з позиції користувача, використання ресурсів та відповідність вимогам до програмного забезпечення.

Оцінка якості програмного продукту з точки зору користувача проводиться за допомогою таких показників, як обсяг оперативної пам'яті, витрати часу на обробку, пропускна спроможність каналів передачі даних. Також важливо враховувати трудомісткість та вартість процесу його створення.

2.2 Визначення трудомісткості розробки програмного забезпечення

Тривалість розробки програмного продукту залежить від його обсягу, трудомісткості розробки, кваліфікації виконавців, а також планових термінів, визначених умовами ринку. Методом структурної аналогії по відповідних каталогах аналогів програмного забезпечення визначається обсяг програмних засобів, у тисячах умовних машинних команд програми аналога

У таблиці 2.1 представлені аналоги програмного забезпечення, функції яких, у більшому або меншому ступені, виконує розроблений програмний продукт. Для нашого варіанта виділено сірим кольором.

Таблиця 2.1. Аналоги програмного забезпечення

Найменування ПЗ	*Обсяг функції ПП = V_0, ум. машинних команд*
1. ПП СУБД	1300 – 8600
2. ПП введення інформації	1800 – 8800
3. ПП оптимізаційних розрахунків	13000 – 10200

Вибравши аналог ПП, що містить Vo в умовних машинних командах, трудомісткості визначати на основі табл.2.2

Таблиця 2.2. Норми часу

Обсяг ПП, тис.умов.машинних команд	Норма часу, люд/год
1.00	229
2.00	244
3.00	262

На підставі отриманого значення, по довіднику, визначається укрупнена норма часу на розробку аналога програмного забезпечення (коректується поправочним коефіцієнтом враховуючої умови розробки ПП, тобто в умовах комп'ютера, Кк=0,7÷0,8): Тар = 244 х 0,8 = 195.20 (люд/годин).

Трудомісткість програмного продукту визначається по кожному етапу розробки окремо на підставі трудомісткості аналога з урахуванням складності

розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул:

$$T_{ТЗ} = T^a p \times L_1 \times K_H \qquad (2.1)$$

$$T_{ТП} = T^a p \times L_2 \times K_H \qquad (2.2)$$

$$T_{РП} = T^a p \times L_3 \times K_H \times K_T \qquad (2.3)$$

Для розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розробки (див. табл. 2.2.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.3.);

Кт – поправочний коефіцієнт, що враховує ступінь використання в розробці типових програм (див. табл. 2.4.).

Таблиця 2.2. Значення питомих коефіцієнтів трудомісткості стадії в загальній трудомісткості розробки ПП

Код стадії	Ступінь новизни		
	А	Б	В
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Для нашого варіанта виділено сірим кольором.

Таблиця 2.3. Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $К_н$
А	Принципово нові ПП	1,75 – 1,2
Б	ПП – розвиток визначеного параметричного ряду	1,0 – 0,8
В	ПП маючий аналог	0,7

Для нашого варіанта виділено сірим кольором.

Таблиця 2.4. Значення коефіцієнта ступеня використання в розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПП типовими програмами, %	Значення $К_т$
60 і вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Для нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=Та*L_1*К_н=195,2*0,12*0,7=16,40 \text{ (люд/годин)} \quad (2.1)$$

Трудомісткість розробки технічного проекту

$$Т_{тп}=Та*L_2*К_н=195,2*0,11*0,7=15,04 \text{ (люд/годин)} \quad (2.2)$$

Трудомісткість розробки робочого проекту

$$Т_{рп}=Та*L_3*К_н*К_т=195,2*0,61*0,7*0,7=58,35 \text{ (люд/годин)} \quad (2.3)$$

Для подальших розрахунків визначили кількість папера, витраченого на кожен етап: технічне завдання $N_{тз}$=2 (стр), розробка ТП $N_{тп}$=15(стр), розробка робочого проекту $N_{рп}$=25 (стр), пояснювальна записка відповідно $N_{пз}$=50 (стр).

Таблиця 2.5. Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин.		
1	2	3	4
1.ТЗ	$T_{Pтз}$=16,40	$T_{кк}=0,7*N_{тз}$= 0,7*2=1,4	$T_{нк}=0,15*N_{тз}$=0,15*2=0,30
2.Розробка ТП	$T_{Pтп}$=15,04	$T_{кк}=0,7*N_{тп}$=0,7*15=10,50	$T_{нк}=0,15*N_{тп}$=0,15*15=2,25
3.Розробка РП	$T_{Ррп}$=58,35	$T_{кк}=0,7*N_{рп}$=0,7*25=17,5	$T_{нк}=0,15*N_{рп}$=0,15*25=3,75
4.Розробка ПЗ	$T_{пз}=1,5*N_{пз}$=1,5*50 =75	$T_{кк}=0,7*N_{тз}$=0,7*50=35	$T_{нк}=0,15*N_{пз}$=0,15*50 =7.5
Усього, в т.ч.:	230,2		
- на розробку	$\Sigma T_р$=152		
- контроль керівника		$\Sigma T_{кк}$= 64,4	
- нормоконтроль			$\Sigma T_{нк}$=13,8

2.3 Розрахунок ціни програмного продукту

У цьому розділі для визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години і витрати на розробку ПП. Розрахунок основної заробітної плати виконавців приведений у таблиці 2.7. Відповідно до статті 8 «Закону про Державний бюджет України на 2024» встановлено мінімальну заробітну плату у місячному розмірі з 1 січня 2024 року - 8000 гривень; мінімальну погодинну тарифну ставку – 41.26 грн.

Таблиця 2.6. Розрахунок основної заробітної плати виконавців

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	152	39.26	5958,15
2.Контроль керівника	65	38,50	2502,50
3.Нормоконт-роль	14	38,50	539,00
Усього	-	-	ΣЗо= 8999,15

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо в таблицю 2.7.

Таблиця 2.7. Розрахунок матеріальних витрат на розробку ПП

Найменування матеріальних витрат	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	60	2.60	160,0
Разом	-	-	-	$В_{мі}$ =160,0
Транспортно– заготівельні витрати (10%)				$Втр\_з = 0,1 \times Вм1$ = 0,1х160.0=16,0
Усього				$В_м=В_{мі}+В_{тр\ з}$=176,0

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості в цілому ПП за формою, приведеною в таблиці 2.8.

Таблиця 2.8. Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	176,0	$В_м$ (див. табл. 2.7)
2. Основна заробітна плата	8999,15	$З_о$ (див. табл. 2.6)
3.Додаткова заробітна плата	1349,87	$Зд = 0,15 \times Зо$ =8999,15х0,15
4.Відрахування до єдиного фонду соціального внеску	2276,78	$Вє.с.в. = 0,22 \times (Зо + Зд)$ =0,22х (8999,15+ 1349,87)
5. Накладні витрати	2699,75	$Внак. = 0,3 \times Зо$ =0,3 х 8999,15
6. Повна собівартість	15501,55	$С_{пов}$= $В_м$ +$З_о$ + $З_д$ +Вє.с.в. + Внак.= 176,0+8999,15+1349,87+2276,78+ 2699,75

Розмір прибутку, що включається в ціну, визначаємо по наступній формулі:

$$П=(С_п*Р)/100= (15501,55*10)/100 = 1550,15грн \qquad (2.4)$$

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о =С_п+П =15501,55+1550,15= 17051,70 \text{ грн} \qquad (2.5)$$

Податок на додану вартість визначаємо по наступній формулі:

$$ПДВ = 0.2 * Ц_о = 17051,70 *0,2 = 3410,34грн; \qquad (2.6)$$

Ціна реалізації розробленого програмного продукту, становитиме:

$$Ц_р=Ц_о+ПДВ = 17051,70 + 3410,34 = 20462,04 \text{ грн} \qquad (2.7)$$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

3.1 Вступ

Охорона праці, як соціальний чинник, відіграє на підприємстві важливу, роль оскільки, якими б важливими не були трудові здобутки, вони не можуть компенсувати людині втраченого здоров'я, а тим більше життя. Те і інше дається лише один раз. Необхідно пам'ятати, що внаслідок нещасних випадків та аварій гинуть на виробництві не просто робітники та службовці, на підготовку яких держава вкладає значні кошти, а перш за все люди – годувальники сімей, батьки та матері дітей. Незадовільний стан охорони праці відображається на економіці держави. Служба охорони праці створюється на підприємствах незалежно від форми власності та видів діяльності для виконання правових, організаційно-технічних, санітарно-гігієнічних, соціально-економічних і лікувально-профілактичних заходів, спрямованих на запобігання нещасних випадків, професійних захворювань і аваріям в процесі праці. Основна мета всіх цих заходів – створити на підприємстві безпечні та здорові умови праці.

У розділі охорона праці дипломного проекту наведені характеристики приміщень. До розгляду взято робоче місце програміста.

3.2 Аналіз небезпечних та шкідливих чинників, що впливають на працівника

Оператори ПК і програмісти зіштовхуються із впливом таких фізично небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, недостатня освітленість робочої зони, електричний струм та інші. Тому на робочому місці програміста повинні бути створені умови для високопродуктивної праці.

Перетворення і обробка інформації проводиться за допомогою ПК. Робота може кваліфікуватися як робота оператором ЕОМ.

3. 3 Розробка заходів з охорони праці

3.3.1 Виробничі приміщення

При плануванні виробничого приміщення врахована санітарна характеристика виробничих процесів, дотримуються норми корисної площі для працюючих, а також нормативи площ для розташування устаткування, що забезпечують безпечну роботу та зручне обслуговування устаткування.

Об'ємно-планувальні рішення будівель та приміщень для роботи з ВДТ мають відповідати вимогам ДСанПіН 3.3.2.007-98. Розміщення робочих місць З ВДТ ЕОМ і ПЕОМ у підвальних приміщеннях, на цокольних поверхах заборонено. Площа на одне робоче місце становить не менше ніж 6,0 м^2, а об'єм – не менше ніж 20,0м^3.

Виробничі приміщення повинні обладнуватися шафами для зберігання документів, полицями, стелажами, тумбами тощо, з урахуванням вимог до плащі приміщення.

У приміщеннях з ВДТ слід щоденно робити вологе прибирання. Приміщення повинні бути оснащені аптечками першої медичної допомоги.

3.3.2 Мікроклімат робочої зони працівників, вентиляція

У виробничих приміщеннях на робочих місцях з ВДТ мають забезпечуватись оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря (ДСанПіН 3.3.2.007-98).

Таблиця 3.1. Норми мікроклімату для приміщень з ВДТ ОТ

Пора року	Категорія робіт	Температура повітря, С, не більше	Відносна вологість повітря %	Швидкість руху повітря, м/с
Холодна	Легка-1а	22-24	40-60	0,1
	Легка-1б	21-23	40-60	0,1
Тепла	Легка-1а	23-25	40-60	0,1
	Легка-1б	22-24	40-60	0,1

Рівні позитивних і негативних іонів у повітрі приміщень з ВДТ мають відповідати санітарно-гігієнічним нормам № 2152-80.

Таблиця 3.2. Рівні позитивних і негативних іонів у повітрі

Рівні	Число іонів в 1 $см^3$ повітря	Число іонів в 1 $см^3$ повітря
	n+	n-
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально допустимі	50000	50000

3.3.3 Освітлення робочого місця, шум, вібрація

Штучне освітлення в приміщеннях з робочими місцями, обладнаними ВДТ має здійснюватися системою загального рівномірного освітлення. У виробничих та адміністративних приміщеннях, у разі переважної роботи з документами, допускається застосування системи комбінованого освітлення – крім системи загального освітлення додатково встановлюються світильники місцевого освітлення.

Значення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300-500лк.

Як джерела світла для штучного освітлення мають застосовуватись переважно люмінесцентні лампи типу ЛД. Допускається застосування ламп розжарювання у світильниках місцевого освітлення.

3.3.4 Організація робочого місця користувача ПК

Робочі місця слід так розташовувати відносно світових прорізів, щоб природне світло падало збоку, переважно зліва. При розміщенні робочих столів з ВДТ слід дотримуватися таких відстаней: між бічними поверхнями ВДТ -1,2м; від тильної поверхні одного ВДТ до екрану іншого – 2,5м.

Екран ВДТ має розташовуватися на оптимальній відстані від очей користувача, що становить 600…700 мм, але не ближче ніж за 600 мм з урахуванням розміру літерно-цифрових знаків і символів.

Клавіатуру розташовують на поверхні столу на відстані 100…300 мм від краю, зверненого до працюючого. У конструкції клавіатури має передбачатися опорний пристрій, який дає змогу змінювати кут нахилу поверхні клавіатури у межах 5…15^0.

При оснащенні робочого місця лазерним принтером параметри лазерного випромінювання повинні відповідати вимогам СанПіН № 5804-91.

ЕОМ ВДТ і ПК, інше устаткування, електропроводи та кабелі за виконанням і ступенем захисту мають відповідати класу зони за НПАОП 40.1-1.01-97, мати апаратуру захисту від струму короткого замикання та інших аварійних режимів. У приміщеннях, де одночасно експлуатується понад п'ять ЕОМ встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення. Не допускається підключати ЕОМ з ВДТ і ПК до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв

3.3.5 Електробезпека

Це система організаційних і технічних заходів та засобів, що забезпечують захист людей від шкідливої і небезпечної дії електричного струму, електричної дуги, електричного поля і статичної електрики.

