ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут телекомунікацій

Кафедра системного аналізу

Пояснювальна записка

до бакалаврської роботи
на ступінь вищої освіти бакалавр
на тему: **«Інформаційна система підтримки прийняття рішень лікаря-терапевта»**

Виконала: студентка 4 курсу,
групи САД–41 спеціальності
124 Системного аналізу_____
(шифр і назва спеціальності)

Гончарук М.Ю.
(прізвище та ініціали)

Керівник Кузьміч М.Ю.
(прізвище та ініціали)

Рецензент_____Кузьміч М.Ю._____
(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут телекомунікацій

Кафедра Системного аналізу_____
Ступінь вищої освіти - «Бакалавр»_____
Спеціальність - 124 Системний аналіз_____

ЗАТВЕРДЖУЮ
Завідувач кафедри
Системного аналізу
_____Гордієнко Т.Б
“ ____”
_____2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

_____Гончарук Микола Юрійович_____

(прізвище, ім‘я, по батькові)

1.Тема роботи: Інформаційна система підтримки прийняття рішень лікаря-терапевта

Керівник роботи____Кузьміч М.Ю._____,

(прізвище, ім‘я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “ “_____2023 року №__.

2. Строк подання студентом роботи_____

3. Вхідні дані до роботи:

Література з питань оформлення ;

Інструменти розробки додатків, мова програмування Delphi;

Науково-технічна література з питань, пов‘язаних з оформленням додатка.

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

1. Аналізувати предметну область;
2. Розробити ER-діаграму;
3. Розробити алгоритм роботи додатка;
4. Спроектувати та розробити систему для додатка;

5. Реалізувати інтерфейсні форми, орієнтовані на специфіку задачі, які забезпечують зручну взаємодію користувача.

Перелік графічного матеріалу

1 Презинтація

2.

3.

4.

Дата видачі завдання

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	До 06.04	виконано
2	Моделювання та проектування	До 06.04	виконано
3	Програмна реалізація додатку	13.04-26.04	виконано
4	Реалізація системи	27.04-09.05	виконано
5	Висновки+презентація	09.05-13.05	виконано
6	Перевірка роботи на антиплагіат+Передзахист	14.05-04.06	виконано
7	Захист роботи	04.06-21.06	виконано
8	Випуск	29.06	виконано

Студент _____ М.Ю.Гончарук
(підпис) (прізвище та ініціали)

Керівник роботи _____ М.Ю.Кузьміч
(підпис) (прізвище та ініціали)

ЗМІСТ

ВСТУП

Актуальність дослідження. Сучасна медицина вимагає швидкого доступу до великих обсягів інформації, яка постійно зростає, і лікарі часто стикаються з необхідністю швидко та ефективно обробляти ці дані для прийняття вагомих медичних рішень.

Інформаційні системи підтримки прийняття рішень стають невід'ємною складовою лікарської практики, допомагаючи лікарям аналізувати, інтерпретувати та використовувати інформацію для прийняття найоптимальніших рішень у лікуванні пацієнтів. Такі системи надають лікарям доступ до медичних баз даних, клінічних протоколів, наукових досліджень та експертних рекомендацій, що сприяє поліпшенню якості медичної допомоги і зниженню ризиків помилок.

Результати дослідження допоможуть встановити ефективність та значимість інформаційних систем підтримки прийняття рішень для лікарів терапевтів, сприятимуть усвідомленому впровадженню таких систем у медичну практику та покращенню якості надання медичної допомоги пацієнтам.

Інформаційні системи підтримки прийняття рішень для лікарів терапевтів можуть мати значний вплив на покращення процесу прийняття медичних рішень. Вони забезпечують доступ до актуальних медичних даних, допомагають в інтерпретації результатів обстежень та діагностичних тестів, а також надають рекомендації щодо оптимального лікування на основі клінічних протоколів та експертних думок.

Завдяки таким системам, лікарі терапевти можуть швидше здійснювати діагностику, визначати оптимальні схеми лікування та моніторити стан пацієнтів на основі об'єктивних даних. Це сприяє підвищенню точності діагнозу, зменшенню ризиків помилок та покращенню результатів лікування. Крім того, інформаційні системи підтримки прийняття рішень можуть сприяти

стандартизації медичної практики, забезпечуючи однаковість підходів до діагностики та лікування різних захворювань.

Однак, при впровадженні інформаційних систем підтримки прийняття рішень важливо враховувати деякі виклики. Наприклад, необхідно забезпечити надійну захист і конфіденційність медичної інформації, оскільки це стосується особистих даних пацієнтів. Крім того, важливо враховувати індивідуальні потреби та професійний досвід лікарів, не перетворюючи їх на прості виконавців інструкцій системи. Лікар завжди має мати можливість аналізувати та оцінювати інформацію, приймати остаточні рішення і враховувати унікальні аспекти кожного пацієнта.

Медицина постійно розвивається, і з'являється все більше нової інформації про захворювання, методи діагностики та лікування. Лікарі терапевти мають багато джерел інформації, яку необхідно обробляти і аналізувати для прийняття вірного рішення. Врахувати всі можливі варіанти та індивідуальні особливості пацієнта може бути складно, і саме тут інформаційна система підтримки прийняття рішень стає незамінною.

Зростає обсяг медичної інформації, яка є розпорошеною в різних джерелах: наукові статті, клінічні дослідження, медичні протоколи, дані обстежень пацієнтів тощо. Лікарям терапевтам потрібна система, яка допоможе систематизувати та організувати цю інформацію, забезпечуючи їм швидкий та зручний доступ до неї.

Враховуючи складність деяких медичних ситуацій та зростання кількості хронічних захворювань, лікарі терапевти зіштовхуються з необхідністю прийняття рішень в умовах обмеженого часу та ресурсів. Інформаційні системи підтримки прийняття рішень можуть надати їм підтримку, аналізуючи дані, надаючи рекомендації та допомагаючи з вибором найефективніших терапевтичних підходів.

Інформаційні системи підтримки прийняття рішень можуть сприяти стандартизації медичної практики, забезпечуючи консистентність в рішеннях лікарів терапевтів та сприяючи підвищенню якості надання медичної допомоги.

Отже, дана тема актуальна, оскільки впровадження інформаційних систем підтримки прийняття рішень може покращити ефективність та безпеку лікування пацієнтів, допомогти лікарям терапевтам зменшити ризики помилок та забезпечити найкращі результати лікування.

Але, незважаючи на це, сьогодні існує потреба у дослідженні, яке б узагальнило, систематизувало існуючі відомості з даної проблеми.

Враховуючи все вищесказане, нами і була обрана тема дипломної роботи: "Інформаційна система підтримки прийняття рішень лікаря терапевта".

Об’єкт дослідження – Загально-теоретична характеристика інструментальних засобів разробки сучасних додатків.

Предмет – методирозробки інформаційної системи підтримки прийняття рішень лікаря терапевта.

Мета роботи: Метою є визначення способів покращення якості медичної допомоги та забезпечення ефективного використання інформаційних ресурсів для прийняття обґрунтованих та ефективних рішень лікарем терапевтом.

Відповідно до мети були визначені наступні **завдання**:

1) провести аналіз сучасних інструментальних засобів додатка для підтримки прийняття рішень лікаря терапевта;

2) обґрунтувати особливості експериментального методу дослідження ефективності використання додатка для підтримки прийняття рішень лікаря терапевта;

3) проаналізувати основні аспекти проектування й розробки додатка для підтримки прийняття рішень лікаря терапевта;

4) проести дослідження ефективності використання додатка для підтримки прийняття рішень лікаря терапевта;

Для розв’язання поставлених завдань нами були використані такі **методи дослідження**: теоретико-критичний аналіз літератури з теми дослідження; зіставлення, узагальнення і синтезування здобутої інформації тощо.

Робота може бути використана студентами ВНЗ для підготовки до семінарських занять, також може бути використана викладачами для проведення лекції, практик тощо.

Структура роботи. Робота складається зі вступу, трьохрозділів, висновків, списку використаних джерел, що містить 62 найменування. Повний обсяг роботи: 82 сторінки.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналіз предметної області

Терапевт - фахівець в області терапії; лікар, що спеціалізується на виявленні, лікуванні, профілактиці внутрішніх хвороб. Він здійснює первинну діагностику, координує взаємодія пацієнта з іншими фахівцями, виписує напрямку на більшість обстежень і процедур, оформляє медичну документацію.

Основні завдання цього фахівця:

- первинний прийом пацієнтів, збір анамнезу, проведення огляду й інших об'єктивних методів обстеження;
- рання діагностика на підставі результату обстеження пацієнта й аналізу його скарг;
- первинна консультація, роз'яснення пацієнтові причин його недуги. Хороший терапевт виступає ще й у ролі психолога, заспокоюючи хворого, надаючи йому інформацію про захворювання в потрібному обсязі;
- призначення ліків, фізіотерапевтичних процедур і інших лікувальних заходів у межах своєї компетенції [24 с. 86];
- призначення лабораторних аналізів і інструментального обстеження;
- у випадку ускладненого плину або неясного генеза захворювання - напрямок до профільного фахівця для більш детальної діагностики й проходження лікування відповідно до його рекомендацій;
- розробка єдиної схеми лікування з урахуванням рекомендацій різних вузьких фахівців;
- ухвалення рішення про госпіталізацію;
- оцінка ризику розвитку хронічного захворювання й вживання заходів до його зниження;

- консультації щодо зміцнення імунітету, профілактики ускладнень, рецидивів і переходу захворювання в хронічну форму;
- регулярне спостереження пацієнтів із хронічними захворюваннями;
- розробка рекомендацій щодо зміни способу життя, умов праці, санаторно-курортного лікування й інших;
- призначення схеми комплексного медичного обстеження при проходженні профосмотра, медкомісії;
- огляд перед вакцинацією й прийняття розв'язок щодо її проведення.

Далі був складений список виявлених користувацьких вимог:

- зберігання даних про пацієнта (Прізвище, Ім'я, По батькові, дата народження і т.д.);

- зберігання даних про запис на прийом (коли й скільки раз був на прийманні, по якій причині) [19, с. 47];

- зберігання даних про історію лікування пацієнта (які ліки були призначені, напрямок на аналізи);

- зберігання даних про реабілітацію пацієнта (якими ліками відбувалося лікування, чи допомогли вони; якщо не допомогли, призначення нових препаратів);

- відомості, отримані шляхом розпиту пацієнта (обстежуваного);

- керування даними про запис на прийом (зміна, видалення, додавання, сортування);

- керування даними, отриманими шляхом розпиту пацієнта (зміна, видалення, додавання, сортування);

- керування даними про лікування пацієнта (зміна, видалення, додавання, сортування);

- розмежування доступу (Для того, щоб захистити дані пацієнта);

- розробка інтерфейсу (Для прискорення прийом пацієнтів).