Основні технічні засоби і заходи забезпечення електробезпеки при нормальному режимі роботи електроустановок включають: ізоляцію струмовідних частин; недоступність струмовідних частин; блоківки безпеки; засоби орієнтації в електроустановках; виконання електроустановок, ізольованих від землі; захисне розділення електричних мереж; компенсацію ємнісних струмів замикання на землю; вирівнювання потенціалів.

Із метою підвищення рівня безпеки, залежно від призначення, умов експлуатації і конструкції, в електроустановках застосовується одночасно більшість з перерахованих технічних засобів і заходів.

Особа відповідальна за електрогосподарство призначається з числа працівників, які мають не нижче IV групи з електробезпеки та відповідний стаж

роботи для обслуговування електроустановок несе персональну відповідальність за допущення працівника використовувати в роботі електричну енергію

3.4 Пожежна безпека

Пожежна небезпека – це можливість виникнення та розвитку пожежі в будь-якій речовині, процесі, стані. Коли людина перебуває в зоні впливу пожежі, то вона може потрапити під дію наступних небезпечних та шкідливих факторів: токсичні продукти згорання, вогонь, підвищена температура середовища, дим, недостатність кисню, руйнування будівельних конструкцій, вибухи, паніка.

Усі працівники повинні вміти користуватись наявними вогнегасниками, іншими первинними засобами пожежогасіння, знати місце їх знаходження.

До первинних засобів пожежогасіння відносяться:

- вогнегасники;
- пожежний інвентар (покривала з негорючого теплоізоляційного полотна, грубововняної тканини або повсті;
- ящики з піском;
- бочки з водою, пожежні відра, совкові лопати) та пожежний інструмент.

Пожежні щити (стенди) встановлюються на території об'єкта з розрахунку один щит (стенд) на площу 5000 м2.

Ящики для піску повинні мати місткість 0,5, 1,0 або 3,0 м3 та бути укомплектованими совковою лопатою. Вибір типу та визначення необхідної кількості вогнегасників здійснюється відповідно до Типових норм належності вогнегасників, затверджених наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи від 02.04.2004 № 151 та зареєстрованих у Міністерстві юстиції України 29.04.2004 за № 554/9153. Евакуаційні шляхи і виходи повинні втримуватися відповідно до НАПБ А.01.001 2004, бути вільними, нічим не захаращуватися і у разі виникнення пожежі забезпечувати безпеку під час евакуації всіх людей, які перебувають у приміщеннях будівель та споруд.

ВИСНОВКИ

Дипломна робота містить загальні відомості про медичну статистику та звітність, найбільш відомі на сьогодні існуючі рішення на українському ринку. Крім того, проаналізовано основні переваги та недоліки описаних аналогів.

Робота містить обґрунтування вибору архітектури системи, загальну характеристику клієнт-серверної архітектури, опис методів реалізації клієнтської та серверної частин. У розділі міститься аргументоване обґрунтування щодо вибору формату зберігання даних, способу доступу до бази даних, вибору програмного середовища для написання комп'ютерних програм комплексу.

Описано структуру бази даних та її елементів, наведено перелік модулів проекту, їх функціонал і використання, перераховано основні користувацькі функції, створені для забезпечення інтерфейсу та описано основні компоненти для взаємодії користувацького інтерфейсу з сервером.

Виконано опис основних функцій розробленої системи, описано режими роботи системи та наведено опис відомостей та звітів, що формуються системою. У дипломі міститься детальна інструкція користувача на всіх етапах роботи з системою, починаючи з встановлення необхідного програмного забезпечення. Виконано розрахунок економічної частини проекту. Розглянуті питання стосовно охорони праці.

Основним результатом роботи є система, що забезпечує комп'ютерну реалізацію обліку та статистичного аналізу даних пацієнтів лікарні, що в свою чергу оптимізує роботу медичного персоналу. Дана система реалізує функції, недоступні для існуючих рішень та є меншовартісною у порівнянні з описаними аналогами.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Предмет та зміст медичної статистики [Електронний ресурс] // Режим доступу: https://studfiles.net/preview/3889805/page:2/

2. ELEKS: Enterprise Software Development, Technology Consulting [Електронний ресурс] // Режим доступу: https://eleks.com

3. Dr. ELEKS | Medical inFornation system [Електронний ресурс] // Режим доступу: https://doctor.eleks.com

4. IRISMED, LLC [Електронний ресурс] // Режим доступу: http://irismed.com.ua

5. Стецюк В. З.. Робоче місце лікаря-педіатра. / В.З. Стецюк, Т.П. Іванова/ І.П. Муха/ Л.Ю. Бабінцева/ Н.В. Ольхович/ Ю.М. Чиж // Медична інформатика та інженерія. – 2018. - № 1. – С. 37-41.

6. What is Client/Server Architecture? - Definition from Techopedia [Електронний ресурс] // Режим доступу: https://bit.ly/2YzorH5

7. О модели взаимодействия клиент-сервер. Архитектура «клиент-сервер» с примерами [Електронний ресурс] // Режим доступу: https://bit.ly/2VGtqZD

8. Клієнт-серверні системи [Електронний ресурс] // Режим доступу: https://bit.ly/2EbTwsF

9. Мова програмування С++ [Електронний ресурс] // Режим доступу: http://cppstudio.com/uk/cat/274/

10. Брюс Эккель. Философия C++. Введение в стандартный C++ – СПб.: Питер, 2004. – 577 с.

11. Романчик В. С., Люлькин А. Е.. Программирование в C++ Builder. – Мн: Минск, 2007. – 128 с.

12. Начинаем работать в Borland C++ Builder [Електронний ресурс] // Режим доступу: http://citforum.ru/programming/application/cb2.shtml

13. Fast Cross-PlatForn App Development Software – Embarcadero [Електронний ресурс] // Режим доступу: https://www.embarcadero.com

14. Реляційна модель даних [Електронний ресурс] // Режим доступу: http://znaimo.com.ua/Реляційна_модель_даних

15. Сервер баз данных (SQL) [Електронний ресурс] // Режим доступу: http://www.stss.ru/solutions/database_server.html

16. Реляційна модель даних [Електронний ресурс] // Режим доступу: https://bit.ly/30mSZgZ

17. Алан Бьюли. Изучаем SQL – М. : Символ-Плюс, 2007. – 312 с.

18. Microsoft SQL Server [Електронний ресурс] // Режим доступу: https://bit.ly/2VHrCQ2

19. Документация по SQL Server [Електронний ресурс]. – Режим доступу: https://goo.gl/VrWkUY

20. КРАТКАЯ СПРАВКА О ПЛАТФОРМЕ MICROSOFT SQL SERVER [Електронний ресурс]. – Режим доступу: https://bit.ly/2Vw5R0Z

21. Справочник по Transact-SQL (компонент Database Engine) [Електронний ресурс]. – Режим доступу: https://bit.ly/2EaPTDo

22. C++ Builder Работа с базой данных через ADO [Електронний ресурс]. – Режим доступу: https://bit.ly/2HkIwLI

23. Работа с базами данных в Borland C++ Builder [Електронний ресурс]. – Режим доступу: https://bit.ly/2HsgvAO

24. Боровский А. Доступ к базам данных с помощью dbGO // Современные средства разработки Borland для Oracle и MS SQL. – Спб. БХВ-Петербург, 2007. – с.168-192

25. Міжнародна статистична класифікація хвороб та споріднених проблем охорони здоров'я. 10-й переклад. Т.1., ч.1, 2 : Пер. з англ. : Женева : Всесвітня організація охорони здоров'я, 1998.

ДОДАТОК А. Програмний код

Файл IP54_Chyzh.cpp

```
#pragma hdrstop
#include "IP54_ChyzhPCH1.h"
//---------------------------------------------------------
USEFORM("DataModule.cpp", DM);                    // модуль даних
USEFORM("Main.cpp", MainForm);                    // головне вікно проекту
USEFORM("About.cpp", FormAbout);                  // форма інформації про проект
USEFORM("Help_Admin.cpp", FormHelpsAdmin);              // форма адміністративного поділу України
USEFORM("Help_Diag.cpp", FormHelpsDiagnoz); // форма довідника шифрів нозологій
USEFORM("FormEdit.cpp", FormEditInfo);            // форма перегляду та редагування інформації про пацієнтів відділення
USEFORM("Append_Patient.cpp", Form_Append);             // форма добавлення та зміни інформації про пацієнтів відділення
USEFORM("FilterSet.cpp", FormFilertSet);          // форма фільтрації списку пацієнтів відділення
USEFORM("Form_ZvitYear.cpp", FormZvitYear);  // форма звіту з охоплення лікування за віковими категоріями
USEFORM("Form_ZvitRegion.cpp", FormZvitRegion); // форма звіту з охоплення лікування за регіонами проживання
USEFORM("Form_ZvitShifr.cpp", FormZvitShifr);  // форма звіту з охоплення лікування за діагнозами нозологій
USEFORM("Chart_Region.cpp", ChartRegoin);     // звіт з охоплення лікування за регіонами проживання у формі діаграми
USEFORM("Chart_Year.cpp", ChartYear);             // звіт з охоплення лікування за віковими категоріями у формі діаграми
USEFORM("Chart_Diag.cpp", ChartDiag);             // звіт з охоплення лікування за діагнозами нозологій у формі діаграми
USEFORM("Form_Message_Dlg.cpp", Message_Dialog);    // вікно показу повідомлення
USEFORM("Form_Show_Message.cpp", Show_Message);  // діалогове вікно вибору подальшої дії
//---------------------------------------------------------
int WINAPI _tWinMain(HINSTANCE, HINSTANCE, LPTSTR, int) {
        try {
                Application->Initialize();
                Application->MainFormOnTaskBar = true;
                Application->CreateForm(__classid(TDM), &DM);
                Application->CreateForm(__classid(TMainForm), &MainForm);
                Application->Run();
        }
        catch (Exception &exception)
        {Application->ShowException(&exception);}
        catch (...) {
                try
                {throw Exception("");}
                catch (Exception &exception)
                {Application->ShowException(&exception);}
        }
        return 0;
}
```

Файл Main.cpp

```
//-------- головне вікно проекту ----------
#pragma hdrstop
#include "Main.h"
//---------------------------------------------------------
#pragma package(smart_init)
#pragma link "ccalendr"
#pragma link "Vcl.HTMLHelpViewer"
#pragma resource "*.dfm"
TMainForm *MainForm;
TDateTime CurentDate;
AnsiString period;
bool UpDate;
int indexColumn;
//---------------------------------------------------------
__fastcall TMainForm::TMainForm(TComponent* Owner) : TForm(Owner) {}
//--------- метод обробки події створення форми ----------
void __fastcall TMainForm::FormCreate(TObject *Sender) {
        TStringList *Conect_string= new TStringList;
        DM->ADOConnection->ConnectionString="";
        DM->ADOConnection->Connected=false;
        if(FileExists("BaseConected.dat")) {
                Conect_string->LoadFromFile("BaseConected.dat");
                AnsiString s0=WideString(Conect_string[0].Text);
                s0.Delete(s0.Pos("\n"),s0.Length());
                DM->ADOConnection->ConnectionString=WideString(s0);
                DM->ADOConnection->Connected=true;
        }
        CurentDate=Now();
        MonthCalendar->Date=CurentDate;
        mmZvitMonth->Enabled=false;
        mmZvitQuarter->Enabled=false;
        mmZvitHalfYear->Enabled=false;
        mmZvit9Month->Enabled=false;
        mmZvitYear->Enabled=false;
        UpDate=false;
}
//---------- метод обробки події зміни розміру форми ----------
void __fastcall TMainForm::FormResize(TObject *Sender) {
        this->Width=645;
```

```
        this->Height=390;
}
//---------- метод обробки події показу форми ----------
void __fastcall TMainForm::FormShow(TObject *Sender) {
        if (!DM->ADOConnection->Connected) { MyShowMessage("Ініціалізація бази даних", "Помилка підключення!","Вказана база даних недоступна.\nПеревірте параметри у файлі конфігурації\n'BaseConected.dat' і перезапустіть проект!");
                Close();
        }
        mBackUp->Checked=DM->Checked_BackUp(false,false);
}
//------- метод обробки пункту меню "Створення резервної копії при виході" --------
void __fastcall TMainForm::BackUpClick(TObject *Sender) {
        mBackUp->Checked=!mBackUp->Checked;
        DM->Checked_BackUp(true,mBackUp->Checked);
}
//--------- метод встановлення доступності елементів головного меню ----------
void __fastcall TMainForm::Enab(bool En) {
        mVed->Enabled=En;
        mAbout->Enabled=En;
        mServis->Enabled=En;
}
//--------- метод обробки пункту меню "Редагування відомостей" ----------
void __fastcall TMainForm::EditInfoClick(TObject *Sender) {
        Application->CreateForm(__classid(TFormEditInfo), &FormEditInfo);
        FormEditInfo->ShowModal();
}
//--------- метод обробки пункту меню "Адміністративний поділ України"----------
void __fastcall TMainForm::ServisHelpsAdminClick(TObject *Sender) {
        Application->CreateForm(__classid(TFormHelpsAdmin), &FormHelpsAdmin);
        FormHelpsAdmin->ShowModal();
}
//---------- метод обробки пункту меню "Довідник нозологій" ----------
void __fastcall TMainForm::ServisHelpsShifrClick(TObject *Sender) {
        Application->CreateForm(__classid(TFormHelpsDiagnoz), &FormHelpsDiagnoz);
        FormHelpsDiagnoz->ShowModal();
}
//---------- метод обробки пункту меню 'Формування звітності' ----------
void __fastcall TMainForm::ZvitCreateClick(TObject *Sender) {
        MonthCalendar->Tag=1;
        ServisCheckDateClick(Sender);
}
//------ метод обробки пункту меню "Охоплення лікуванням за віковими категоріями" -------
void __fastcall TMainForm::Zvit_YearClick(TObject *Sender) {
        int tag=((TControl *)Sender)->Tag;
        AnsiString tabName;
        switch  (tag) {
                case 1  :  {tabName="Zv_IncidenceMonth"; break;}
                case 11 : {tabName="Zv_IncidenceQuarter"; break;}
                case 21 : {tabName="Zv_IncidenceHalfYear"; break;}
                case 31 : {tabName="Zv_IncidenceYear";          break;}
                case 41 : {tabName="Zv_IncidenceYear";          break;}
        }
        DM->Query_Year->SQL->Clear();
        DM->Query_Year->SQL->Add("Select * from "+tabName);
        DM->Query_sumYear->SQL->Clear();
        DM->Query_sumYear->SQL->Add("Select SUM(Boys_adm) AS Boys_adm,SUM(Boys_disc) AS Boys_disc,SUM(Boys) AS Boys,SUM(Girls_adm) AS Girls_adm,SUM(Girls_disc) AS Girls_disc,SUM(Girls) AS Girls,SUM(Together_adm) AS Together_adm,SUM(Together_disc) AS Together_disc,SUM(Together) AS Together from "+tabName);
        Application->CreateForm(__classid(TFormZvitYear), &FormZvitYear);
        FormZvitYear->Tag=tag;
        FormZvitYear->ShowModal();
}
//------ метод обробки пункту меню "Охоплення лікуванням за шифрами нозологій" --------
void __fastcall TMainForm::Zvit_ShifrClick(TObject *Sender) {
        int tag=((TControl *)Sender)->Tag;
        AnsiString tabName;
        switch  (tag) {
                case 2  :  {tabName="Zv_DiagnosisMonth"; break;}
                case 12 : {tabName="Zv_DiagnosisQuarter"; break;}
                case 22 : {tabName="Zv_DiagnosisHalfYear"; break;}
                case 32 : {tabName="Zv_Diagnosis9Month"; break;}
                case 42 : {tabName="Zv_DiagnosisYear";          break;}
        }
        DM->Query_Shifr->SQL->Clear();
        DM->Query_Shifr->SQL->Add("Select a.*, b.* from "+tabName+" a, Diagnosis b Where a.Id_diagnosis=b.Id");
        DM->Query_sumShifr->SQL->Clear();
        DM->Query_sumShifr->SQL->Add("Select SUM(Year_1) AS Year_1,SUM(Year_2) AS Year_2,SUM(Year_3) AS Year_3,SUM(Year_4) AS Year_4,SUM(Year_5) AS Year_5,SUM(Year_6) AS Year_6,SUM(Year_7) AS Year_7,SUM(Year_8) AS Year_8,SUM(Year_9) AS Year_9,SUM(Year_10) AS Year_10,SUM(Year_11) AS Year_11,SUM(Year_12) AS Year_12,SUM(Year_13) AS Year_13,SUM(Year_14) AS Year_14,SUM(Year_15) AS Year_15,SUM(Year_16) AS Year_16,SUM(Together) AS  Together from "+tabName);
        Application->CreateForm(__classid(TFormZvitShifr), &FormZvitShifr);
        FormZvitShifr->Tag=tag;
```

```
		FormZvitShifr->ShowModal();
}
//------ метод обробки пункту меню "Охоплення лікуванням за регіонами" ----------
void __fastcall TMainForm::Zvit_RegionClick(TObject *Sender) {
		int tag=((TControl *)Sender)->Tag;
		AnsiString tabName;
		switch (tag) {
			case 3 :  {tabName="Zv_RegionMonth";		break;}
			case 13 : {tabName="Zv_RegionQuarter"; break;}
			case 23 : {tabName="Zv_RegionHalfYear";break;}
			case 33 : {tabName="Zv_Region9Month";		break;}
			case 43 : {tabName="Zv_RegionYear";  break;}
		}
		DM->Query_Region->SQL->Clear();
		DM->Query_sumRegion->SQL->Clear();
		DM->Query_Region->SQL->Add("Select a.*, b.* from "+tabName+" a, Region b Where a.Id_region=b.Id and Together>0");
		DM->Query_sumRegion->SQL->Add("Select SUM(Year_1) AS Year_1,SUM(Year_2) AS Year_2,SUM(Year_3) AS
Year_3,SUM(Year_4) AS Year_4,SUM(Year_5) AS Year_5,SUM(Year_6) AS Year_6,SUM(Year_7) AS Year_7,SUM(Year_8) AS
Year_8,SUM(Year_9) AS Year_9,SUM(Year_10) AS Year_10,SUM(Year_11) AS Year_11,SUM(Year_12) AS Year_12,SUM(Year_13) AS
Year_13,SUM(Year_14) AS Year_14,SUM(Year_15) AS Year_15,SUM(Year_16) AS Year_16,SUM(Together) AS  Together from
"+tabName);
		Application->CreateForm(__classid(TFormZvitRegion), &FormZvitRegion);
		FormZvitRegion->Tag=tag;
		FormZvitRegion->ShowModal();
}
//-------- метод обробки пункту меню "Зміна дати" ----------
void __fastcall TMainForm::ServisCheckDateClick(TObject *Sender) {
		Enab(false);
		PanelImage1->Visible=false;
		PanelImage2->Visible=false;
		PanelDate->Top=35;
		PanelDate->Left=185;
		PanelDate->Visible=true;
		ActiveControl=MonthCalendar;
}
//-------- метод обробки кліку на кнопці "Ок" (панель зміни дати) ----------
void __fastcall TMainForm::ButtonDateOkClick(TObject *Sender) {
		CurentDate=MonthCalendar->Date;
		PanelDate->Visible=false;
		if (MonthCalendar->Tag) {
			Application->ProcessMessages();
			PanelWait->Visible=true;
			Application->ProcessMessages();
			Word Year, Month, Day;
			DecodeDate(CurentDate, Year, Month, Day);
			int wot_zvit = (!(Month%3)) ? (Month/3) : 0;
			DM->ADOProc_CreateZvitAll->Parameters->Refresh();
			DM->ADOProc_CreateZvitAll->Parameters->ParamByName("@date_zvit")->Value=CurentDate;
			DM->ADOProc_CreateZvitAll->Parameters->ParamByName("@wot_zvit")->Value=wot_zvit;
			DM->ADOProc_CreateZvitAll->Prepared=true;
			DM->ADOProc_CreateZvitAll->ExecProc();
			mmZvitMonth->Enabled=true;
			mmZvitQuarter->Enabled=(bool)wot_zvit;
			mmZvitHalfYear->Enabled=wot_zvit==2;
			mmZvit9Month->Enabled=wot_zvit==3;
			mmZvitYear->Enabled=wot_zvit==4;
			PanelWait->Visible=false;
		}
		PanelImage1->Visible=true;
		PanelImage2->Visible=true;
		Enab(true);
		MonthCalendar->Tag=0;
}
//---------- метод обробки кліку на кнопці "Відмова" (панель зміни дати) -------
void __fastcall TMainForm::ButtonDateCancelClick(TObject *Sender) {
		PanelImage1->Visible=true;
		PanelImage2->Visible=true;
		PanelDate->Visible=false;
		Enab(true);
		MonthCalendar->Tag=0;
}
//---------- метод обробки пункту меню "Про автора" ----------
void __fastcall TMainForm::AboutClick(TObject *Sender) {
		Application->CreateForm(__classid(TFormAbout), &FormAbout);
		FormAbout->ShowModal();
}
//---------- метод обробки події перед закриттям форми ----------
void __fastcall TMainForm::FormCloseQuery(TObject *Sender, bool &CanClose) {
		#ifdef NDEBUG
		if (UpDate && mBackUp->Checked) {
			Label1->Caption="Не хвилюйтесь, створюється резервна копія!";
			Application->ProcessMessages();
			PanelWait->Visible=true;
```

```
                Application->ProcessMessages();
                DM->SP_BackUp->Parameters->Refresh();
                DM->SP_BackUp->Prepared=true;
                DM->SP_BackUp->ExecProc();
                PanelWait->Visible=false;
        }
        #endif
}
//-------- метод обробки пункту меню "Вихід" ------------
void __fastcall TMainForm::ExitClick(TObject *Sender) {
        Close();
}
```