Наступним кроком виявимо функціональні вимоги для того, щоб виконати користувацькі вимоги. Функціональні вимоги (functional requirements) визначають функціональні можливості розроблювального ПО для реалізації користувацьких вимог. Виділимо дані вимоги для нашого ключового бізнес процесу:

- база даних, що містить таблиці "Особисті дані пацієнтів", "Дати відвідувань", "Причини відвідування", "Лікування пацієнта", "Напрямок на аналіз", "Аналіз сечі", "Аналіз калу", "Аналіз крові", "Реабілітація пацієнта";
- виведення на екран даних з перерахованих вище таблиць [25, .с 87];
- додавання даних про пацієнта;
- редагування даних про пацієнта у всіх таблицях;
- пошук конкретного пацієнта по ПІБ у всіх таблицях;
- програма повинна коректно працювати на всіх комп'ютерах медзакладу;
- дані зберігаються безпосередньо в створюваному додатку;
- установлення пароля на саму базу даних;
- розробка інтерфейсу для простоти використання продукту.

Приоритизация вимог дозволяє зрозуміти, які вимоги користувачеві необхідно реалізувати в першу чергу й на що слід звернути особливу увагу. Це дозволить уникнути зайвих матеріальних і тимчасових витрат на проектування й розробку модулів або функціонала. Розставимо пріоритети для раніше представлених користувацьких і функціональних вимог:

Пріоритет №1 - дані вимоги є основою розроблювальної системи. Без виконання даного пункту реалізувати систему не вийде.

Пріоритет №2 - дані вимоги критичні для функціональних можливостей системи, але не сильно відіб'ються на роботі програми при їхній відсутності.

Пріоритет №3 - дані вимоги не є ключовими для повноцінного функціонування самої програми, але їх було б непогано реалізувати.

Після збору й аналізу користувацьких і функціональних вимог створюється список вимог. Він дозволяє зв'язати й зіставити всі вимоги, їх пріоритети й програмні компоненти, відповідальні за реалізацію вимог. Список вимог полегшує процес відстеження вимог розроблювачем і дозволяє впевнитися в їхнім повному виконанні перед закінченням робіт. Вимоги для розроблювального продукту представлено в Таблиці 1.1. [31, с. 60]

Таблиця 1.1.

Список вимог для розроблювального продукту

№	Користувацька вимога	Функціональна вимога	Програмна компонент	Пріоритет вимоги
1	Зберігання даних про пацієнта	Таблиця даних "Особисті дані пацієнтів"	Програма по веденню даних	Пріоритет №1
2	Зберігання даних про запис на прийом	Таблиця даних "Дати відвідувань"		
3	Відомості, отримані шляхом розпиту пацієнта	Таблиця даних "Причини відвідування"		
4	Зберігання даних про історію лікування пацієнта	Таблиця даних "Лікування пацієнта"		
5	Зберігання даних про те, на який аналіз був спрямований пацієнт	Таблиця даних "Напрямок на аналіз"		

6	Зберігання даних про результати аналізу сечі	Таблиця даних "Аналіз сечі"		
7	Зберігання даних про результати аналізу калу	Таблиця даних "Аналіз калу"		
8	Зберігання даних про результати аналізу крові	Таблиця даних "Аналіз крові"		
9	Зберігання даних про те, як відбувається реабілітація пацієнта	Таблиця даних "Реабілітація пацієнта"		
10	Керування даними про запис на прийом (зміна, видалення, додавання, сортування)	Можливість редагування, видалення, додавання, пошуку й сортування записів у таблиці даних "Дати відвідувань"	Програма для додавання, редагування, видалення, пошуку, сортування й перегляду даних	Пріоритет №2
11	Керування даними, отриманими шляхом розпиту пацієнта (зміна,	Можливість редагування, видалення, додавання, пошуку й сортування записів у		

	видалення, додавання, сортування)	таблиці даних "Причини відвідування"		
12	Керування даними про лікування пацієнта	В таблиці даних "Лікування пацієнта"		
13	Керування даними про напрямок на аналіз	У таблиці даних "Напрямок на аналіз"		
14	Керування даними про аналіз сечі	В таблиці даних "Аналіз сечі"		
15	Керування даними про аналіз калу	В таблиці даних "Аналіз калу"		
16	Керування даними про аналіз крові	В таблиці даних "Аналіз крові"		
17	Керування даними про реабілітацію пацієнта	В таблиці даних "Реабілітація пацієнта"		
18	Розмежування доступу	Встановлення пароля на саму базу даних	Програма авторизації користувача	Пріоритет №3
19	Розробка інтерфейсу	Розробка інтерфейсу	Програма спрощення роботи користувача	

1.2. Огляд існуючих програмних засобів

Під час розгляду питання щодо створення додатку для моніторингу здоров'я пацієнта виявилося, що найпопулярнішим рішенням є використання технологій, що функціонують тільки на платформі iOS або Android. В зв'язку з цим, було проведено аналіз програмних засобів, розроблених відомими компаніями.

Проведено аналіз програмного рішення Apple HealthKit, яке виявилося потужним інструментом для покращення медичного обслуговування. Цей додаток дозволяє зберігати та відстежувати дані про здоров'я та фізичні активності користувача на пристроях iPhone та Apple Watch. Завдяки цьому рішенню, користувач може зберігати всі свої важливі дані в одному місці та моніторити свій стан здоров'я на щодень [14 ,с. 97].

Одна з важливих функцій HealthKit полягає в тому, що користувач може легко поділитися своїми медичними даними зі своїм лікарем або іншими медичними фахівцями. Це значно спрощує комунікацію між пацієнтом і медичним персоналом, дозволяючи швидше та ефективніше здійснювати консультації та діагностику.

Apple HealthKit реалізовано на платформі iOS, використовуючи мову програмування Swift, що забезпечує швидку та надійну роботу додатку. Це рішення, розроблене компанією Apple, вже має велику популярність та широке застосування серед користувачів, які прагнуть активно контролювати свій стан здоров'я та підтримувати здоровий спосіб життя.

Попри популярність даного додатку, він має свої недоліки, які варто врахувати. Перш за все, його обмежена сумісність з платформами, оскільки він працює лише на iOS, що ускладнює можливість використовувати його на інших пристроях. Крім того, додаток не має інтернаціоналізації, тому ним можуть користуватися тільки англомовні користувачі.

Рис. 1.1. Apple HealthKit

З урахуванням зазначених вимог до програмного рішення, можна виділити наступні критичні аспекти: зрозумілий інтерфейс, багатофункціональність, підтримка режиму безпідключення до Інтернету та можливість завантажувати та обмінюватися документами зі своїм терапевтом. Варто зазначити, що додаток має обмежену сумісність і працює лише на платформі iOS, а також підтримує тільки англійську мову [39, с. 65].

Однак, з метою покращення додатку, можна врахувати ці недоліки та працювати над їх виправленням. Наприклад, розширити сумісність додатку з іншими платформами, надати можливість вибору мови інтерфейсу для

користувача, розробити версію додатку з мультилінгвальною підтримкою. Такі кроки сприятимуть поширенню та поліпшенню функціональності додатку, забезпечуючи зручність та доступність його використання для більш широкого кола користувачів.

Проведено аналіз програмного рішення Google Fit, яке, подібно до Apple HealthKit, розширює можливості медичного обслуговування. Цей додаток дозволяє користувачеві збирати широкий спектр інформації про їх здоров'я та передавати ці дані своєму лікарю. Важливою особливістю даного додатка є його широкий функціонал, який включає опитувальники, можливість завантаження даних та спілкування з терапевтом.

Google Fit підтримує платформи iOS та Android, що робить його доступним для багатьох користувачів. Інтерфейс додатка також надає можливість вибору мови користувача, що сприяє більш широкому застосуванню додатка на ринку.

Однак, слід відзначити кілька недоліків Google Fit. Перш за все, додаток вимагає значний обсяг пам'яті, що може бути проблематичним для деяких користувачів, які мають обмежені ресурси пристрою. Крім того, згідно з джерелом [1], багато користувачів стикаються з проблемами оновлення додатка, що може призвести до нестабільної роботи та втрати функціональності.

З метою поліпшення Google Fit, можна розглянути можливість оптимізації використання пам'яті та вирішення проблем оновлення, щоб забезпечити більш широку доступність та стабільну роботу додатка для всіх користувачів [20, с. 88].

Проведений аналіз показав, що дане програмне рішення відповідає вимогам, які були визначені раніше. Додаток має багатий функціонал, що дозволяє виконувати різноманітні завдання пов'язані з моніторингом здоров'я. Його інтернаціоналізація дозволяє користувачам з різних країн використовувати додаток на їх рідній мові. Наявність підтримки для платформ iOS та Android розширює базу користувачів і забезпечує доступність додатку на різних пристроях.

Однак, треба відзначити деякі недоліки даного рішення. Відсутність можливості обміну документами з терапевтом обмежує комунікаційні можливості між пацієнтом та медичним персоналом. Також великий обсяг пам'яті, необхідний для функціонування додатку, може стати проблемою для користувачів з обмеженими ресурсами пристроїв. Додатково, проблеми з оновленням додатку можуть призвести до незручностей для користувачів, а також вплинути на його функціональність та безпеку.