Файл Append_Patient.cpp

```
//------- форма добавлення та зміни інформації про пацієнтів відділення --------
#pragma hdrstop
#include "Append_Patient.h"
//-------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm_Append *Form_Append;
struct TListAdminPodil {
        AnsiString Text;                 // покажчик на інформаційну частину
        int Id;
        TListAdminPodil *Region;        // покажчик на список елементів наступного рівня
        TListAdminPodil *Rayon;         // покажчик на наступний елемент списку цього рівня
        TListAdminPodil() {             // конструктор за замовчуванням
                Text = "";
                Id = 0;
                Region = NULL;
                Rayon = NULL;
        }
};
TListAdminPodil *RootAdminPodil=NULL;
extern TDateTime CurentDate;
extern bool UpDate;
//-------------------------------------------------
__fastcall TForm_Append::TForm_Append(TComponent* Owner) : TForm(Owner) {  }
//------ метод обробки події створення форми ---------
void __fastcall TForm_Append::FormCreate(TObject *Sender) {
        DM->Query->SQL->Clear();
        DM->Query->SQL->Add("Select * from Sex");
        DM->Query->Active=true;
        ComboBox_Sex->Items->Clear();
        ComboBox_Sex_ID->Items->Clear();
        while (!DM->Query->Eof) {
                ComboBox_Sex->Items->Add(DM->Query->FieldByName("SexName")->AsString);
                ComboBox_Sex_ID->Items->Add(DM->Query->FieldByName("ID")->AsString);
                DM->Query->Next();
        }
        DM->Query->Active=false;
        DM->Query->SQL->Clear();
        DM->Query->SQL->Add("Select * from Diagnosis order by Shifr");
        DM->Query->Active=true;
        ComboBox_Shifr1->Items->Clear();
        ComboBox_Diagnoz1->Items->Clear();
        ComboBox_Shifr2->Items->Clear();
        ComboBox_Diagnoz2->Items->Clear();
        ComboBox_Shifr3->Items->Clear();
        ComboBox_Diagnoz3->Items->Clear();
        ComboBox_Shifr4->Items->Clear();
        ComboBox_Diagnoz4->Items->Clear();
        ComboBox_Diagnoz_ID->Items->Clear();
        ComboBox_therapy->Items->Clear();
        while (!DM->Query->Eof) {
                ComboBox_Shifr1->Items->Add(DM->Query->FieldByName("Shifr")->AsString);
                ComboBox_Diagnoz1->Items->Add(DM->Query->FieldByName("DiagnozName")->AsString);
                ComboBox_Shifr2->Items->Add(DM->Query->FieldByName("Shifr")->AsString);
                ComboBox_Diagnoz2->Items->Add(DM->Query->FieldByName("DiagnozName")->AsString);
                ComboBox_Shifr3->Items->Add(DM->Query->FieldByName("Shifr")->AsString);
                ComboBox_Diagnoz3->Items->Add(DM->Query->FieldByName("DiagnozName")->AsString);
                ComboBox_Shifr4->Items->Add(DM->Query->FieldByName("Shifr")->AsString);
                ComboBox_Diagnoz4->Items->Add(DM->Query->FieldByName("DiagnozName")->AsString);
                ComboBox_Diagnoz_ID->Items->Add(DM->Query->FieldByName("ID")->AsString);
                ComboBox_therapy->Items->Add(DM->Query->FieldByName("therapy")->AsString);
                DM->Query->Next();
        }
        DM->Query->Active=false;
        DM->Query->Active=false;
        DM->Query->SQL->Clear();
        DM->Query->SQL->Add("Select * from Region");
        DM->Query->Active=true;
```

```
        while (!DM->Query->Eof) {
        TTreeNode *Root=TreeUA->Items->AddChild(NULL,DM->Query->FieldByName("RegionName")-> AsString);
        AnsiString _sql="Select * from RayonCity where id_region="+DM->Query->FieldByName("ID")-> AsString;
                DM->Query2->Active=false;
                DM->Query2->SQL->Clear();
                DM->Query2->SQL->Add(_sql);
                DM->Query2->Active=true;
                TListAdminPodil *newRegion=new TListAdminPodil;
                newRegion->Text=DM->Query->FieldByName("RegionName")->AsString;
                newRegion->Id=DM->Query->FieldByName("ID")->AsInteger;
                if (!RootAdminPodil)
                  RootAdminPodil=newRegion;
                else {
                  TListAdminPodil *tmp=RootAdminPodil;
                  while (tmp->Region) tmp=tmp->Region;
                  tmp->Region=newRegion;
                }
                DM->Query2->First();
                while (!DM->Query2->Eof) {
                        TreeUA->Items->AddChild(Root,DM->Query2->FieldByName("RayonName")->AsString);
                        TListAdminPodil *newRayon=new TListAdminPodil;
                        newRayon->Text=DM->Query2->FieldByName("RayonName")->AsString;
                        newRayon->Id=DM->Query2->FieldByName("ID")->AsInteger;
                        newRegion->Rayon=newRayon;
                        newRegion=newRayon;
                        DM->Query2->Next();
                  }
                DM->Query->Next();
        }
        DM->Query->Active=false;
        DM->Query2->Active=false;
        BitBtn_Append->Left=350;
        BitBtn_Update->Left=350;
}
//-------- метод обробки події показу форми ------------
void __fastcall TForm_Append::FormShow(TObject *Sender) {
        if (this->Tag) Full_PanelAdd();
                else Clear_PanelAdd();
        ActiveControl=Edit_FIO;
}
//-------- метод обробки події зміни розміру форми ---------
void __fastcall TForm_Append::FormResize(TObject *Sender) {
        this->Width=1260;
        this->Height=800;
        BitBtn_Abort->Left=Panel_NewButton->Width-500;
}
//------ визначення ID регіону та району проживання за їх назвами ---------
void __fastcall SerchAdminPodil(AnsiString nameRegion, AnsiString nameRayon, int &Id_Region, int &Id_Rayon) {
        TListAdminPodil *tmpRegion=RootAdminPodil;
        while (tmpRegion->Text!=nameRegion) tmpRegion=tmpRegion->Region;
        Id_Region=tmpRegion->Id;
        while (tmpRegion->Text!=nameRayon) tmpRegion=tmpRegion->Rayon;
        Id_Rayon=tmpRegion->Id;
}
//-------- визначення назв регіону та району проживання за їх ID ----------
void __fastcall SerchAdminPodil(int Id_Region, int Id_Rayon, AnsiString &nameRegion, AnsiString &nameRayon) {
        TListAdminPodil *tmpRegion=RootAdminPodil;
        while (tmpRegion->Id!=Id_Region) tmpRegion=tmpRegion->Region;
        nameRegion=tmpRegion->Text;
        while (tmpRegion->Id!=Id_Rayon && tmpRegion->Rayon) tmpRegion=tmpRegion->Rayon;
        if (tmpRegion->Id==Id_Rayon) nameRayon=tmpRegion->Text;
                else {nameRegion=""; nameRayon=" ";}
}
//-------- метод встановлення ключа онлайн-довідки в залежності від режиму використання форми ---------
void __fastcall TForm_Append::Full_HelpContext(UnicodeString hlp) {
        HelpKeyword=hlp;
        Panel_NewRigth->HelpKeyword=hlp;
        Edit_FIO->HelpKeyword=hlp;
        Edit_Birth->HelpKeyword=hlp;
        DateTimePicker_newBirth->HelpKeyword=hlp;
        ComboBox_Sex->HelpKeyword=hlp;
        Edit_Region->HelpKeyword=hlp;
        Edit_Rayon->HelpKeyword=hlp;
        Edit_Adress->HelpKeyword=hlp;
        Edit_Add->HelpKeyword=hlp;
        DateTimePicker_newAdd->HelpKeyword=hlp;
        Edit_Dec->HelpKeyword=hlp;
        DateTimePicker_newDec->HelpKeyword=hlp;
        ComboBox_Shifr1->HelpKeyword=hlp;
        ComboBox_Diagnoz1->HelpKeyword=hlp;
        ComboBox_Shifr2->HelpKeyword=hlp;
        ComboBox_Diagnoz2->HelpKeyword=hlp;
        ComboBox_Shifr3->HelpKeyword=hlp;
```

```
        ComboBox_Diagnoz3->HelpKeyword=hlp;
        ComboBox_Shifr4->HelpKeyword=hlp;
        ComboBox_Diagnoz4->HelpKeyword=hlp;
        Memo_Now_therapy->HelpKeyword=hlp;
        TreeUA->HelpKeyword=hlp;
        Panel_NewButton->HelpKeyword=hlp;
}
//------ метод заповнення візуальних компонентів форми даними поточного пацієнта -------
void __fastcall TForm_Append::Full_PanelAdd() {
        Full_HelpContext(TEXT("Зміна інформації про пацієнта"));
        DM->Query->Active=false;
        DM->Query->SQL->Clear();
        DM->Query->SQL->Add("Select * from Child where Id="+IntToStr(this->Tag));
        DM->Query->Active=true;
        Edit_FIO->Text=DM->Query->FieldByName("FullName")->AsString;
        Edit_Birth->Text=DM->Query->FieldByName("date_birth")->AsString;
ComboBox_Sex->ItemIndex=Id_in_ComboBox(ComboBox_Sex_ID, DM->Query->FieldByName("id_sex")-> AsString);
        AnsiString nameRegion, nameRayon;
SerchAdminPodil(DM->Query->FieldByName("id_region")->AsInteger,DM->Query-> FieldByName("id_rayon_city")-> AsInteger,
nameRegion, nameRayon);
        Edit_Region->Text=nameRegion;
        Edit_Rayon->Text=nameRayon;
        Edit_Adress->Text=DM->Query->FieldByName("DetailAdress")->AsString;
        Edit_Add->Text=DM->Query->FieldByName("date_admission")->AsString;
        Edit_Dec->Text=DM->Query->FieldByName("date_discharge")->AsString;
        ComboBox_Shifr1->ItemIndex=Id_in_ComboBox(ComboBox_Diagnoz_ID, DM->Query-> FieldByName("id_diagnosis1")-
>AsString);
        ComboBox_Diagnoz1->ItemIndex=ComboBox_Shifr1->ItemIndex;
        ComboBox_Shifr2->ItemIndex=Id_in_ComboBox(ComboBox_Diagnoz_ID, DM->Query-> FieldByName("id_diagnosis2")-
>AsString);;
        ComboBox_Diagnoz2->ItemIndex=ComboBox_Shifr2->ItemIndex;
        ComboBox_Shifr3->ItemIndex=Id_in_ComboBox(ComboBox_Diagnoz_ID, DM->Query-> FieldByName("id_diagnosis3")-
>AsString);;
        ComboBox_Diagnoz3->ItemIndex=ComboBox_Shifr3->ItemIndex;
        ComboBox_Shifr4->ItemIndex=Id_in_ComboBox(ComboBox_Diagnoz_ID, DM->Query-> FieldByName("id_diagnosis4")-
>AsString);;
        ComboBox_Diagnoz4->ItemIndex=ComboBox_Shifr4->ItemIndex;
        Memo_Now_therapy->Lines->Clear();
        Memo_Now_therapy->Lines->Add(DM->Query->FieldByName("Now_therapy")->AsString);
        DateTimePicker_newBirth->Visible=false;
        DateTimePicker_newAdd->Visible=false;
        DateTimePicker_newDec->Visible=false;
        CheckBox1->Checked=false;
        CheckBox2->Checked=false;
        CheckBox3->Checked=false;
        CheckBox4->Checked=false;
        TreeUA->FullCollapse();
        TreeUA->Select(TreeUA->Items->GetFirstNode());
        BitBtn_Append->Visible=false;
        BitBtn_Update->Visible=true;
        BitBtn_Abort->Caption="Скасувати зміни";
        Label_Title->Caption="Уведіть інформацію про даного пацієнта";
}
//-------- метод очистки візуальних компонентів форми для уведення нових даних ---------
void __fastcall TForm_Append::Clear_PanelAdd() {
        Full_HelpContext(TEXT("Добавлення нового пацієнта"));
        Edit_FIO->Text="";
        Edit_Birth->Text="";
        ComboBox_Sex->ItemIndex=-1;
        Edit_Region->Text="";
        Edit_Rayon->Text="";
        Edit_Adress->Text="";
        Edit_Add->Text="";
        Edit_Dec->Text="";
        ComboBox_Shifr1->ItemIndex=-1;
        ComboBox_Diagnoz1->ItemIndex=-1;
        ComboBox_Shifr2->ItemIndex=-1;
        ComboBox_Diagnoz2->ItemIndex=-1;
        ComboBox_Shifr3->ItemIndex=-1;
        ComboBox_Diagnoz3->ItemIndex=-1;
        ComboBox_Shifr4->ItemIndex=-1;
        ComboBox_Diagnoz4->ItemIndex=-1;
        Memo_Now_therapy->Lines->Clear();
        CheckBox1->Checked=true;
        CheckBox2->Checked=true;
        CheckBox3->Checked=true;
        CheckBox4->Checked=true;
        DateTimePicker_newBirth->Visible=false;
        DateTimePicker_newAdd->Visible=false;
        DateTimePicker_newDec->Visible=false;
        TreeUA->FullCollapse();
        TreeUA->Select(TreeUA->Items->GetFirstNode());
        BitBtn_Append->Visible=true;
```

```
        BitBtn_Update->Visible=false;
        BitBtn_Abort->Caption="Скасувати додавання";
        Label_Title->Caption="Уведіть інформацію про нового пацієнта";
}
//-------- метод обробки події вибору прапорця ----------
void __fastcall TForm_Append::CheckBoxClick(TObject *Sender) {
        int tag=((TControl *)Sender)->Tag;
        AnsiString text=Memo_Now_therapy->Text.Trim();
        if (Memo_Now_therapy->Text.Trim().Length()==0) Memo_Now_therapy->Lines->Clear();
        switch  (tag) {
                case 1 : { if (CheckBox1->Checked) {    ComboBox_therapy->ItemIndex=ComboBox_Shifr1->ItemIndex;
                                                        Memo_Now_therapy->Lines->Add(ComboBox_therapy->Text); }
                           break;}
                case 2 : { if (CheckBox2->Checked) {    ComboBox_therapy->ItemIndex=ComboBox_Shifr2->ItemIndex;
                                                        Memo_Now_therapy->Lines->Add(ComboBox_therapy->Text); }
                           break;}
                case 3 : { if (CheckBox3->Checked) {    ComboBox_therapy->ItemIndex=ComboBox_Shifr3->ItemIndex;
                                                        Memo_Now_therapy->Lines->Add(ComboBox_therapy->Text); }
                           break;}
                case 4 : { if (CheckBox4->Checked) {    ComboBox_therapy->ItemIndex=ComboBox_Shifr4->ItemIndex;
                                                        Memo_Now_therapy->Lines->Add(ComboBox_therapy->Text); }
                           break;}
                }
}
//-------- метод перевірки уведеної інформації на повноту та коректність ----------
bool __fastcall TForm_Append::Checking_Info(int &err) {
        bool rezult=Edit_FIO->Text.Length()>0;
        err=1;
        if (rezult) { rezult=Edit_Birth->Text.Length()>0; err=2; }
        if (rezult) { rezult=(ComboBox_Sex->ItemIndex>-1); err=3;    }
        if (rezult) { rezult=Edit_Region->Text.Length()>0; err=4; }
        if (rezult) { rezult=Edit_Rayon->Text.Length()>0; err=4; }
        if (rezult) { rezult=Edit_Adress->Text.Length()>0; err=6; }
        if (rezult) { rezult=Edit_Add->Text.Length()>0; err=7; }
        if (rezult) { rezult=(ComboBox_Shifr1->ItemIndex>-1); err=8; }
        return  rezult;
}
//------ метод заповнення параметрів збереженої SQL-процедури ---------
void __fastcall TForm_Append::Fill_Proc_Parameters(TADOStoredProc *ADOProc) {
        ComboBox_Sex_ID->ItemIndex = ComboBox_Sex->ItemIndex;
        int Id_Region=0, Id_Rayon=0;
        SerchAdminPodil(Edit_Region->Text,Edit_Rayon->Text,Id_Region,Id_Rayon);
        ADOProc->Parameters->Refresh();
        ADOProc->Parameters->ParamByName("@_FullName")->Value=Edit_FIO->Text;
        ADOProc->Parameters->ParamByName("@_date_birth")->Value=StrToDateTime(Edit_Birth->Text);
        ADOProc->Parameters->ParamByName("@_id_sex")->Value=StrToInt(ComboBox_Sex_ID->Text);
        ADOProc->Parameters->ParamByName("@_id_region")->Value=Id_Region;
        ADOProc->Parameters->ParamByName("@_id_rayon_city")->Value=Id_Rayon;
        ADOProc->Parameters->ParamByName("@_DetailAdress")->Value=Edit_Adress->Text;
        ADOProc->Parameters->ParamByName("@_date_admission")->Value=StrToDateTime(Edit_Add->Text);
        ADOProc->Parameters->ParamByName("@_Now_therapy")->Value=Memo_Now_therapy->Text.Trim();
        if (Edit_Dec->Text.Length()>0)              ADOProc->Parameters->ParamByName("@_date_discharge")-
>Value=StrToDateTime(Edit_Dec->Text);
        ComboBox_Diagnoz_ID->ItemIndex = ComboBox_Shifr1->ItemIndex;
        ADOProc->Parameters->ParamByName("@_id_diagnosis1")->Value=StrToInt(ComboBox_Diagnoz_ID->Text);
        if (ComboBox_Shifr2->ItemIndex>-1) { ComboBox_Diagnoz_ID->ItemIndex = ComboBox_Shifr2->ItemIndex;
        ADOProc->Parameters->ParamByName("@_id_diagnosis2")->Value=StrToInt(ComboBox_Diagnoz_ID->Text);
        }
        if (ComboBox_Shifr3->ItemIndex>-1) { ComboBox_Diagnoz_ID->ItemIndex = ComboBox_Shifr3->ItemIndex;
        ADOProc->Parameters->ParamByName("@_id_diagnosis3")->Value=StrToInt(ComboBox_Diagnoz_ID->Text);
        }
        if (ComboBox_Shifr4->ItemIndex>-1) { ComboBox_Diagnoz_ID->ItemIndex = ComboBox_Shifr4->ItemIndex;
        ADOProc->Parameters->ParamByName("@_id_diagnosis4")->Value=StrToInt(ComboBox_Diagnoz_ID->Text);
        }
}
//------ метод обробки кліку на кнопці " Додати інформацію в базу / Внести зміни  в базу" ------
void __fastcall TForm_Append::BitBtn_AppendClick(TObject *Sender) {
  int err=0;
  if (Checking_Info(err)) {
        int Id=0;
        if (this->Tag) {
                Fill_Proc_Parameters(DM->ADOProc_Update_Patient);
                DM->ADOProc_Update_Patient->Parameters->ParamByName("@id_rec")->Value=this->Tag;
                DM->ADOProc_Update_Patient->Prepared=true;
                DM->ADOProc_Update_Patient->ExecProc();
                Id=this->Tag;
        } else {    Fill_Proc_Parameters(DM->ADOProc_Append_Patient);
                DM->ADOProc_Append_Patient->Parameters->ParamByName("@Id")->Value=0;
                DM->ADOProc_Append_Patient->Prepared=true;
                DM->ADOProc_Append_Patient->ExecProc();
                Id=DM->ADOProc_Append_Patient->Parameters->ParamByName("@Id")->Value;
        }
        FormEditInfo->Query_Child->Tag=Id;
```

```
            DM->TableAfterInsertDelete(FormEditInfo->Query_Child,"ID_ch");
            MainForm->mmZvitMonth->Enabled=false;
            MainForm->mmZvitQuarter->Enabled=false;
            MainForm->mmZvitHalfYear->Enabled=false;
            MainForm->mmZvit9Month->Enabled=false;
            MainForm->mmZvitYear->Enabled=false;
            UpDate=true;
            this->Tag=0;
            Close();
 }
  else {    if (this->Tag) MyShowMessage("Редагування даних", "Помилка редагування даних!","Неповне заповнення даних про
поточного пацієнта.\nПродовжіть заповнення.");
            else MyShowMessage("Додавання нового пацієнта", "Помилка додавання даних!","Неповне заповнення даних про нового
пацієнта.\nПродовжіть заповнення.");
            switch  (err) {
              case 1 : {ActiveControl=Edit_FIO; break;}
              case 2 : {ActiveControl=Edit_Birth; break;}
              case 3 : {ActiveControl=ComboBox_Sex; break;}
              case 4 : {ActiveControl=TreeUA; break;}
              case 6 : {ActiveControl=Edit_Adress; break;}
              case 7 : {ActiveControl=Edit_Add; break;}
              case 8 : {ActiveControl=ComboBox_Shifr1; break;}
            }
            TreeUA->FullCollapse();
            TreeUA->Select(TreeUA->Items->GetFirstNode());
 }
}
//------- метод обробки кліку на полях місця проживання пацієнта ----------
void __fastcall TForm_Append::Edit_RayonClick(TObject *Sender) {
            ActiveControl=TreeUA;
}
//------ метод обробки подвійного кліку на елементі дерева адміністративного поділу України ----------
void __fastcall TForm_Append::TreeUADblClick(TObject *Sender) {
            if (TreeUA->Selected->Level) {
                        Edit_Region->Text=TreeUA->Selected->Parent->Text;
                        Edit_Rayon->Text=TreeUA->Selected->Text;
                        ActiveControl=Edit_Adress;
            }
}
//------ метод обробки отримання фокусу полями з датою ---------
void __fastcall TForm_Append::Edit_newDateChange(TObject *Sender) {
            switch (((TControl *)Sender)->Tag) {
                        case 1 : { if (Edit_Birth->Text.Length()>0) DateTimePicker_newBirth->Date=Edit_Birth->Text;
                                                else DateTimePicker_newBirth->Date=CurentDate;
                                                DateTimePicker_newBirth->Top=((TControl *)Sender)->Top;
                                                DateTimePicker_newBirth->Left=((TControl *)Sender)->Left;
                                                DateTimePicker_newBirth->Visible=true;
                                                ActiveControl=DateTimePicker_newBirth;; break;}
                        case 2 : {if (Edit_Add->Text.Length()>0) DateTimePicker_newAdd->Date=Edit_Add->Text;
                                                else DateTimePicker_newAdd->Date=CurentDate;
                                                DateTimePicker_newAdd->Top=((TControl *)Sender)->Top;
                                                DateTimePicker_newAdd->Left=((TControl *)Sender)->Left;
                                                DateTimePicker_newAdd->Visible=true;
                                                ActiveControl=DateTimePicker_newAdd;; break;}
                        case 3 : {if (Edit_Dec->Text.Length()>0) DateTimePicker_newDec->Date=Edit_Dec->Text;
                                                else DateTimePicker_newDec->Date=CurentDate;
                                                DateTimePicker_newDec->Top=((TControl *)Sender)->Top;
                                                DateTimePicker_newDec->Left=((TControl *)Sender)->Left;
                                                DateTimePicker_newDec->Visible=true;
                                                ActiveControl=DateTimePicker_newDec;; break;}
            }
}
//------ метод обробки натиску клавіш на полях уведення -----
void __fastcall TForm_Append::Edit_newKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) {
            if (Key==VK_RETURN) Perform(WM_NEXTDLGCTL,0,0);
}
//------ метод обробки втрати фокусу компонентами типу "календар" ----------
void __fastcall TForm_Append::DateTimePicker_newExit(TObject *Sender) {
            switch  (((TControl *)Sender)->Tag) {
              case 1 : {if (DateTimePicker_newBirth->Date.operator int()==0) Edit_Birth->Text="";
                                else Edit_Birth->Text=FormatDateTime("ddddd",DateTimePicker_newBirth->Date);
                        DateTimePicker_newBirth->Visible=false; break;}
              case 2 : {if (DateTimePicker_newAdd->Date.operator int()==0) Edit_Add->Text="";
                                else Edit_Add->Text=FormatDateTime("ddddd",DateTimePicker_newAdd->Date);
                        DateTimePicker_newAdd->Visible=false; break;}
              case 3 : {if (DateTimePicker_newDec->Date.operator int()==0) Edit_Dec->Text="";
                                else Edit_Dec->Text=FormatDateTime("ddddd",DateTimePicker_newDec->Date);
                        DateTimePicker_newDec->Visible=false; break;}
            }
}
//------ метод обробки натиску клавіш на компонентах типу "календар" --------
void __fastcall TForm_Append::DateTimePicker_newDateKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) {
            if (Key==VK_RETURN)  Perform(WM_NEXTDLGCTL,0,0);
```

```
        if (Key==VK_DELETE) {  switch  (((TControl *)Sender)->Tag) {
                                                        case 1 : {DateTimePicker_newBirth->Date=0; break;}
                                                        case 2 : {DateTimePicker_newAdd->Date=0; break;}
                                                        case 3 : {DateTimePicker_newDec->Date=0; break;}
                                                }
          Perform(WM_NEXTDLGCTL,0,0);
        }
}
//-------- метод обробки події вибору позиції в списку шифрів діагнозів ---------
void __fastcall TForm_Append::ComboBox_ShifrChange(TObject *Sender) {
        int tag=((TControl *)Sender)->Tag;
        switch  (tag) {
                case 1 : {ComboBox_Diagnoz1->ItemIndex=ComboBox_Shifr1->ItemIndex;
                          ComboBox_therapy->ItemIndex=ComboBox_Shifr1->ItemIndex; break;}
                case 2 : {ComboBox_Diagnoz2->ItemIndex=ComboBox_Shifr2->ItemIndex;
                          ComboBox_therapy->ItemIndex=ComboBox_Shifr2->ItemIndex; break;}
                case 3 : {ComboBox_Diagnoz3->ItemIndex=ComboBox_Shifr3->ItemIndex;
                          ComboBox_therapy->ItemIndex=ComboBox_Shifr3->ItemIndex; break;}
                case 4 : {ComboBox_Diagnoz4->ItemIndex=ComboBox_Shifr4->ItemIndex;
                          ComboBox_therapy->ItemIndex=ComboBox_Shifr4->ItemIndex; break;}
                }
        if (CheckBox1->Checked) {
                if (Memo_Now_therapy->Text.Trim().Length()==0) Memo_Now_therapy->Lines->Clear();
                Memo_Now_therapy->Lines->Add(ComboBox_therapy->Text);
        }
}
//------ метод обробки події вибору позиції в списку назв діагнозів -------
void __fastcall TForm_Append::ComboBox_DiagnozChange(TObject *Sender) {
        int tag=((TControl *)Sender)->Tag;
        switch  (tag) {
                case 1 : {ComboBox_Shifr1->ItemIndex=ComboBox_Diagnoz1->ItemIndex;
                          ComboBox_therapy->ItemIndex=ComboBox_Diagnoz1->ItemIndex; break;}
                case 2 : {ComboBox_Shifr2->ItemIndex=ComboBox_Diagnoz2->ItemIndex;
                          ComboBox_therapy->ItemIndex=ComboBox_Diagnoz1->ItemIndex; break;}
                case 3 : {ComboBox_Shifr3->ItemIndex=ComboBox_Diagnoz3->ItemIndex;
                          ComboBox_therapy->ItemIndex=ComboBox_Diagnoz1->ItemIndex; break;}
                case 4 : {ComboBox_Shifr4->ItemIndex=ComboBox_Diagnoz4->ItemIndex;
                          ComboBox_therapy->ItemIndex=ComboBox_Diagnoz1->ItemIndex; break;}
                }
        if (CheckBox1->Checked) {if (Memo_Now_therapy->Text.Trim().Length()==0) Memo_Now_therapy->Lines->Clear();
                Memo_Now_therapy->Lines->Add(ComboBox_therapy->Text);
   }
}
//--------- метод обробки натиску клавіші на компонентах "список, що розкривається" ----
void __fastcall TForm_Append::ComboBox_ShifrKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) {
        if (Key==VK_DELETE) {  switch  (((TControl *)Sender)->Tag) {
                                                        case 1 : {ComboBox_Shifr1->ItemIndex=-1; break;}
                                                        case 2 : {ComboBox_Shifr2->ItemIndex=-1; break;}
                                                        case 3 : {ComboBox_Shifr3->ItemIndex=-1; break;}
                                                        case 4 : {ComboBox_Shifr3->ItemIndex=-1; break;}
                                                }
          Perform(WM_NEXTDLGCTL,0,0); Perform(WM_NEXTDLGCTL,0,0); Perform(WM_NEXTDLGCTL,0,0);
        }
}
//--------- метод обробки кліку на кнопці "Скасувати додавання / Скасувати зміни" ---------
void __fastcall TForm_Append::BitBtn_AbortClick(TObject *Sender) {
        this->Close();
}
//-------- метод обробки події закриття форми ---------
void __fastcall TForm_Append::FormClose(TObject *Sender, TCloseAction &Action) {
        FormEditInfo->ActiveControl=FormEditInfo->DBGridFIO;
}
```

Файл FormEdit.cpp

```
//--------- форма перегляду та редагування інформації про пацієнтів відділення ----------
#pragma hdrstop
#include "FormEdit.h"
//---------------------------------------------------
#pragma package(smart_init)
#pragma link "frxClass"
#pragma link "frxDBSet"
#pragma resource "*.dfm"
TFormEditInfo *FormEditInfo;
AnsiString SQL_part1, TitleFilters;
extern TDateTime CurentDate;
TDateTime DateBefore;
//---------------------------------------------------
__fastcall TFormEditInfo::TFormEditInfo(TComponent* Owner) : TForm(Owner) { }
//--------- метод встановлення властивості SQL-запиту джерела даних ----------
void __fastcall TFormEditInfo::OpenChild_AllRecords(AnsiString _sql) {
        Query_Child->Active=false;
        Query_Child->SQL->Clear();
```

```
        Query_Child->SQL->Add(_sql+"order by FullName");
        Query_Child->Active=true;
}
//--------- метод обробки події створення форми --------
void __fastcall TFormEditInfo::FormCreate(TObject *Sender) {
        Diagnoz->Active=true;
        TitleFilters="";
        SQL_part1="Select *,a.ID As ID_ch from Child As a inner join Sex on Sex.ID = a.id_sex inner join RayonCity on RayonCity.ID =
a.id_rayon_city inner join Region on Region.ID=RayonCity.id_region inner join Diagnosis on Diagnosis.ID = a.id_diagnosis1 left join
Diagnosis As D2 on D2.ID = a.id_diagnosis2 left join Diagnosis As D3 on D3.ID = a.id_diagnosis3 left join Diagnosis As D4 on D4.ID =
a.id_diagnosis4 ";
        OpenChild_AllRecords(SQL_part1);
        Application->CreateForm(__classid(TForm_Append), &Form_Append);
        Application->CreateForm(__classid(TFormFilertSet), &FormFilertSet);
        DateBefore=CurentDate;
}
//--------- метод обробки події зміни розміру форми ----------
void __fastcall TFormEditInfo::FormResize(TObject *Sender) {
        this->Width=1260;
        this->Height=710;
        BitBtn_Exit->Left=PanelBottom->Width-150;
        TitleFilters="";
}
//-------- метод обробки натиску клавіш на формі ----------
void __fastcall TFormEditInfo::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) {
        if (Key==VK_INSERT && Shift.Contains(ssCtrl))    BitBtn_EditAppendClick(BitBtn_Append);
        if (Key==VK_DELETE && Shift.Contains(ssCtrl))    BitBtn_DeleteClick(Sender);
        if ((Key=='E' || Key=='e') && Shift.Contains(ssCtrl)) BitBtn_EditAppendClick(BitBtn_Edit);
        if ((Key=='F' || Key=='f') && Shift.Contains(ssCtrl)) ActiveControl=Edit_SerchFIO;
        if ((Key=='Q' || Key=='q') && Shift.Contains(ssCtrl)) BitBtn_fillterSetClick(BitBtn_fillterSet);
        if ((Key=='W' || Key=='w') && Shift.Contains(ssCtrl)) BitBtn_fillterSetClick(BitBtn_fillterReplace);
        if ((Key=='A' || Key=='a') && Shift.Contains(ssCtrl)) BitBtn_fillterClearClick(Sender);
        if ((Key=='P' || Key=='p') && Shift.Contains(ssCtrl)) BitBtn_PrintClick(Sender);
}
//--------- метод обробки кліку на кнопці "Додати пацієнта" ---------
void __fastcall TFormEditInfo::BitBtn_EditAppendClick(TObject *Sender) {
        Form_Append->Tag= !(((TControl *)Sender)->Tag) ? 0 : Query_Child->FieldByName("Id")->AsInteger;
        Form_Append->ShowModal();
}
//---------- метод обробки натиску клавіші на полях --------
void __fastcall TFormEditInfo::DBEditKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) {
        if (Key==VK_RETURN) {Perform(WM_NEXTDLGCTL,0,0);
                if ((((TControl *)Sender)->Tag>=20) && (((TControl *)Sender)->Tag<=25))
                Perform(WM_NEXTDLGCTL,0,0);
        }
}
//--------- метод обробки кліку на кнопці "Вилучити пацієнта" ----------
void __fastcall TFormEditInfo::BitBtn_DeleteClick(TObject *Sender) {
        if (MyMessageDlg("Список пацієнтів","Запит на вилучення","Ви впевнені, що хочете\nвилучити дані поточного пацієнта?",
mtWarning) == mrYes)  DM->DeleteRecord(Query_Child, "Id", "Child");
        ActiveControl=DBGridFIO;   }
//---------- метод обробки втрати фокусу поля уведення зразка пошуку ----------
void __fastcall TFormEditInfo::Edit_SerchFIOExit(TObject *Sender) {
         Edit_SerchFIO->Text="";
}
//-------- метод обробки уведення символів в поле пошуку за ПІП ----------
void __fastcall TFormEditInfo::Edit_SerchFIOKeyUp(TObject *Sender, WORD &Key, TShiftState Shift) {
        if (Key==VK_RETURN)         ActiveControl=DBGridFIO;
        else        if (Edit_SerchFIO->Text.Length()>0) {
                        Query_Child->DisableControls();
                        if (!Query_Child->Locate("FullName",Edit_SerchFIO->Text.Trim(),TLocateOptions()<<
loPartialKey<<loCaseInsensitive)) {
                                Query_Child->First();
                                bool flag=false;
                                while (!Query_Child->Eof && !flag) {
                                        AnsiString tit=AnsiUpperCase(Query_Child->FieldByName("FullName")->AsAnsiString);
                                        if (tit.Pos(AnsiUpperCase(Edit_SerchFIO->Text.Trim()))) flag=true;
                                                else Query_Child->Next();
                                }
                                if (!flag) Query_Child->First();
                               }
                               Query_Child->EnableControls();
        }
}
//------- метод обробки кліку на кнопках "Новий фільтр" та "Змінити фільтр" ----------
void __fastcall TFormEditInfo::BitBtn_fillterSetClick(TObject *Sender) {
        FormFilertSet->ShowModal();
        if (Query_Child->Eof)
          MyShowMessage("Список пацієнтів", "Результат фільтрації","Список пацієнтів\nза вказаними критиріями
пошуку\nпустий!");
}
//-------- метод обробки кліку на кнопці "Зняти фільтр" ---------
void __fastcall TFormEditInfo::BitBtn_fillterClearClick(TObject *Sender) {
```

```
        FormFilertSet->BitBtn_Clear_FilterClick(Sender);
        OpenChild_AllRecords(SQL_part1);
        TitleFilters="";
        ActiveControl=DBGridFIO;
}
//-------- метод обробки кліку на кнопці "Друк списку" ----------
void __fastcall TFormEditInfo::BitBtn_PrintClick(TObject *Sender) {
        TfrxMemoView *Memo_FilterTit = dynamic_cast <TfrxMemoView *> (frxReport_AfterFiltered->FindObject("Memo_FilterTit"));
        Memo_FilterTit->Memo->Clear();
        if (TitleFilters.Length()>0) Memo_FilterTit->Memo->Add("Результат фільтрації по: "+TitleFilters);
        else Memo_FilterTit->Memo->Add("Повний перелік");
        QueryReport_AftterFiltered->Active=false;
        QueryReport_AftterFiltered->SQL->Clear();
        QueryReport_AftterFiltered->SQL->Add(Query_Child->SQL->Text);
        QueryReport_AftterFiltered->Active=true;
        frxReport_AfterFiltered->ShowReport();
        QueryReport_AftterFiltered->Active=false;
}
//-------- метод обробки кліку на кнопці "Вихід" ----------
void __fastcall TFormEditInfo::ButtonExitClick(TObject *Sender) {
        Close();
}
//-------- метод  обробки події закриття форми ----------
void __fastcall TFormEditInfo::FormClose(TObject *Sender, TCloseAction &Action) {
        if (Query_Child->State==dsInsert || Query_Child->State==dsEdit)  Query_Child->Post();
        Diagnoz->Active=false;
        Query_Child->Active=false;
}
```