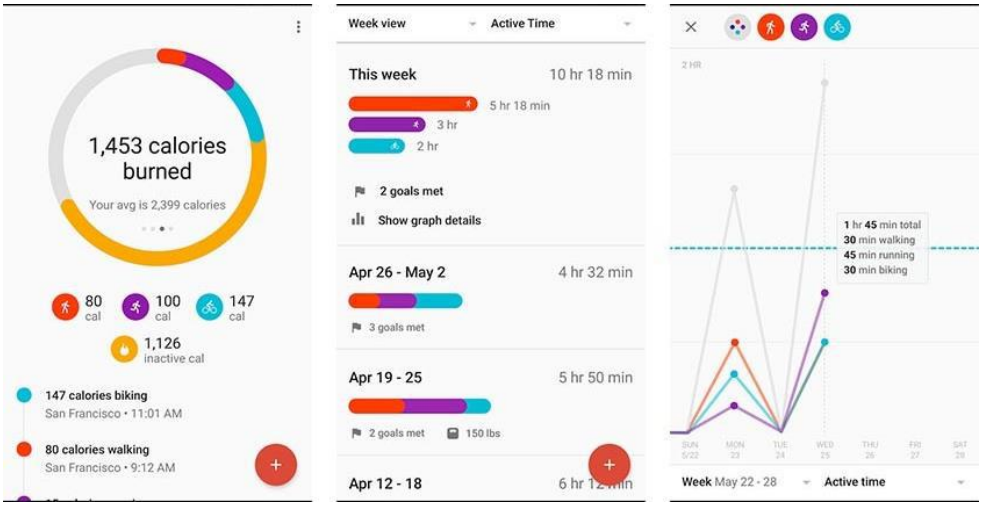


Рис. 1.2. Medisafe

Детально проаналізувавши мобільний додаток Medisafe, можна відзначити його значно більший функціонал у порівнянні з попередніми розглянутими рішеннями. Цей додаток дозволяє відстежувати прийом будь-яких прописаних ліків, надаючи більш гнучкі можливості користувачам. Важливою особливістю Medisafe є можливість додавати як регулярні ліки (курсові), так і одноразові, які можуть бути прийняті в залежності від потреби (наприклад, знеболюючі або снодійні препарати) [17, с. 46].

Додаток також надає зручність у спілкуванні з лікарем, дозволяючи завантажувати звіти у форматах PDF і XLS[3]. Користувачі можуть внести дані про свій тиск, рівень глюкози в крові, вагу та інші параметри для подальшого аналізу та моніторингу. Додаток пропонує також інші зручні опції, які, можливо, менш важливі, але все життєво необхідні. Наприклад, синхронізація нагадувань

з годинником Android Wear або можливість встановлення різних мелодій для окремих препаратів, що допомагає розрізняти їх.

На жаль, деякі недоліки варто відмітити. Розробники додатку поки не включили функцію, яка показуватиме початок і кінець прийому ліків, а також кількість прийнятих і залишених таблеток (наразі програма враховує лише кількість днів курсу). Також відсутня можливість встановлення перерви в прийомі медикаментів, що може бути важливим для деяких користувачів, які мають особливі потреби у графіку прийому ліків [10, с. 48].

Окрім цього, розглянуті переваги додатку Medisafe, такі як більш широкий функціонал, можливість інтернаціоналізації та наявність зручного інтерфейсу, сприяють поліпшенню медичного обслуговування та сприяють більш ефективному контролю над лікуванням.

Рис. 1.3. Ілюстрація роботи Medisafe

З урахуванням поставлених вимог для аналізу програмного рішення, Medisafe виявляється переважною альтернативою. Цей додаток має багатофункціональний підхід, який включає відстеження прописаних ліків, можливість додавати як регулярні, так і одноразові ліки, та забезпечує зручний спосіб спілкування з лікарем.

Крім того, Medisafe надає можливість роботи без доступу до Інтернету, що важливо для користувачів, які часто перебувають у зоні з обмеженим зв'язком.

Завдяки підтримці інтернаціоналізації, додаток стає доступним для користувачів різних країн і культур, що розширює його ринкову привабливість [23, с. 87].

Інтуїтивно-зрозумілий інтерфейс сприяє зручній взаємодії з системою, дозволяючи користувачам швидко орієнтуватися і використовувати всі доступні функції. Проте, варто зазначити, що Medisafe обмежений доступністю лише на платформі Android, що може вплинути на користувацьку базу, оскільки користувачі інших платформ не зможуть скористатися цим додатком.

1.3. Інструменти для розробки

При розробці інформаційної системи, були задіяні наступні програмні засоби:

1 Borland Delphi 7 - інтегроване середовище розробки (ИСР) програмного додатка.

2 Lucidchart - веб-додаток для розробки інформаційних схем.

3 Mysql - система керування базами даних.

4 Microsoft Word - текстовий редактор.

У якості системи керування базами даних (СУБД) обрана Mysql. Даний вибір обумовлений принципом роботи, який дозволяє забезпечувати повнофункціональну роботу бази даних при низьких навантаженнях на апаратне забезпечення [29, с. 65].

При порівнянні продуктивності запитів, Mysql упевнено лідирує перед іншими СУБД, наприклад, такими як MSSQL і Oracle.

Крім того, вибір даної СУБД обумовлений наступними її характеристиками:

- Mysql є клієнт- серверної СУБД;
- Mysql вільно розповсюджуваний продукт;
- Mysql є крос- платформним продуктом;

- Mysql є реляційною базою даних;

Основні характеристики СУБД Mysql:

- Многопоточность. Підтримка декількох одночасних запитів;
- Оптимізація зв'язків із приєднанням багатьох даних за один прохід;
- Записи фіксованої й змінної довжини;
- ODBC драйвер у комплекті з вихідним файлом;
- Гнучка система привілеїв і паролів;
- До 16 ключів у таблиці. Кожний ключ може мати до 15 полів;
- Підтримка ключових полів і спеціальних полів в операторові;
- Підтримка чисел довгої від 1 до 4 байт (ints, float, double, fixed), рядків змінної довжини й міток часу;
- Інтерфейс із мовами C і perl;
- Заснована на потоках, швидка система пам'яті;
- Утиліта перевірки й ремонту таблиці (isamchk);
- Усі дані зберігаються у форматі ISO8859_1;
- Усі операції роботи з рядками не обертають уваги на регістр символів в оброблюваних рядках;
- Псевдоніми застосовні як до таблиць, так і до окремих стовпчиків у таблиці;
- Усі поля мають значення за замовчуванням. Можна використовувати на будь-якій підмножині полів;
- Легкість керування таблицею, включаючи додавання й видалення ключів і полів.

Обґрунтування вибору Lucidchart

У якості засобу розробки моделей бізнес- процесів, був обраний візуальний засіб для побудови діаграм на основі HTML5 - Lucidchart. Сервіс містить велика кількість зразків і прикладів:

- блок схем;

- Uml-Моделей;
- Er- Моделей даних;
- каркасів / макетів;
- системних діаграм;
- моделей бізнес- процесів (BPMN 2.0);
- організаційних схем;
- схем зв'язків;
- схем сайтів;

Також сервіс Lucidchart дає можливість збереження створених діаграм у форматах: PDF, PNG і JPG

Обґрунтування вибору Microsoft Office Word

Microsoft Word є однієї із самих затребуваних програм сучасності, використовуваних в офісній роботі. Даний текстовий редактор - найбільш потужний і розповсюджений [40, с. 36].

Будучи текстовим редактором, він дозволяє суттєво полегшити роботу з написання текстів. Ведення кореспонденції, обробка тексту, створення ділової й офіційної переписки - усе це стало простіше проводити завдяки Microsoft Word. Вигідними особливостями програми є її функціональність і простота використання, завдяки чому навіть недосвідчений користувач здатний легко розібратися в ній.

Microsoft Word забезпечує простої й зручне форматування текстового файлу за рахунок продуманих інструментів і зрозумілого інтерфейсу. Завдяки широкому спектру функцій текстовий процесор Microsoft Word нагадує настільну видавничу систему. Серед функціональних можливостей програми можна відзначити:

-наявність цілого ряду шрифтів різного розміру й накреслення символів;

-наявність способів виділення тексту;

-можливість установити параметри абзаців, міжрядкові інтервали;

-можливість проведення автоматичної перевірки правопису, добору синонімів;

-автоматична нумерація сторінок і переноси слів на новий рядок;

-можливість створення таблиць і гіпертексту з посиланнями й багато чого іншого.

Як видне, функціонал даної програми дуже різноманітний, завдяки цьому текст буде мати привабливий вигляд, зручний для читання й розуміння. Крім того, можливість автоформатирования, застосування стилів, а також заздалегідь підібрані шаблони дозволяють робити роботу з Microsoft Word простий і зручної.

Таким чином, програма Microsoft Word - це самий зручний варіант для створення тексту і його наступного форматування з можливістю гнучкого настроювання шрифту, стилю написання, оформлення самої сторінки.

На даній моделі відображені всі існуючі таблиці бази даних, а також зв'язки між таблицями.

Er-Модель бази даних створена за допомогою онлайн- сервісу Lucidchart.

1.4. Вибір мови програмування

Для розробки додатка для тестування локальної комп'ютерної мережі можна використовувати різні мови програмування, але Delphi може бути відмінним вибором з декількох причин.

По-перше, Delphi є мовою програмування з великим потенціалом для створення графічного інтерфейсу користувача. Це означає, що розробники можуть швидко і легко створювати інтуїтивно зрозумілі інтерфейси для взаємодії з додатком для тестування мережі [24, с. 87].

По-друге, Delphi має багато вбудованих функцій та компонентів, які дозволяють розробникам легко тестувати та моніторити мережу. Деякі з цих компонентів включають в себе TCP / IP і UDP, що дозволяє розробникам легко

здійснювати з'єднання з різними мережними пристроями, а також моніторити трафік мережі [31, с. 64].

По-третє, Delphi є мовою програмування з довголітнім стажем, що означає, що вона має широке співтовариство розробників та налагоджену інфраструктуру. Це забезпечує наявність різноманітних інструментів, ресурсів та документації, які дозволяють розробникам легко створювати та підтримувати додатки.

Таким чином, використання Delphi для розробки додатка для тестування локальної комп'ютерної мережі може бути кращим вибором для розробників, які хочуть швидко створювати інтуїтивно зрозумілі інтерфейси, моніторити та тестувати мережу, а також мати доступ до налагодженої інфраструктури та ресурсів.

Delphi - імперативний, структурований, об'єктно-орієнтований, высокоуровневый мова програмування зі строгою статичною типізацією змінних. Основна область використання - написання прикладного програмного забезпечення.

Ця мова програмування є діалектом мови Object Pascal. Споконвічно мова Object Pascal ставився до трохи іншої мови, яка була розроблений у фірмі Apple в 1986 році групою Ларри Теслера. Однак, починаючи з Delphi 7, в офіційних документах компанії Borland назва Delphi стало використовуватися для позначення мови, раніше відомого як Object Pascal.

Delphi - це мова програмування, заснований на Pascal. Споконвічно він створювався для розробки додатків для ОС Windows, потім став застосовуватися й для інших цілей. Зараз він не занадто популярний, а сфера його використання досить вузька, але мова усе ще потрібний [25, с. 89].

Синтаксис Delphi схожий на Pascal, адже ця мова - по суті, діалект "паскаля". Відмінність Delphi в тому, що він об'єктно-орієнтований, тобто сутності в ньому представлені як об'єкти з методами й властивостями.