Файл FilterSet.cpp

```
//---------- форма фільтрації списку пацієнтів відділення ----------
#pragma hdrstop
#include "FilterSet.h"
//-------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TFormFilertSet *FormFilertSet;
extern TDateTime CurentDate;
extern AnsiString SQL_part1,TitleFilters;
//-------------------------------------------
__fastcall TFormFilertSet::TFormFilertSet(TComponent* Owner) : TForm(Owner) {  }
//-------- метод обробки події створення форми ---------
void __fastcall TFormFilertSet::FormCreate(TObject *Sender) {
        TADOTable *TableInfo= new TADOTable (this);
        TableInfo->Connection=DM->ADOConnection;
        TableInfo->Active=false;
        TableInfo->TableName="Sex";
        TableInfo->Active=true;
        C_Box_Sex->Items->Clear();
        C_Box_Sex_ID->Items->Clear();
        while (!TableInfo->Eof) { C_Box_Sex->Items->Add(TableInfo->FieldByName("SexName")->AsString);
                C_Box_Sex_ID->Items->Add(TableInfo->FieldByName("ID")->AsString);
                TableInfo->Next();
        }
        TableInfo->Active=false;
        TableInfo->TableName="Diagnosis";
        TableInfo->Active=true;
        C_Box_Shifr->Items->Clear();
        C_Box_Diagnoz->Items->Clear();
        while (!TableInfo->Eof) { C_Box_Shifr->Items->Add(TableInfo->FieldByName("Shifr")->AsString);
                C_Box_Diagnoz->Items->Add(TableInfo->FieldByName("DiagnozName")->AsString);
                C_Box_Diagnoz_ID->Items->Add(TableInfo->FieldByName("ID")->AsString);
                TableInfo->Next();
        }
        TableInfo->Active=false;;
        TableInfo->TableName="Region";
        TableInfo->Active=true;
        C_Box_Region->Items->Clear();
        C_Box_Region_ID->Items->Clear();
        while (!TableInfo->Eof) { C_Box_Region->Items->Add(TableInfo->FieldByName("RegionName")->AsString);
                C_Box_Region_ID->Items->Add(TableInfo->FieldByName("ID")->AsString);
                TableInfo->Next();
        }
        TableInfo->Active=false;
        Filter_Date_Clear();
}
//---------- метод обробки події зміни розміру форми ---------
void __fastcall TFormFilertSet::FormResize(TObject *Sender) {
        this->Width=995;
        this->Height=465;
}
//-------- метод обробки натиску клавіші на формі --------
void __fastcall TFormFilertSet::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift) {
```

```
        if ((Key=='Q' || Key=='q') && Shift.Contains(ssCtrl)) BitBtn_Apply_FilterClick(Sender);
        if ((Key=='A' || Key=='a') && Shift.Contains(ssCtrl)) BitBtn_Clear_FilterClick(Sender);
}
//-------- метод встановлення всім полям з датами значення поточної дати ----------
void __fastcall TFormFilertSet::Filter_Date_Clear() {
        DTFilterr_Birth_exactly->Date=CurentDate;
        DTFilterr_Birth_in->Date=CurentDate;
        DTFilterr_Birth_to->Date=CurentDate;
        DTFilterr_Add_exactly->Date=CurentDate;
        DTFilterr_Add_in->Date=CurentDate;
        DTFilterr_Add_to->Date=CurentDate;
        DTFilterr_Dec_exactly->Date=CurentDate;
        DTFilterr_Dec_in->Date=CurentDate;
        DTFilterr_Dec_to->Date=CurentDate;
}
//------ метод обробки кліку на кнопці " Очистити всі фільтри" ----------
void __fastcall TFormFilertSet::BitBtn_Clear_FilterClick(TObject *Sender) {
        TitleFilters="";
        Check_Filter_Sex->Checked=false;
        C_Box_Sex->ItemIndex=-1;
        Check_Filter_Region->Checked=false;
        C_Box_Region->ItemIndex=-1;
        Check_Filter_Birth_exactly->Checked=false;
        Check_Filter_Birth_in->Checked=false;
        Check_Filter_Birth_to->Checked=false;
        Check_Filter_Add_exactly->Checked=false;
        Check_Filter_Add_in->Checked=false;
        Check_Filter_Add_to->Checked=false;
        Check_Filter_Dec_exactly->Checked=false;
        Check_Filter_Dec_in->Checked=false;
        Check_Filter_Dec_to->Checked=false;
        Filter_Date_Clear();
        Check_Filter_Shifr->Checked=false;
        C_Box_Shifr->ItemIndex=-1;
        C_Box_Diagnoz->ItemIndex=-1;
        FormEditInfo->BitBtn_fillterReplace->Enabled=false;
        FormEditInfo->BitBtn_fillterClear->Enabled=false;
}
//-------- формування рядка Select-запиту ----------
AnsiString __fastcall Append_Text_Select(AnsiString field, bool &flag, AnsiString _text, TDateTime _date) {
        AnsiString _sql=(flag) ? " AND " : "";
        if (_text.Length()>0)  _sql+=" ("+field+_text+") ";
                else _sql+=" ("+field+"'"+FormatDateTime("dd/mm/yyyy",_date)+"') ";
        flag=true;
        return _sql;
}
//-------- метод обробки кліку на кнопці "Знайти всіх пацієнтів" ----------
void __fastcall TFormFilertSet::BitBtn_Apply_FilterClick(TObject *Sender) {
        TitleFilters="";
        FormEditInfo->BitBtn_fillterReplace->Enabled=true;
        FormEditInfo->BitBtn_fillterClear->Enabled=true;
        AnsiString _sql=SQL_part1+" WHERE ";
        bool flag=false;
        if (C_Box_Sex->ItemIndex>-1) { C_Box_Sex_ID->ItemIndex = C_Box_Sex->ItemIndex;
                _sql+=Append_Text_Select("a.id_sex=", flag, C_Box_Sex_ID->Text,CurentDate);
                TitleFilters=TitleFilters+"'"+C_Box_Sex->Text+"'; ";
        }
        else Check_Filter_Sex->Checked=false;
        if (C_Box_Region->ItemIndex>-1) { C_Box_Region_ID->ItemIndex = C_Box_Region->ItemIndex;
                _sql+=Append_Text_Select("a.id_region=", flag, C_Box_Region_ID->Text,CurentDate);
                TitleFilters=TitleFilters+"'"+C_Box_Region->Text+"'; ";
        }
        else Check_Filter_Region->Checked=false;
        if (C_Box_Shifr->ItemIndex>-1) { C_Box_Diagnoz_ID->ItemIndex = C_Box_Shifr->ItemIndex;
                _sql+=(flag) ? " AND " : "";
                _sql+=" ((a.id_diagnosis1="+C_Box_Diagnoz_ID->Text+") OR (a.id_diagnosis2="+C_Box_Diagnoz_ID-> Text+") OR
(a.id_diagnosis3="+C_Box_Diagnoz_ID->Text+") OR (a.id_diagnosis4="+C_Box_Diagnoz_ID->Text+"))";
                flag=true;
                TitleFilters=TitleFilters+"'"+C_Box_Diagnoz->Text+"'; ";
        }
        else Check_Filter_Shifr->Checked=false;
        if (Check_Filter_Birth_exactly->Checked || Check_Filter_Birth_in->Checked || Check_Filter_Birth_to->Checked)
                TitleFilters=TitleFilters+"Дата народження - ";
        if (Check_Filter_Birth_exactly->Checked) { _sql+=Append_Text_Select("a.date_birth=", flag, "",DTFilterr_Birth_exactly->Date);
                TitleFilters=TitleFilters+FormatDateTime("dd/mm/yyyy",DTFilterr_Birth_exactly->Date)+"'; ";
        }
        if (Check_Filter_Birth_in->Checked) { _sql+=Append_Text_Select("a.date_birth>=",flag, "", DTFilterr_Birth_in->Date);
                TitleFilters=TitleFilters+"після "+FormatDateTime("dd/mm/yyyy",DTFilterr_Birth_in->Date)+"'; ";
        }
        if (Check_Filter_Birth_to->Checked) { _sql+=Append_Text_Select("a.date_birth<=", flag, "", DTFilterr_Birth_to->Date);
                TitleFilters=TitleFilters+"до "+FormatDateTime("dd/mm/yyyy",DTFilterr_Birth_to->Date)+"'; ";
        }
        if (Check_Filter_Add_exactly->Checked || Check_Filter_Add_in->Checked || Check_Filter_Add_to->Checked)
```

```
                TitleFilters=TitleFilters+"Дата поступлення - ";
        if (Check_Filter_Add_exactly->Checked) { _sql+=Append_Text_Select("a.date_admission=", flag, "", DTFilterr_Add_exactly->Date);
                TitleFilters=TitleFilters+FormatDateTime("dd/mm/yyyy",DTFilterr_Add_exactly->Date)+"; ";
        }
        if (Check_Filter_Add_in->Checked) { _sql+=Append_Text_Select("a.date_admission>=", flag, "", DTFilterr_Add_in->Date);
                TitleFilters=TitleFilters+"після "+FormatDateTime("dd/mm/yyyy",DTFilterr_Add_in->Date)+"; ";
        }
        if (Check_Filter_Add_to->Checked) { _sql+=Append_Text_Select("a.date_admission<=", flag, "", DTFilterr_Add_to->Date);
                TitleFilters=TitleFilters+"до "+FormatDateTime("dd/mm/yyyy",DTFilterr_Add_to->Date)+"; ";
        }
        if (Check_Filter_Dec_exactly->Checked || Check_Filter_Dec_in->Checked || Check_Filter_Dec_to->Checked)
                TitleFilters=TitleFilters+"Дата виписки - ";
        if (Check_Filter_Dec_exactly->Checked) { _sql+=Append_Text_Select("a.date_discharge=", flag, "", DTFilterr_Dec_exactly->Date);
                TitleFilters=TitleFilters+FormatDateTime("dd/mm/yyyy",DTFilterr_Dec_exactly->Date)+"; ";
        }
        if (Check_Filter_Dec_in->Checked) { _sql+=Append_Text_Select("a.date_discharge>=", flag, "", DTFilterr_Dec_in->Date);
                TitleFilters=TitleFilters+"після "+FormatDateTime("dd/mm/yyyy",DTFilterr_Dec_in->Date)+"; ";
        }
        if (Check_Filter_Dec_to->Checked) { _sql+=Append_Text_Select("a.date_discharge<=", flag, "", DTFilterr_Dec_to->Date);
                TitleFilters=TitleFilters+"до "+FormatDateTime("dd/mm/yyyy",DTFilterr_Dec_to->Date)+"; ";
        }
        FormEditInfo->OpenChild_AllRecords(_sql);
        FormEditInfo->ActiveControl=FormEditInfo->DBGridFIO;
        this->Close();
}
//------ метод встановлення доступності кнопок в залежності від виконаного вибору ---------
void __fastcall TFormFilertSet::BitBtn_Apply_Enabled(TObject *Sender) {
        BitBtn_Apply_Filter->Enabled=((C_Box_Sex->ItemIndex>-1) || (C_Box_Region->ItemIndex>-1) || (C_Box_Shifr->ItemIndex>-1) ||
Check_Filter_Birth_exactly->Checked || Check_Filter_Add_exactly->Checked || Check_Filter_Dec_exactly->Checked ||
Check_Filter_Birth_in->Checked || Check_Filter_Add_in->Checked || Check_Filter_Dec_in->Checked || Check_Filter_Birth_to->Checked ||
Check_Filter_Add_to->Checked || Check_Filter_Dec_to->Checked);
        BitBtn_Clear_Filter->Enabled=BitBtn_Apply_Filter->Enabled;
}
//-------- метод обробки події вибору прапорця "Стать" ----------
void __fastcall TFormFilertSet::Check_Filter_SexClick(TObject *Sender) {
        C_Box_Sex->Visible=Check_Filter_Sex->Checked;
        C_Box_Sex->ItemIndex =(Check_Filter_Sex->Checked) ? C_Box_Sex->ItemIndex : -1;
        BitBtn_Apply_Enabled(Sender);
        if (Check_Filter_Sex->Checked) ActiveControl=C_Box_Sex;
}
//--------- метод обробки події вибору прапорця "Регіон проживання" ----------
void __fastcall TFormFilertSet::Check_Filter_RegionClick(TObject *Sender) {
        C_Box_Region->Visible=Check_Filter_Region->Checked;
        C_Box_Region->ItemIndex =(Check_Filter_Region->Checked) ? C_Box_Region->ItemIndex : -1;
        BitBtn_Apply_Enabled(Sender);
        if (Check_Filter_Region->Checked) ActiveControl=C_Box_Region;
}
//------- метод обробки події вибору прапорця "точно" в рядку "Дата народження" ---------
void __fastcall TFormFilertSet::Check_Filter_Birth_exactlyClick(TObject *Sender) {
        DTFilterr_Birth_exactly->Visible=Check_Filter_Birth_exactly->Checked;
        BitBtn_Apply_Enabled(Sender);
        Check_Filter_Birth_in->Visible=!Check_Filter_Birth_exactly->Checked;
        Check_Filter_Birth_to->Visible=!Check_Filter_Birth_exactly->Checked;
        if (Check_Filter_Birth_exactly->Checked) ActiveControl=DTFilterr_Birth_exactly;
}
//--------- метод обробки події вибору прапорця "з" в рядку "Дата народження" ---------
void __fastcall TFormFilertSet::Check_Filter_Birth_inClick(TObject *Sender) {
        DTFilterr_Birth_in->Visible=Check_Filter_Birth_in->Checked;
        BitBtn_Apply_Enabled(Sender);
        Check_Filter_Birth_exactly->Visible=!(Check_Filter_Birth_in->Checked || Check_Filter_Birth_to->Checked);
        if (Check_Filter_Birth_in->Checked) ActiveControl=DTFilterr_Birth_in;
}
//-------- метод обробки події вибору прапорця "по" в рядку "Дата народження" ---------
void __fastcall TFormFilertSet::Check_Filter_Birth_toClick(TObject *Sender) {
        DTFilterr_Birth_to->Visible=Check_Filter_Birth_to->Checked;
        BitBtn_Apply_Enabled(Sender);
        Check_Filter_Birth_exactly->Visible=!(Check_Filter_Birth_in->Checked || Check_Filter_Birth_to->Checked);
        if (Check_Filter_Birth_to->Checked) ActiveControl=DTFilterr_Birth_to;
}
//--------- метод обробки події вибору прапорця "точно" в рядку "Дата поступлення" ----------
void __fastcall TFormFilertSet::Check_Filter_Add_exactlyClick(TObject *Sender) {
        DTFilterr_Add_exactly->Visible=Check_Filter_Add_exactly->Checked;
        BitBtn_Apply_Enabled(Sender);
        Check_Filter_Add_in->Visible=!Check_Filter_Add_exactly->Checked;
        Check_Filter_Add_to->Visible=!Check_Filter_Add_exactly->Checked;
        if (Check_Filter_Add_exactly->Checked) ActiveControl=DTFilterr_Add_exactly;
}
//-------- метод обробки події вибору прапорця "з" в рядку "Дата поступлення" --------
void __fastcall TFormFilertSet::Check_Filter_Add_inClick(TObject *Sender) {
        DTFilterr_Add_in->Visible=Check_Filter_Add_in->Checked;
        BitBtn_Apply_Enabled(Sender);
```

```
        Check_Filter_Add_exactly->Visible=!(Check_Filter_Add_in->Checked || Check_Filter_Add_to->Checked);
        if (Check_Filter_Add_in->Checked) ActiveControl=DTFilterr_Add_in;
}
//-------- метод обробки події вибору прапорця "по" в рядку "Дата поступлення" ---------
void __fastcall TFormFilertSet::Check_Filter_Add_toClick(TObject *Sender) {
        DTFilterr_Add_to->Visible=Check_Filter_Add_to->Checked;
        BitBtn_Apply_Enabled(Sender);
        Check_Filter_Add_exactly->Visible=!(Check_Filter_Add_in->Checked || Check_Filter_Add_to->Checked);
        if (Check_Filter_Add_to->Checked) ActiveControl=DTFilterr_Add_to;
}
//------ метод обробки події вибору прапорця "точно" в рядку "Дата виписки" -------
void __fastcall TFormFilertSet::Check_Filter_Dec_exactlyClick(TObject *Sender) {
        DTFilterr_Dec_exactly->Visible=Check_Filter_Dec_exactly->Checked;
        BitBtn_Apply_Enabled(Sender);
        Check_Filter_Dec_in->Visible=!Check_Filter_Dec_exactly->Checked;
        Check_Filter_Dec_to->Visible=!Check_Filter_Dec_exactly->Checked;
        if (Check_Filter_Dec_exactly->Checked) ActiveControl=DTFilterr_Dec_exactly;
}
//-------- метод обробки події вибору прапорця "з" в рядку "Дата виписки" ----------
void __fastcall TFormFilertSet::Check_Filter_Dec_inClick(TObject *Sender) {
        DTFilterr_Dec_in->Visible=Check_Filter_Dec_in->Checked;
        BitBtn_Apply_Enabled(Sender);
        Check_Filter_Dec_exactly->Visible=!(Check_Filter_Dec_in->Checked || Check_Filter_Dec_to->Checked);
        if (Check_Filter_Dec_in->Checked) ActiveControl=DTFilterr_Dec_in;
}
//--------- метод обробки події вибору прапорця "по" в рядку "Дата виписки" ------
void __fastcall TFormFilertSet::Check_Filter_Dec_toClick(TObject *Sender) {
        DTFilterr_Dec_to->Visible=Check_Filter_Dec_to->Checked;
        BitBtn_Apply_Enabled(Sender);
        Check_Filter_Dec_exactly->Visible=!(Check_Filter_Dec_in->Checked || Check_Filter_Dec_to->Checked);
        if (Check_Filter_Dec_to->Checked) ActiveControl=DTFilterr_Dec_to;
}
//-------- метод обробки події вибору прапорця "Діагноз" ------
void __fastcall TFormFilertSet::Check_Filter_ShifrClick(TObject *Sender) {
        C_Box_Shifr->Visible=Check_Filter_Shifr->Checked;
        C_Box_Diagnoz->Visible=Check_Filter_Shifr->Checked;
        C_Box_Shifr->ItemIndex =(Check_Filter_Shifr->Checked) ? C_Box_Shifr->ItemIndex : -1;
        C_Box_Diagnoz->ItemIndex=C_Box_Shifr->ItemIndex;
        BitBtn_Apply_Enabled(Sender);
        if (Check_Filter_Shifr->Checked)  ActiveControl=C_Box_Shifr;
}
//--------- метод обробки події вибору позиції зі списку шифрів діагнозів ----------
void __fastcall TFormFilertSet::C_Box_ShifrChange(TObject *Sender) {
        C_Box_Diagnoz->ItemIndex=C_Box_Shifr->ItemIndex;
        BitBtn_Apply_Enabled(Sender);
}
//--------- метод обробки події вибору позиції зі списку назв діагнозів ----------
void __fastcall TFormFilertSet::C_Box_DiagnozChange(TObject *Sender) {
        C_Box_Shifr->ItemIndex=C_Box_Diagnoz->ItemIndex;
        BitBtn_Apply_Enabled(Sender);
}
//---------- метод обробки кліку на кнопці "Відмова" ----------
void __fastcall TFormFilertSet::BitBtn_Exit_FilterClick(TObject *Sender) {
        if (!BitBtn_Clear_Filter->Enabled) { FormEditInfo->OpenChild_AllRecords(SQL_part1);
                FormEditInfo->BitBtn_fillterReplace->Enabled=false;
                FormEditInfo->BitBtn_fillterClear->Enabled=false;
        }
        FormEditInfo->ActiveControl=FormEditInfo->DBGridFIO;
}
```