Більш рання назва Delphi - Object Pascal. Також Delphi називається середовище розробки для мови. Зараз її повна назва - Embarcadero Delphi, хоча за час існування вона перемінила кілька імен.

Мова створювалася для розробки комп'ютерних додатків. Зараз сфера його застосування ширше й одночасно вже. Ним користуються великі проекти з багаторічною історією для підтримки свого ПО, він задіяний у ряді специфічних сфер - наприклад, системи контролю й обліку доступу, електронні черги, внутрішні корпоративні програми. Також Delphi використовується при роботі з базами даних. Часто його використовують для підтримки legacy- коду - такого, який був написаний давно й уважається морально застарілим.

Delphi можна зустріти в сфері інди-геймдева. Наприклад, на ньому написаний популярний движок Gamemaker для створення простих ігор.

Ще один варіант застосування мови - навчання. Pascal, на якому він заснований, створювався в тому числі як навчальна мова, і звідси виросли його особливості: чіткість і зрозумілість коду, строгість синтаксису. Це було потрібно, щоб не заплутувати початківців, а в підсумку привело до того, що мова полегшало розширювати. Зараз від навчання на Pascal і Delphi потроху відходять, але ця практика дотепер збереглася.

Об'єктно-орієнтований підхід. Delphi створений для розробки в парадигмі ООП. У ньому реалізовані всі основні принципи об'єктно-орієнтованого програмування: спадкування, поліморфізм, інкапсуляція, є об'єкти й класи як особливий тип даних. При цьому він дозволяє писати не тільки в об'єктно-орієнтованому підході - просто орієнтований на нього [24, с. 75].

Чіткість і структурованість. Мова створена дуже ясним і однозначним. У нього строгий синтаксис, а значна частина речей задається явно й вручну. Споконвічно це було зроблено для навчання, але пізніше дозволило уникнути сильного ускладнення мови, коли він розширився. Синтаксис улаштований так,

щоб не допускати неоднозначності. Багато речей, які теоретично дозволені в інших, менш "твердих" мовах, в Delphi вважаються помилкою.

Стругаючи типізація. Типи в Delphi привласнюються строго, явно й статично. Це означає, що при створенні змінної розроблювач відразу вказує її тип, і він не міняється надалі . При спробі привласнити цієї змінної значення іншого типу програма не компілюється й видає помилку. Такий підхід робить мова менш гнучкою, але захищає від несподіваних помилок. Тобто якщо розроблювач привласнить змінної щось не те, він зрозуміє це ще на етапі компіляції, а не після відправлення працюючої на вид програми на деплой.

Ручне керування пам'яттю. Більшість сучасних мов, на яких пишуть десктопные або мобільні додатки, працюють із пам'яттю автоматично. Спеціальні процеси виділяють частина оперативної пам'яті для розв'язку завдань програми, а збирачі сміття пізніше очищають невикористовувану пам'ять. Розроблювачеві не потрібно управляти цим процесом. Але в Delphi всі не так.

В Delphi розроблювач і виділяє, і очищає пам'ять вручну, окремими командами. Автоматичної роботи з оперативною пам'яттю за замовчуванням не передбачене. Здається, начебто це робить написання коду складніше, але на практиці це дозволяє зробити додаток більш швидким і менш вимогливим до ресурсів комп'ютера. Правда, від розроблювача потрібно певний рівень кваліфікації, інакше можна помилитися й допустити витік пам'яті.

Консервативність. Мова міняється повільно, і його основні принципи залишаються незмінними. Програмування на Delphi зараз будується приблизно по тим же особливостям, що раніше. Консервативний підхід підтримується й строгою типізацією, і загальною "твердістю" синтаксису. Зараз, коли мови постійно міняються, ця особливість - і плюс, і мінус. Мінус - тому що мові важко встигати за новими тенденціями. Плюс - тому що програми на Delphi залишаються передбачуваними. Тому мову нерідко використовують там, де важливо, щоб система залишалася стабільної довгий час [18, с. 65].

Отже, Delphi є потужною мовою програмування, яка має багато переваг для розробки додатків для тестування локальної комп'ютерної мережі. Зокрема, ця мова дозволяє легко створювати інтуїтивно зрозумілі інтерфейси, моніторити та тестувати мережу, а також мати доступ до налагодженої інфраструктури та ресурсів завдяки широкому співтовариству розробників. Тому використання Delphi може бути кращим вибором для розробки додатка для тестування локальної комп'ютерної мережі.

Для розробки додатків для тестування локальної комп'ютерної мережі, мова Delphi може бути вигідною з багатьох причин.

Delphi має простий та зрозумілий синтаксис, що дозволяє легко створювати інтуїтивно зрозумілі інтерфейси. Зокрема, це дозволяє розробникам швидко створювати графічний інтерфейс для тестування мережі, який буде зрозумілий і простий для користувачів.

Delphi має багато вбудованих бібліотек, які дозволяють з легкістю моніторити та тестувати локальну мережу. Наприклад, компоненти Indy можуть бути використані для розробки різноманітних мережевих додатків, таких як клієнти, сервери та протоколи. Крім того, Delphi надає доступ до налагодженої інфраструктури та ресурсів завдяки широкому співтовариству розробників.

Delphi має багато інструментів для налагодження та тестування додатків, таких як компілятори та інтегровані середовища розробки. Ці інструменти дозволяють розробникам ефективно відлагоджувати та тестувати свої додатки, що є важливим етапом у розробці будь-якого програмного забезпечення.

Таким чином, використання мови Delphi може бути вигідним для розробки додатків для тестування локальної комп'ютерної мережі завдяки її простоті, вбудованим бібліотекам, налагоджувальним інструментам та широкому співтовариству розробників.

1.5. Постановка задачі

Даний підрозділ повинен містити в собі вимоги до системи в цілому, до функцій, виконуваних системою, видам забезпечення. Склад вимог залежить від виду, призначення, специфічних особливостей розроблювальної системи.

Залежно від виду системи приводяться вимоги до інформаційного, програмному, технічному, лінгвістичному, метрологічному, методичному, організаційному й іншим видам забезпечення проектованої й впроваджуваної автоматизованої системи.

Проблеми, які можуть виникнути при створенні програмного забезпечення для автоматизації який або системи, не завжди можна зрозуміти. Дуже важко чітко описати ті дії, які повинна виконувати система. Опис функціональних можливостей і обмежень, що накладаються на систему, називається вимогами до цієї системи. Вимоги до продукту повинні бути встановлені таким чином, що могло б гарантувати їхня адекватність і вірний "переклад" з мови користувача. Для початку необхідно виявити всі можливі вимоги, пропоновані до розроблювального програмного продукту.

Виявити їх можна за допомогою аналізу вимог - частина процесу розробки програмного забезпечення, що включає в себе збір вимог до програмного забезпечення (ПО), їхню систематизацію, виявлення взаємозв'язків, а також документування. У процесі збору вимог важливо брати до уваги можливі протиріччя вимог різних зацікавлених осіб, таких як замовники, розроблювачі або користувачі. Процес формування й аналізу вимог проходить через ряд етапів:

- Аналіз предметної області. Аналітики повинні вивчити предметну область, де буде експлуатуватися система.

- Збір вимог. Це процес взаємодії з особами, що формують вимоги. Під час цього процесу триває аналіз предметної області.

- Призначення пріоритетів. У будь-якому наборі вимог одні з них будуть більш важливі, чому інші. На цьому етапі разом з особами, що формують вимоги, визначаються найбільш важливі вимоги. Ідентифікація попередніх вимог до нового продукту дозволяє, відповідно до вимог клієнта, визначити характеристики нового продукту.

За останні роки комп'ютерні технології сильно розбудовуються й впроваджуються в усі можливі сфери життя людину. Вони мають великий перелік функціональних можливостей, у тому числі й можливість підвищити ефективність роботи лікаря лікувальної установи.

У недалекому минулому лікувальні установи: поліклініки й працюючі в них лікарі, мали певні утруднення при прийманні пацієнтів. Наприклад, дуже довго й трудомістко проходив процес документального оформлення прийом пацієнтів, актуальними були проблеми втрати карти пацієнта в реєстратурі й пошуку її в інших лікарів.

Усе це приводило до утвору черги пацієнтів на прийом до лікаря, утруднялася робота лікарів, медичного персоналу й у цілому робота лікувальної установи. У зв'язку із чим гостро назріло завдання раціоналізувати механізм документального оформлення огляду хворих і прийом пацієнтів у лікувальній установі, що послужило стимулом до широкого впровадження комп'ютерних технологій в область охорони здоров'я. Лікарям необхідно, щоб технічні засоби й розроблене до них програмне забезпечення були простими й зручними при використанні, не порушували звичного стилю роботи [31, с. 78].

Головне проблемою, з якої зустрічаються лікарі при прийманні пацієнта - це недолік часу. Значний час лікаря при прийманні витрачається на ознайомлення з медичною картою обстежуваного. Розробка медичної бази даних і її використання в роботі лікаря дозволить прискорити даний процес, більш якісно проводити обстеження й знизити навантаження на лікаря, дасть можливість швидкого пошуку в базі даних потрібної інформації про пацієнта. Наприклад,

подивитися всю історію хвороби, дати прийом, проведені обстеження і їх результати й ін.

Розробку такого додатка можна зробити в програмі MS Access. Дане середовище сумісне з операційною системою Microsoft Windows, яка використовується на більшості комп'ютерів медичних установ, що важливо для зручності використання.

Метою даної роботи є розробка програмного забезпечення для використання на робочому місці лікаря- терапевта у вигляді автоматизованої системи. Програмне забезпечення дозволить зберігати всю інформацію про пацієнта, його відвідування, історію хвороб, полегшити й прискорити роботу лікаря. Завдяки даній системі знімається проблема із втратою амбулаторних карт пацієнтів, тому що вся інформація буде зберігатися в одній базі даних. Щоб досягтися вищевказаної мети нам потрібно реалізувати наступні завдання:

- зробити детальний аналіз спиралевидной методології впровадження систем;

- ідентифікувати вимоги й сформулювати список вимог;

- зробити проектування процесів і оргструктури в моделях AS- IS і TO-BE нотації ARIS VACD і UML AD до 3-4 рівнів деталізації;

- моделювання розроблювальних користувацьких інтерфейсів;

- проектування структури даних і нормалізація таблиць даних.

Розроблена база даних дозволить упорядкувати й систематизувати роботу терапевта. А це, у свою чергу, може значно підвищити якість роботи, прискорити прийом пацієнтів і відповідно розв'язати проблему із чергами на прийом до лікаря.

РОЗДІЛ 2. МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

2.1. Розробка ER –діаграми

Описана діаграма класів надає загальний огляд основних компонентів системи, включаючи лікаря терапевта, пацієнта, історію медичного обстеження, симптоми, рецепти, базу даних, інтерфейс користувача та систему прийняття рішень. Ця структура може слугувати основою для подальшого розроблення функціональності системи та взаємодії з користувачем [32, с. 64].

Важливо пам'ятати, що розробка детальної архітектури системи вимагає уточнення вимог, аналізу сценаріїв використання та зв'язків між класами. Також рекомендується використовувати добрі практики проектування програмного забезпечення, такі як принципи SOLID, для створення гнучкої та розширюваної системи.

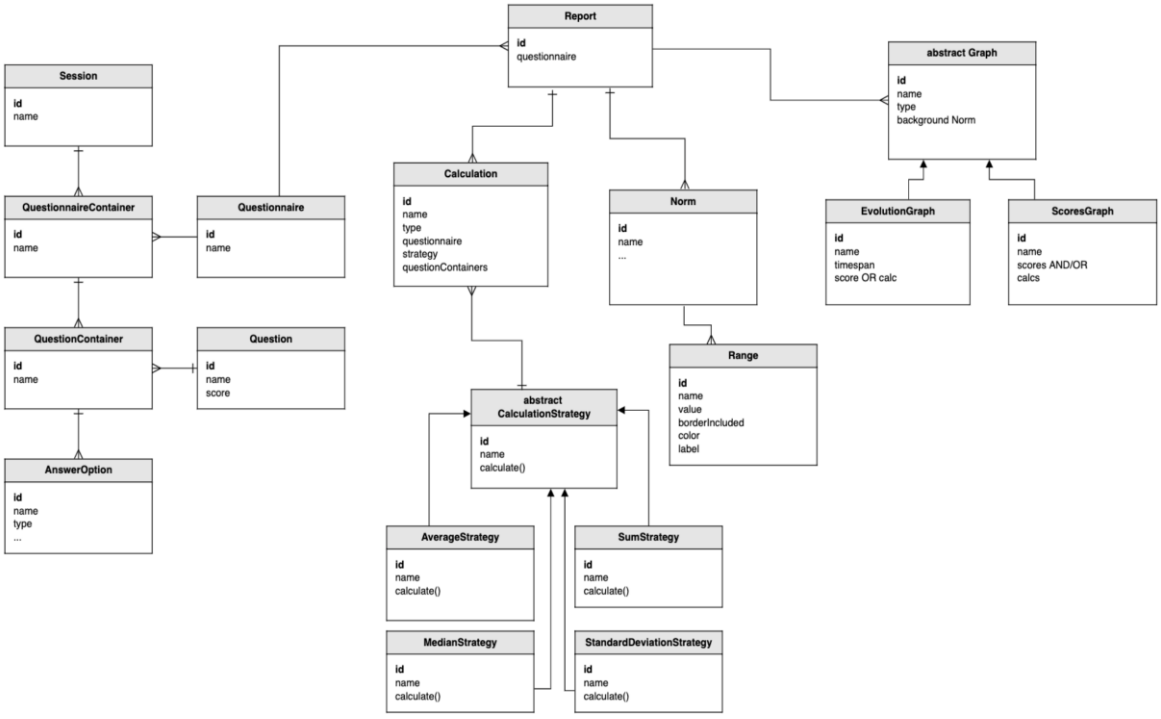


Рис. 2.1. ER-діаграма

Оис діаграми класів для іІнформаційної системи підтримки прийняття рішень лікаря терапевта:

1) Клас "Симптом":

Властивості:

- назва: рядок
- опис: рядок

Методи:

- отримати_назву(): повертає назву симптома
- отримати_опис(): повертає опис симптома

2) Клас "Рецепт":

Властивості:

- назва: рядок
- інструкції: рядок

Методи:

- отримати_назву(): повертає назву рецепту
- отримати_інструкції(): повертає інструкції з використання рецепту

3) Клас "База даних":

Властивості:

- історії: список об'єктів класу "Історія"
- симптоми: список об'єктів класу "Симптом"
- рецепти: список об'єктів класу "Рецепт"

Методи:

- додати_історію(): додає новий об'єкт "Історія" до бази даних
- отримати_історії(): повертає список всіх історій в базі даних
- додати_симптом(): додає новий об'єкт "Симптом" до бази даних
- отримати_симптоми(): повертає список всіх симптомів в базі даних
- додати_рецепт(): додає новий об'єкт "Рецепт" до бази даних
- отримати_рецепти(): повертає список всіх рецептів в базі даних

4) Клас "Інтерфейс користувача":

Методи:

- відображення_історії(): відображає історію пацієнта на екрані
- відображення_симптомів(): відображає доступні симптоми на екрані
- відображення_рецептів(): відображає доступні рецепти на екрані
- отримати_вибір_симптомів(): отримує вибір користувача щодо симптомів
- отримати_вибір_рецепту(): отримує вибір користувача щодо рецепту

5) Клас "Система прийняття рішень":

Властивості:

- база_даних: об'єкт класу "База даних"
- інтерфейс: об'єкт класу "Інтерфейс користувача"

Методи:

- виконати_діагностику(): виконує діагностику пацієнта на основі симптомів
- показати_результати(): відображає результати діагностики та рекомендований рецепт
- виписати_рецепт(): виписує рецепт для пацієнта

Таблиця 2.1.

Перелік діаграм

№	Тип діаграми	Роль у процесі проєктування
1	Діаграма послідовності	У першому наближенні визначає набір класів та їхню послідовність взаємодії для вирішення певної задачі
2	Діаграма класів	Визначає набір класів та їхню взаємодію задля вирішення певної задачі

Діаграма послідовності для роботи додатку наведена у частині графічного матеріалу.

Таблиця 2.2.

На діаграмі задіяні наступні класи

Клас	Відповідальність
TherapistContainer ::React.Component	Відображення форм для заповнення даних
PatientContainer ::React.Component	Обгортка для відображення компонент
Request module	Модуль відправлення запитів
Backend API	Модуль виконання запитів
UsersReducer::ReduxStore	Сховище

Загалом, діаграма класів для Інформаційної системи підтримки прийняття рішень лікаря терапевта може включати наступні класи: "Лікар терапевт", "Пацієнт", "Історія", "Симптом", "Рецепт", "База даних", "Інтерфейс користувача" та "Система прийняття рішень" [21, с. 48].

Клас "Лікар терапевт" представляє лікаря і має методи для отримання історії пацієнта, внесення діагнозу та випису рецепту. Клас "Пацієнт" відображає пацієнта з властивостями, такими як ім'я, вік та список симптомів. Клас "Історія" містить інформацію про медичну історію пацієнта, включаючи пацієнта, діагноз та рецепт.

Класи "Симптом" і "Рецепт" визначають окремі симптоми та рецепти відповідно, з методами для отримання назви та опису. Клас "База даних" зберігає дані про історії, симптоми та рецепти. Він має методи для додавання та отримання даних.

Клас "Інтерфейс користувача" відповідає за відображення інформації для користувача та отримання вибору симптомів та рецептів. Клас "Система прийняття рішень" поєднує всі компоненти системи, виконує діагностику пацієнта на основі симптомів та надає результати діагностики та виписує рецепт.

2.2. Процедури та функції

Бізнес-Процес - це стійка цілеспрямована послідовність виконання функцій, спрямована на створення результату, що має цінність для споживача [6]. При проектуванні ключових бізнес- процесів використовують дві моделі: AS- IS (як є) і TO-BE (як буде):

- модель AS- IS, яка описує стан моделируемой предметної області на момент створення моделі;

- модель TO-BE, що описує можливе майбутній стан предметної області, у яке вона перейде в результаті оптимізації існуючої системи й впровадження нових технологій [7].

Перш ніж намагатися вибрати існуючу або створити власну інформаційну систему, потрібно проаналізувати, як працює система на даний момент часу. Після цього будується функціональна модель AS- IS. Аналіз цієї моделі дозволить виявити недоліки й зрозуміти, у чому будуть полягати переваги нових бізнес- процесів. У моделі TO-BE саме є можливість виправлення таких недоліків. Дана модель потрібна для оцінки наслідків впровадження інформаційної системи й аналізу альтернативних шляхів виконання роботи й документування того, як система буде функціонувати в майбутньому.Бізнес-Процес - це діяльність або набір дій, які виконують певну організаційну мету.Бізнес- Процеси повинні мати цілеспрямованість, бути максимально конкретними й мати послідовні результати [25, с. 76].

Сила методології ARIS полягає в її комплексності, яка проявляється у взаємозв'язку моделей, побудованих у різних нотаціях. Методологія ARIS дозволяє описувати діяльність організації з різних точок зору, при цьому отримані моделі деякою мірою зв'язано між собою [8]. Основними мінусами даного підходу є:

- Наявність інструментального середовища ARIS. Вона є дорогою й досить складною у використанні. Але існує спрощена й безкоштовна версія за назвою ARIS Express.

- Важко реалізовані завдання на практиці, тому що тягнуть велику витрату ресурсів (людських, матеріальних і фінансових) протягом тривалого часу.

Однієї з найважливіших нотацій ARIS є нотація Value-added Chain Diagram. Вона використовується для опису бізнес- процесів організації на верхньому рівні.

Головною відмінністю даної моделі від інших процессних моделей є те, що інформаційні й матеріальні потоки на схемі VACD зображуються не стрілками, а об'єктами. При цьому для кожного типу потоку використовується свій об'єкт. На моделі VACD методології ARIS на відміну від класичного підходу також використовуються логічні зв'язки між роботами, які дозволяють відобразити логічну послідовність виконання робіт [42, с. 58].

Дана нотація підходить для вивчення організації в цілому. Але для виявлення можливих проблем цієї нотації може бути недостатньо, вона допомагає визначити ті процеси, які й потребують змін і доробці. При описі бізнес- процесів скористаємося додатком ARIS Express. Елементи, використовувані в даній нотації, наведено в рис. 2.2.