Збережені SQL-процедури

Процедура добавлення в базу інформації про нового пацієнта

```
CREATE PROCEDURE Append_patient
        @_FullName NVARCHAR(50),
        @_date_birth DATETIME,
        @_id_sex int,
        @_id_region int,
        @_id_rayon_city int,
        @_DetailAdress NVARCHAR(50),
        @_date_admission DATETIME,
        @_id_diagnosis1 int,
        @_date_discharge DATETIME = NULL,
        @_id_diagnosis2 int = NULL,
        @_id_diagnosis3 int = NULL,
        @_id_diagnosis4 int = NULL,
        @_Now_therapy NVARCHAR(600) = NULL,
        @id INT OUTPUT
AS
BEGIN
        SET NOCOUNT ON;
```

```
        INSERT INTO [dbo].[Child]
(FullName,date_birth,id_sex,id_region,id_rayon_city,DetailAdress,date_admission,date_discharge,id_diagnosis1,
id_diagnosis2,id_diagnosis3,id_diagnosis4,Now_therapy)
    VALUES
        (@_FullName, @_date_birth, @_id_sex, @_id_region, @_id_rayon_city, @_DetailAdress, @_date_admission, @_date_discharge,
@_id_diagnosis1,@_id_diagnosis2,@_id_diagnosis3,@_id_diagnosis4, @_Now_therapy)
    SET @id= @@IDENTITY
END
```