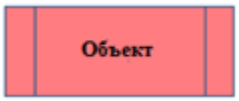
Графічне уявлення	Найменування	Опис
	Процесс	Опис робіт, виконуваних співробітниками організації
	Вхідний або вихідний об'єкт	Елементи, що містять носії інформації
	Організаційна одиниця	Відбиває ланки, відповідальні за виконання процесу

Рис. 2.2. Умовні позначки нотації ARIS VACD

При моделюванні поведінки проектованої системи з'являється необхідність представити процес зміни її станів і деталізувати особливості алгоритмічної й логічної реалізації виконуваних системою операцій. Звичайно для цього використовувалися блок-схеми або структурні схеми алгоритмів. Блок-схема - розповсюджений тип схем, що описують алгоритми або процеси, у яких окремі кроки зображуються у вигляді блоків різної форми, з'єднаних між собою лініями, що вказують напрямок послідовності. У даній роботі один з обраних мов був UML Activity Diagram.

Мова UML (Unified Modeling Language), або уніфікована мова моделювання, призначена для опису, візуалізації, проектування й документування об'єктно-орієнтованих систем і бізнес- процесів з орієнтацією на їхню наступну реалізацію у вигляді програмного забезпечення [19, с. 56].

Для моделювання процесу виконання операцій у мові UML використовуються діаграми діяльності (Activity diagram). Вони служать для моделювання послідовності дій, які виконуються різними елементами, що входять до складу системи. Інакше кажучи, цей вид діаграм розкриває деталі алгоритмічної реалізації операцій, виконуваних системою, тому діаграма діяльності схожа на звичайну блок-схему [10]. Але на відміну від блок-схеми, діаграма діяльності також показує одночасно паралельну й послідовну діяльність.

UML AD нотація проста для вивчення непідготовленим користувачем, вона інтуїтивно зрозуміла. Дана нотація використовує широко відомі графічні елементи (табл.3.2). В UML AD нотації зображення процесів дуже схоже на блок-схеми, тобто багатьом користувачам зображення в UML AD нотації відразу будуть підсвідомо ясні.

Графічний елемент	Опис	Графічний елемент	Опис

Рис. 2.3. Умовні позначки нотації ARIS VACD

На рис. 2.2. і 2.3 розглядається перший рівень проектування процесу роботи лікаря- терапевта в анотації ARIS VACD моделі " AS-IS", яка має на увазі під собою проектування процесів у даний момент часу й моделі "TO-BE" після впровадження електронної бази даних.

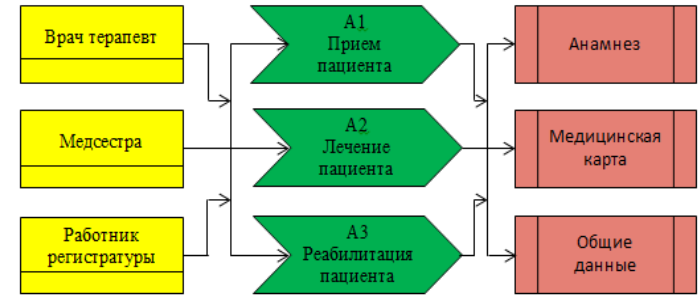


Рис. 2.4. Уявлення процесу в ARIS VACD на першому рівні в моделі " AS-IS"

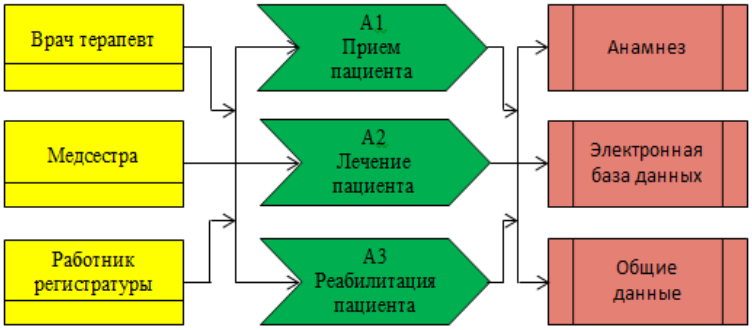


Рис. 2.5. Уявлення процесу в ARIS VACD на першому рівні в моделі "TO-BE"

Для того щоб створити програмне забезпечення, потрібно більше інформації, тому необхідно провести більш докладне проектування. Для цього слід зробити другий і третій рівень деталізації [42, с. 67].

На рис. 2.5 і 2.6 представлений другий рівень деталізації для уточнення процесу "Прийом пацієнта". Другий рівень деталізації процесів "Лікування пацієнта" і "Реабілітація пацієнта" представлений нижче.

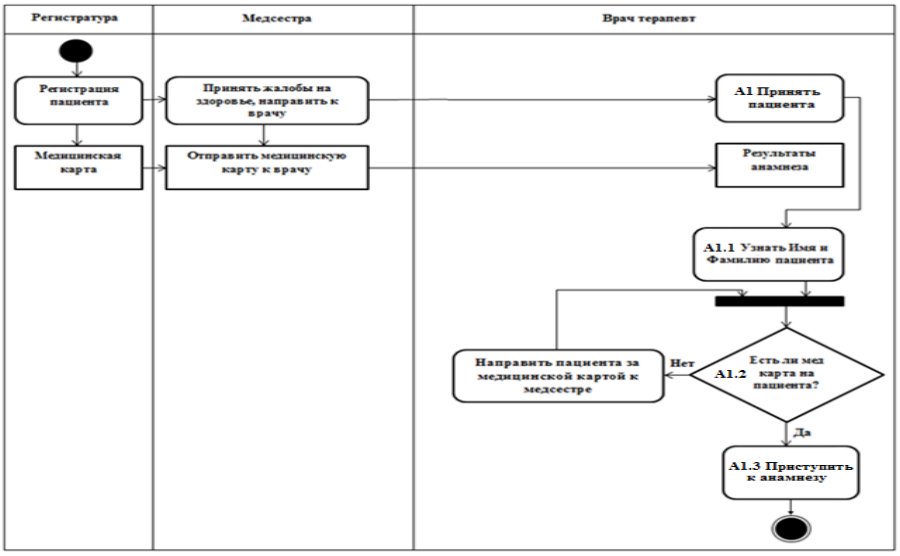


Рис. 2.5. Уявлення подпроцесса в UML AD на другому рівні в моделі " AS-IS" для уточнення процесу "Прийом пацієнта"

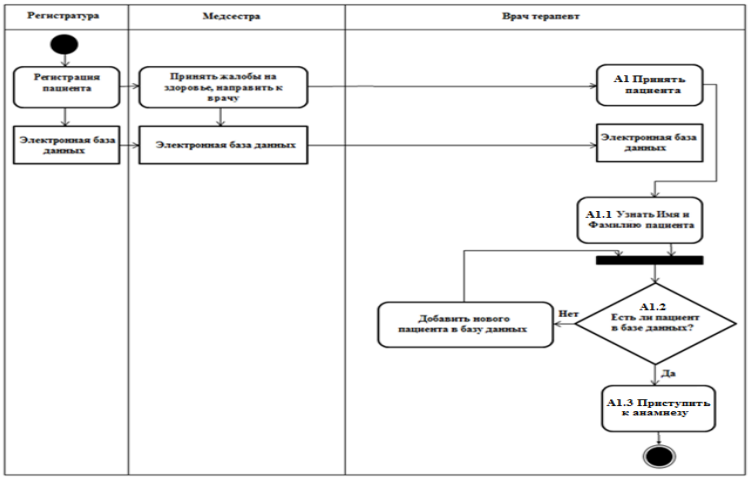


Рис. 2.6. Уявлення подпроцесса в UML AD на другому рівні в моделі "TO-BE" для уточнення процесу "Прийом пацієнта"

На рис. 2.7. і 2.8. представлений третій рівень деталізації для уточнення процесу "Приступитися до анамнезу". Третій рівень деталізації процесів "Лікування пацієнта" і "Реабілітація пацієнта" представлений нижче.

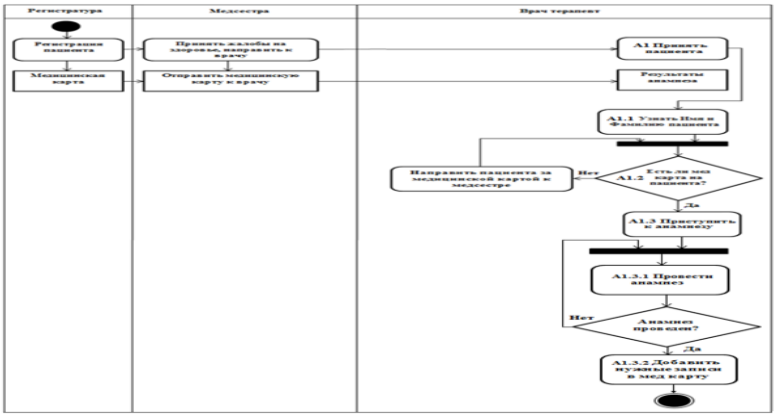


Рис. 2.7. Уявлення процесу в UML AD на третьому рівні в моделі " AS-IS" для уточнення процесу "Приступитися до анамнезу"

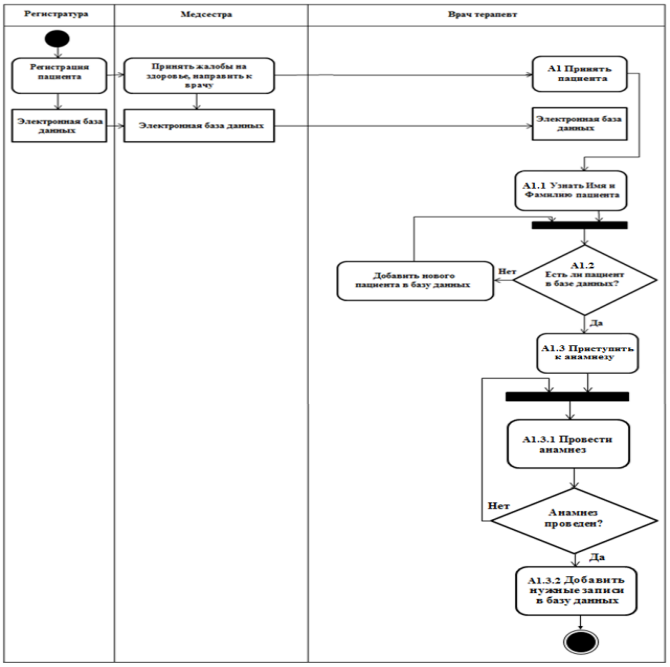


Рис. 2.8. Уявлення процесу в UML AD на третьому рівні в моделі "TO-BE" для уточнення процесу "Приступитися до анамнезу"

Для більш наочної вистави вищевказаних бізнес- процесів на 1-3 рівнях, побудуємо карту бізнес- процесів у моделі "TO-BE" (рис. 2.9).