Процедура оновлення інформації в базі існуючого пацієнта

```
CREATE PROCEDURE Update_patient
        @id_rec INT,
        @_FullName NVARCHAR(50),
        @_date_birth DATETIME,
        @_id_sex int,
        @_id_region int,
        @_id_rayon_city int,
        @_DetailAdress NVARCHAR(50),
        @_date_admission DATETIME,
        @_id_diagnosis1 int,
        @_date_discharge DATETIME = NULL,
        @_id_diagnosis2 int = NULL,
        @_id_diagnosis3 int = NULL,
        @_id_diagnosis4 int = NULL,
        @_Now_therapy NVARCHAR(600) = NULL
AS
BEGIN
        SET NOCOUNT ON;
        UPDATE [dbo].[Child] Set
                FullName=@_FullName,
                date_birth=@_date_birth,
                id_sex=@_id_sex,
                id_region=@_id_region,
                id_rayon_city=@_id_rayon_city,
                DetailAdress=@_DetailAdress,
                date_admission=@_date_admission,
                date_discharge = @_date_discharge,
                id_diagnosis1 = @_id_diagnosis1,
                id_diagnosis2 = @_id_diagnosis2,
                id_diagnosis3 = @_id_diagnosis3,
                id_diagnosis4 = @_id_diagnosis4,
                Now_therapy = @_Now_therapy
        WHERE ID=@id_rec
END
```

Процедура створення всіх звітів за всі періоди звітності

```
CREATE PROCEDURE CreateZvit_All
        @date_zvit DATETIME,
        @wot_zvit int        – 1 - квартал; 2 - півріччя; 3 - 9 місяців; 4 - рік
AS
BEGIN
        EXECUTE CreateZvit_Incidence  @date_zvit, @wot_zvit     –(з роподілом на хлопчики та дівчатка)
        EXECUTE CreateZvit_Diagnosis  @date_zvit, @wot_zvit     –(з роподілом по нозологіях)
        EXECUTE CreateZvit_Region  @date_zvit, @wot_zvit        –(з роподілом за регіонами)
END
```

Процедура формування звіту з охоплення лікування за віковими категоріями

```
CREATE PROCEDURE CreateZvit_Incidence
        @date_zvit DATETIME,
        @wot_zvit int        – 1 - квартал; 2 - півріччя; 3 - 9 місяців; 4 - рік
AS
BEGIN
        TRUNCATE TABLE Zv_IncidenceMonth
        TRUNCATE TABLE Zv_IncidenceQuarter
        TRUNCATE TABLE Zv_IncidenceHalfYear
        TRUNCATE TABLE Zv_Incidence9Month
        TRUNCATE TABLE Zv_IncidenceYear
        Declare @i int
        DECLARE @boys_adm int=0
        DECLARE @boys_disc int=0
        DECLARE @girls_adm int=0
        DECLARE @girls_disc int=0
        DECLARE @Month int =MONTH(@date_zvit)
        DECLARE @date_beg DATETIME =DATEFROMPARTS(YEAR(@date_zvit), @Month, 1)
        DECLARE @date_end DATETIME =EOMONTH(@date_zvit)
        SET @i=0
        While (@i<=15) begin
```

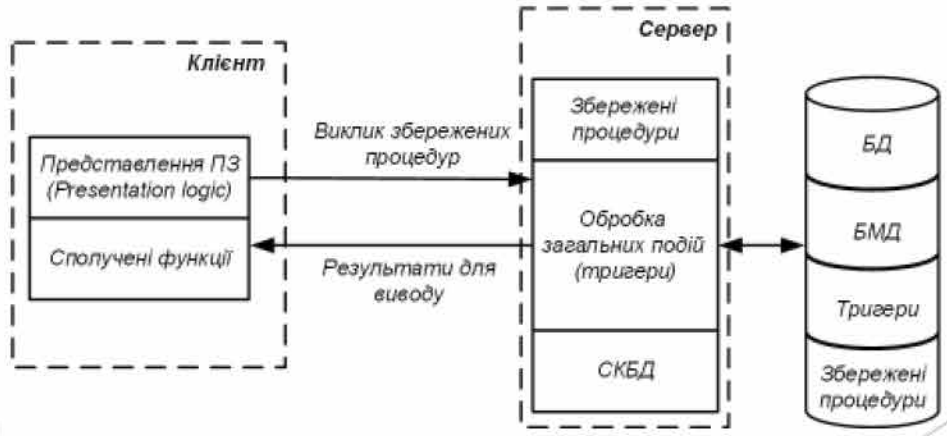
```
                    EXECUTE Calc_ChidrenIncidence  @date_beg, @date_end, @i, @boys_adm OUTPUT, @boys_disc OUTPUT,
@girls_adm OUTPUT, @girls_disc OUTPUT
                    INSERT INTO Zv_IncidenceMonth ([Years], [Boys_adm], [Boys_disc], [Girls_adm], [Girls_disc])
                    VALUES ('від '+str(@i,2)+ ' до '+str(@i+1,2)+' р.', @boys_adm, @boys_disc, @girls_adm, @girls_disc)
                    set @i=@i+1
          end
          UPDATE Zv_IncidenceMonth Set Boys=Boys_adm+Boys_disc, Girls=Girls_adm+Girls_disc,
Together_adm=Boys_adm+Girls_adm, Together_disc=Boys_disc+Girls_disc, Together=Boys_adm+Boys_disc+Girls_adm+Girls_disc
          if (@wot_zvit<>0) begin
                    set @date_beg = DATEFROMPARTS(YEAR(@date_zvit), @Month-2, 1)
                    set @date_end = EOMONTH(@date_zvit)
                    SET @i=0
                    While (@i<=15) begin
                              EXECUTE Calc_ChidrenIncidence  @date_beg, @date_end, @i, @boys_adm OUTPUT, @boys_disc
OUTPUT, @girls_adm OUTPUT, @girls_disc OUTPUT
                              INSERT INTO Zv_IncidenceQuarter ([Years], [Boys_adm], [Boys_disc], [Girls_adm], [Girls_disc])
                              VALUES ('від '+str(@i,2)+ ' до '+str(@i+1,2)+' р.', @boys_adm, @boys_disc, @girls_adm, @girls_disc)
                              set @i=@i+1
                    end
                    UPDATE Zv_IncidenceQuarter Set Boys=Boys_adm+Boys_disc, Girls=Girls_adm+Girls_disc,
Together_adm=Boys_adm+Girls_adm, Together_disc=Boys_disc+Girls_disc, Together=Boys_adm+Boys_disc+Girls_adm+Girls_disc
          end
          if (@wot_zvit=2 or @wot_zvit=3 or @wot_zvit=4) begin
                    set @date_beg = DATEFROMPARTS(YEAR(@date_zvit), 1, 1)
                    set @date_end = EOMONTH(@date_zvit)
                    SET @i=0
                    While (@i<=15) begin
                              EXECUTE Calc_ChidrenIncidence  @date_beg, @date_end, @i, @boys_adm OUTPUT, @boys_disc
OUTPUT, @girls_adm OUTPUT, @girls_disc OUTPUT
                    if (@wot_zvit=2) begin
                    INSERT INTO Zv_IncidenceHalfYear ([Years], [Boys_adm], [Boys_disc], [Girls_adm], [Girls_disc])
                    VALUES ('від '+str(@i,2)+ ' до '+str(@i+1,2)+' р.', @boys_adm, @boys_disc, @girls_adm, @girls_disc)
                              end;
                    if (@wot_zvit=3) begin
                    INSERT INTO Zv_Incidence9Month ([Years], [Boys_adm], [Boys_disc], [Girls_adm], [Girls_disc])
                    VALUES ('від '+str(@i,2)+ ' до '+str(@i+1,2)+' р.', @boys_adm, @boys_disc, @girls_adm, @girls_disc)
                              end;
                              if (@wot_zvit=4) begin
                                        INSERT INTO Zv_IncidenceYear ([Years], [Boys_adm], [Boys_disc], [Girls_adm], [Girls_disc])
                                        VALUES ('від '+str(@i,2)+ ' до '+str(@i+1,2)+' р.', @boys_adm, @boys_disc, @girls_adm,
@girls_disc)
                              end
                              set @i=@i+1
                    end
                    if (@wot_zvit=2) begin
                              UPDATE Zv_IncidenceHalfYear Set Boys=Boys_adm+Boys_disc, Girls=Girls_adm+Girls_disc,
Together_adm=Boys_adm+Girls_adm, Together_disc=Boys_disc+Girls_disc, Together=Boys_adm+Boys_disc+Girls_adm+Girls_disc
                    end
                    if (@wot_zvit=3) begin
                              UPDATE Zv_Incidence9Month Set Boys=Boys_adm+Boys_disc, Girls=Girls_adm+Girls_disc,
Together_adm=Boys_adm+Girls_adm, Together_disc=Boys_disc+Girls_disc, Together=Boys_adm+Boys_disc+Girls_adm+Girls_disc
                    end
                    if (@wot_zvit=4) begin
                              UPDATE Zv_IncidenceYear Set Boys=Boys_adm+Boys_disc, Girls=Girls_adm+Girls_disc,
Together_adm=Boys_adm+Girls_adm, Together_disc=Boys_disc+Girls_disc, Together=Boys_adm+Boys_disc+Girls_adm+Girls_disc

                    end
          end
END
```

ДОДАТОК Б. Слайди мультимедійної презентації

Розробка системи обліку даних пацієнтів медичного закладу

Плиска Вадим Петрович

Модель роботи клієнт-серверної архітектури



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Плиска Вадим Петрович

(прізвище, ім'я та по батькові)

Спеціальність: *123 «Комп'ютерна інженерія»*

Освітня програма: *«Обслуговування комп'ютерних систем і мереж»*

Тема дипломного проекту: *Розробка системи обліку даних пацієнтів медичного закладу*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 96 сторінок. У пояснювальній записці докладно описані методи та розробка програмного забезпечення для електронного реєстру медичної установи. Графічна частина складається з окремих слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Плиска В.П. поступово та послідовно виконував всі етапи, проявив ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Плиска В.П. під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою.*

Вважаю, що теоретична підготовка дипломника достатня і він готовий до захисту проекту.

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Плиска В.П. показав вміння організовано працювати над поставленим завданням, застосовувати знання у галузі програмування, електроніки, математики, розробляти, встановлювати та налаштовувати спеціалізоване програмне забезпечення, складати презентації, користуючись сучасними комп'ютерними програмними засобами, такими як Embarcadero RAD Studio C++ та Microsoft SQL Server, Microsoft PowerPoint, Microsoft Visio.*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проекту *Скорняков В.С.*

Місце роботи і посада керівника дипломного проекту *ВСП "Одеський технічний фаховий коледж ОНТУ", викладач спецдисциплін комісії комп'ютерних технологій та програмної інженерії.*

Підпис

«10» 06 2023 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Плиски Вадима Петровича

(прізвище, ім'я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Освітня програма *«Обслуговування комп'ютерних систем та мереж»*

Керівник дипломного проекту (роботи) *Скорняков В'ячеслав Сергійович*

(прізвище, ім'я та по батькові)

Тема дипломного проекту (роботи) *Розробка системи обліку даних пацієнтів медичного закладу*

Обсяг розрахунково-пояснювальної записки *98* сторінок

Обсяг графічної (презентаційної) частини *15* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений проблемі обліку даних пацієнтів медичного закладу та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації, що містить приклади роботи програми.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (аналізу предметної області, проектування, програмної реалізації, тестування програмного забезпечення), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 15 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скріншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки відмінна, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Реалізовано систему обліку даних пацієнтів, що може бути використана для організації роботи медичного закладу.*

Система має простий та зручний інтерфейс.

База даних передбачає додавання розширеної інформації.

д) основні недоліки дипломного проекту

Графічний інтерфейс користувача виглядає перевантаженим.

Складна організація таблиць бази даних.

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *Царьов Роман Юрійович*

Місце роботи і посада рецензента *Державний університет інтелектуальних технологій і зв'язку, ст. викладач, зав. кафедри комп'ютерної інженерії та інформаційних систем*

Підпис:

« 18 » 07 2024 р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григоріївна Краснокутська

ID перевірки:
1016374155

Дата перевірки:
19.06.2024 06:41:27 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
19.06.2024 06:44:38 EEST

ID користувача:
100011688

Назва документа: 4КС-57_Плиска

Кількість сторінок: 42 Кількість слів: 5278 Кількість символів: 39353 Розмір файлу: 2.34 MB ID файлу: 1016181844

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

16.9%
Схожість

Найбільша схожість: 5.31% з Інтернет-джерелом (https://ela.kpi.ua/handle/123456789/49671)

16.9% Джерела з Інтернету	231

Сторінка 44

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	161

Підозріле форматування	9 сторінок

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Плиска В.П.,
здобувач освіти гр. 4КС-57, та

Скорняков В.С.,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка системи обліку даних пацієнтів медичного закладу» (автор роботи – Плиска В.П., керівник роботи – Скорняков В.С.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Плиска В.П./

Керівник _____ / Скорняков В.С. /

«10» червня 2024 р.