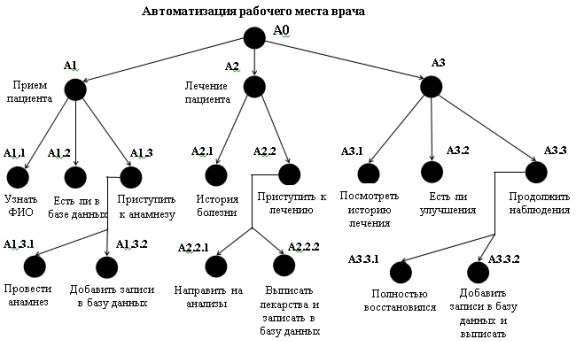


Рис. 2.9. Карта процесів у моделі "TO-BE"

2.3. Архітектура даних розроблювальної системи

Архітектура даних інформаційної системи підтримки прийняття рішень лікаря терапевта може бути складною і різноманітною, враховуючи різні аспекти збору, зберігання та обробки медичної інформації. Основними компонентами такої системи можуть бути:

База даних пацієнтів: Ця складова містить основну інформацію про пацієнтів, таку як особисті дані, анамнез, результати обстежень та діагнози. Дані можуть бути організовані за допомогою структурованої моделі даних, яка дозволяє ефективно зберігати та отримувати інформацію про пацієнтів.

Медичні протоколи та лікарські рекомендації: Ця складова містить стандартизовані протоколи та рекомендації для діагностики та лікування різних захворювань. Вона дозволяє лікарям терапевтам отримувати доступ до актуальної інформації про найкращі практики та рекомендації щодо лікування конкретних хвороб [24, с. 86].

Інструменти аналізу даних: Ці інструменти дозволяють проводити аналіз медичних даних, використовуючи різні методи, такі як статистичний аналіз, машинне навчання та штучний інтелект. Вони можуть надавати лікарям терапевтам важливі інсайти щодо діагностики, прогнозування ходу захворювання та вибору оптимального лікування.

Інтеграція з іншими системами: Інформаційна система підтримки прийняття рішень лікаря терапевта може бути інтегрована з іншими медичними системами, такими як системи електронної медичної картки, системи лабораторних досліджень або системи медичного обладнання. Це дозволяє обмінюватися даними та отримувати повну інформацію про пацієнтів та їх стан з різних джерел.

Заходи безпеки даних: З огляду на конфіденційність медичних даних, система повинна мати механізми захисту даних, такі як шифрування, контроль доступу та аудит дій користувачів. Забезпечення конфіденційності та цілісності медичних даних є критично важливим аспектом архітектури [39, с. 71].

Усі ці компоненти об'єднуються, щоб створити комплексну інформаційну систему підтримки прийняття рішень для лікарів терапевтів. Вона допомагає покращити якість медичного обслуговування, забезпечує доступ до актуальних даних та інструментів аналізу, а також сприяє ефективному взаємодії між лікарем та пацієнтом.

На основі порядку використання даних, розглянутого вище, усю інформацію можна розбити на класи, наведені в таблиці 2.2.

Таблиця 2.2.

Класи даних

Назва класу	Дані	Тип даних	Розмірність
Особисті дані пацієнтів	ПІБ	Текстові	65
	Дата народження	Дата/час	Короткий формат дати
	Домашня адреса	Текстова	70
	Номер паспорта	Текстові	30
	Телефон	Текстові	20

	Підлога	Текстова	1
	ОМС	Текстові	20
Дати відвідувань	ПІБ лікаря	Текстові	85
	Пацієнт	Числовий	Довге ціле
	Дата відвідування	Дата/час	Короткий формат дати
Причина відвідування	Пацієнт	Числовий	Довге ціле
	Причина відвідування	Текстові	200
	Дата відвідування	Числовий	Довге ціле
Лікування пацієнта	Пацієнт	Числовий	Довге ціле
	Причина відвідування	Числовий	Довге ціле
	Дата відвідування	Дата/час	Довгий ціле
	Лікування	Текстові	255
Напрямку на аналіз	Пацієнт	Числовий	Довге ціле
	ПІБ лікаря	Числовий	Довге ціле
	Дата відвідування	Числовий	Довге ціле
	Вид аналізу	Текстові	50
	Друк	Текстові	1
Аналіз крові	Пацієнт	Числовий	Довге ціле
	Показники	Текстові	50
	Норма	Текстові	50
	Результат аналізу	Текстові	50

	Вердикт	Текстові	50
Аналіз сечі	Пацієнт	Числовий	Довге ціле
	Показники	Текстові	50
	Норма	Текстові	50
	Результат аналізу	Текстові	50
	Вердикт	Текстові	50
Аналіз калу	Пацієнт	Числовий	Довге ціле
	Показники	Текстові	50
	Норма	Текстові	50
	Результат аналізу	Текстові	50
	Вердикт	Текстові	50
Реабилитация пациента	Пацієнт	Числовий	Довге ціле
	Лікар	Числовий	Довге ціле
	Дата відвідування	Числовий	Довге ціле
	Дата проведення огляду	Дата/час	Короткий формат дати
	Результат проведення огляду	Текстові	150
	Примітка	Текстова	150

Далі необхідно провести процедуру нормалізації даних. Стандартно всі дані приводять до третьої нормальної форми (НФ), тому як нормалізації такого рівня буває досить. Відповідно першої НФ, в одному полі таблиці повинен зберігатися тільки один атрибут. Таким чином, у класі "Особисті дані пацієнтів"

і "Дати відвідувань" дані "ПІБ" і "ПІБ лікаря" необхідно розбити на три окремі рядки: "Прізвище", "Ім'я", "По батькові".

Відповідно другий НФ, повинні дотримуватися умови першої НФ і кожний не ключовий атрибут повинен залежати від ключа. Отже, у класи "Особисті дані пацієнтів" і "Дати відвідувань" додається графа "Код пацієнта", а в таблиці "Дати відвідувань" і "Причина відвідувань" додається графа "Код причини відвідування" [24, с. 86].

У третьої НФ повинні дотримуватися умови другий НФ і всі не ключові поля, уміст яких може ставитися до декільком записам таблиці, повинне виноситися в окремі таблиці. Таким чином, після поділу всіх даних на класи й проведення процедури нормалізації, виходить нормалізована таблиця класу даних і архітектура розроблювального додатка, наведена на рис. 2.3.

Таблиця 2.3.

Класи даних (нормалізована)

Назва класу	Дані	Тип даних	Розмірність
Особисті дані пацієнтів	🔑 код пацієнта	Лічильник	Довге ціле
	Прізвище	Текстове	20
	Ім'я	Текстові	15
	По батькові	Текстові	30
	Дата народження	Дата/час	Короткий формат дати
	Домашня адреса	Текстова	70
	Номер паспорта	Текстові	30
	Телефон	Текстові	20
	Підлога	Текстова	1

	ОМС	Текстові	20
Дати відвідувань	🔑код відвідування	Лічильник	Довге ціле
	Прізвище лікаря	Текстове	50
	Ім'я лікаря	Текстові	15
	По батькові лікаря	Текстові	20
	Код пацієнта	Числовий	Довге ціле
	Дата відвідування	Дата/час	Короткий формат дати
Причина відвідування	🔑код причини відвідування	Лічильник	Довге ціле
	Код пацієнта	Числовий	Довге ціле
	Причина відвідування	Текстові	200
	Дата відвідування	Числовий	Довге ціле
Лікування пацієнта	🔑код лікування	Лічильник	Довге ціле
	Код пацієнта	Числовий	Довге ціле
	Причина відвідування	Числовий	Довге ціле
	Дата відвідування	Числовий	Довге ціле
	Лікування	Текстові	255
Напрямку на аналіз	🔑код аналізу	Лічильник	Довге ціле
	Код пацієнта	Числовий	Довге ціле
	Прізвище лікаря	Числове	Довге ціле
	Ім'я лікаря	Числовий	Довге ціле
	По батькові лікаря	Числовий	Довге ціле
	Дата відвідування	Числовий	Довге ціле

	Вид аналізу	Текстові	50
	Друк	Текстові	1
Аналіз крові	🔑 код аналізу	Лічильник	Довге ціле
	Код пацієнта	Числовий	Довге ціле
	Показники	Текстові	50
	Норма	Текстові	50
	Результат аналізу	Текстові	50
	Вердикт	Текстові	50
Аналіз сечі	🔑 код аналізу	Лічильник	Довге ціле
	Код пацієнта	Числовий	Довге ціле
	Показники	Текстові	50
	Норма	Текстові	50
	Результат аналізу	Текстові	50
	Вердикт	Текстові	50
Аналіз калу	🔑 код аналізу	Лічильник	Довге ціле
	Код пацієнт	Числовий	Довге ціле
	Показники	Текстові	50
	Норма	Текстові	50
	Результат аналізу	Текстові	50
	Вердикт	Текстові	50
Реабілітація пацієнта	🔑 код реабілітації	Лічильник	Довге ціле
	Код пацієнта	Числовий	Довге ціле
	Лікар	Числовий	Довге ціле
	Дата відвідування	Числовий	Довге ціле

	Дата проведення огляду	Дата/час	Короткий формат дати
	Результат проведення огляду	Текстові	150
	Примітка	Текстова	150

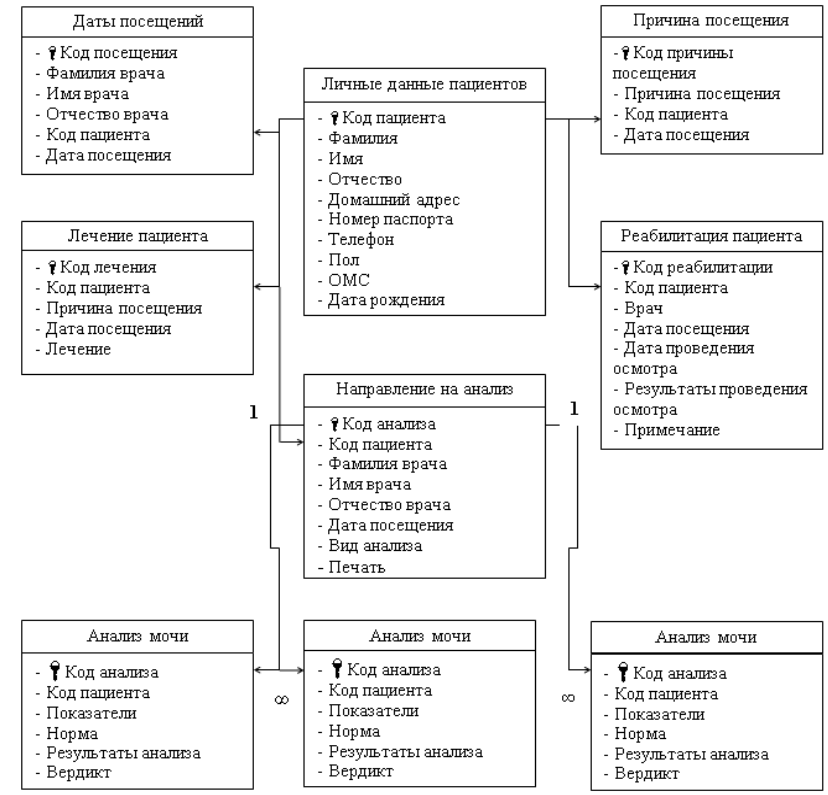


Рис. 2.10. Архітектура даних розроблювального додатка

Архітектура даних є ключовим аспектом розробки інформаційної системи підтримки прийняття рішень лікаря терапевта. Вона визначає, як дані будуть зберігатися, організовуватися та оброблятися в системі [40 ,с .32].

Важливим елементом архітектури даних є вибір відповідного типу бази даних. У контексті системи підтримки прийняття рішень лікаря терапевта можуть використовуватися реляційні бази даних або нереляційні (NoSQL) бази даних в залежності від вимог до швидкодії, гнучкості та обсягу даних.

Структура бази даних повинна відображати основні сутності системи, такі як лікарі, пацієнти, медична історія, симптоми, рецепти тощо. Застосування концепції нормалізації даних допомагає уникнути дублювання інформації та забезпечує цілісність даних.

Важливо враховувати вимоги до безпеки та конфіденційності медичних даних. Застосування механізмів шифрування, ролевого контролю доступу та моніторингу може допомогти забезпечити захист даних в системі.

Врахування масштабованості інформаційної системи також є важливим аспектом архітектури даних. Планування розподіленої архітектури та використання кластеризації або реплікації даних можуть забезпечити високу доступність та продуктивність системи [8, с. 54].

В процесі розробки архітектури даних розроблювальної інформаційної системи підтримки прийняття рішень лікаря терапевта важливо брати до уваги специфіку медичного домену та вимоги користувачів. Краща практика - проведення аналізу вимог, планування та тестування архітектури даних для забезпечення високої якості та ефективності системи.

Загалом, правильно спроектована архітектура даних є ключовим фактором успіху інформаційної системи підтримки прийняття рішень лікаря терапевта. Вона забезпечує ефективну організацію, зберігання та обробку даних, а також забезпечує безпеку, доступність та масштабованість системи.

РОЗДІЛ 3. РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКУ

3.1. Опис розроблювального додатка

Для найбільш зручної роботи кінцевого користувача із програмою необхідно створити максимально зрозумілі користувацькі інтерфейси програми. При вході в базу даних користувачеві має бути вибрати, чи потрібне йому додати нового або подивитися особисті дані пацієнта, або подивитися історію його хвороби.

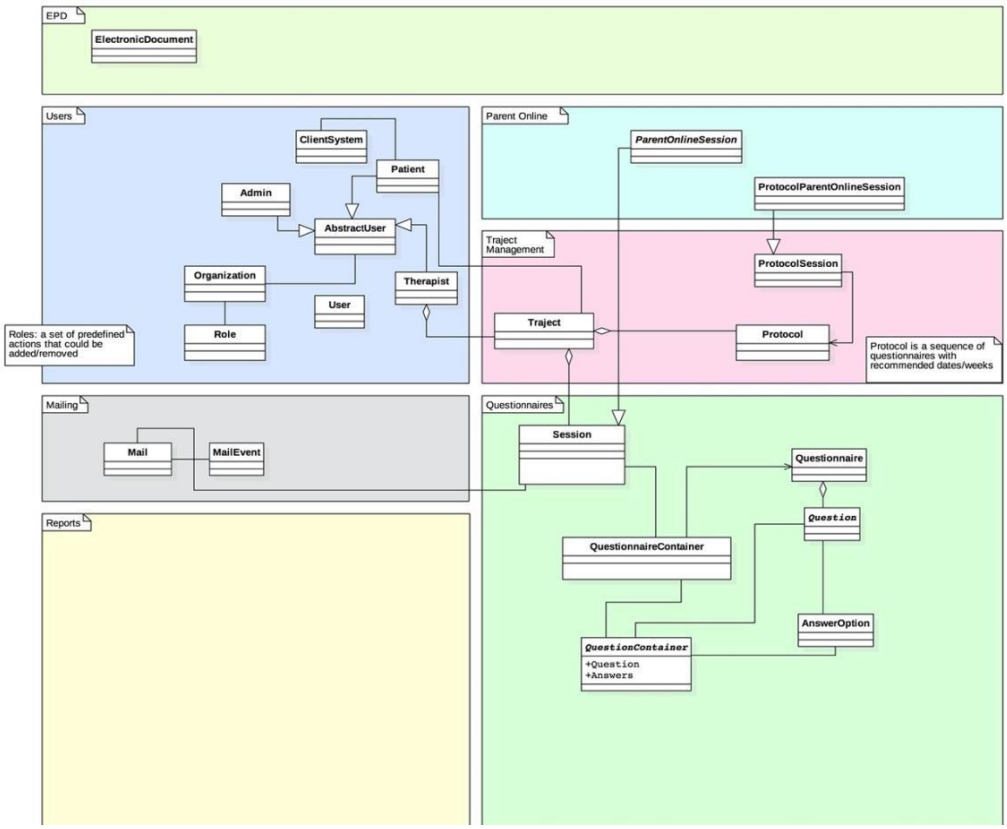


Рис. 3.1. Діаграма класів

Допустимо, користувач вирішив занести дані про нового пацієнта, або подивитися інформацію про пацієнта. Для цього слід нажати на кнопку "Додавання й пошук пацієнта". На екрані відкриється вікно з інформацією про пацієнта, яке презентовано на рис. 3.2.

Рис. 3.2. Інтерфейс "Додавання й пошук пацієнта"

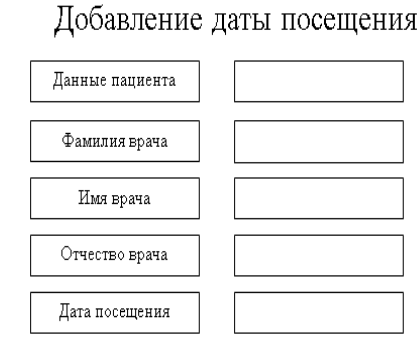


Рис. 3.3. Інтерфейс "Додавання дати відвідування"

Якщо користувачеві необхідно додати історію хвороби пацієнта, або подивитися його історію, слід нажати на кнопку "Додавання причини відвідування". На екрані з'явиться вікно з історією хвороби відповідного пацієнта, представлене на рис. 3.4. У нього вносяться такі дані, як прізвище, ім'я, по батькові пацієнта, причина відвідування, дата відвідування. Після внесення потрібної інформації вона відобразиться в таблиці "Причина відвідування".

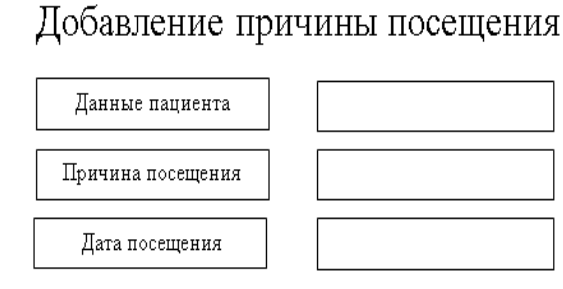


Рис. 3.4. Інтерфейс "Додавання причини відвідування"

Далі слід внести інформації про ліки, які призначив лікар пацієнтові. Для цього слід нажати на кнопку "Додати лікування пацієнта". На екрані відобразиться вікно, представлене на рис. 3.5. Після внесення потрібної інформації вона з'явиться в таблиці "Лікування пацієнта" [23, .с 76].

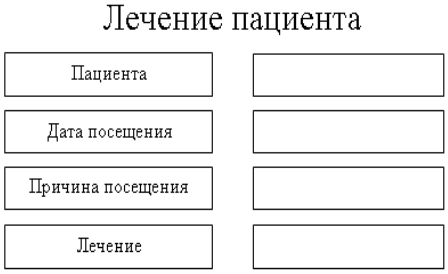


Рис. 3.5. Інтерфейс "Додати лікування пацієнта"

Якщо лікар не може точно визначити, у чому полягає причина хвороби, то пацієнт направляється на здачу аналізів. Для цього лікар- терапевт повинен роздрукувати пацієнтові напрямок на аналіз. Форма для печатки представлено на рис. 3.6.

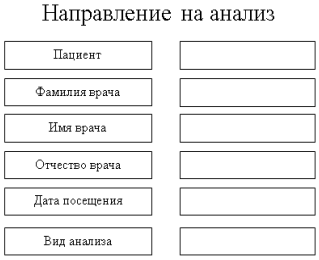


Рис. 3.6. Інтерфейс "Напрямок на аналіз"

Після того, як будуть готові результати аналізи пацієнта, лікар може призначити йому лікування. Результати аналізу будуть заноситися у форму, представлену на рис. 3.7. Дана форма представлена для аналізу крові. Форми для аналізу мочи й калу буде мати такий же вид.

Рис. 3.7. Інтерфейс "Аналіз крові"

У випадку поганого самопочуття пацієнта, появи ускладнень або поганих аналізів, пацієнта слід покласти в палату під спостереження. Результати оглядів будуть заноситися у форму, представлену на рис. 3.8. У примітку буде заноситися інформація про стан пацієнта [30, с. 54].

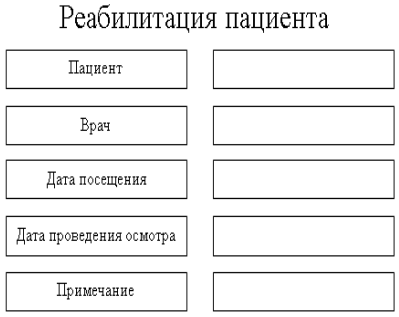


Рис. 3.8. Інтерфейс "Реабілітація пацієнта"

Для більш наочної вистави роботи програми, створимо її блок-схему.

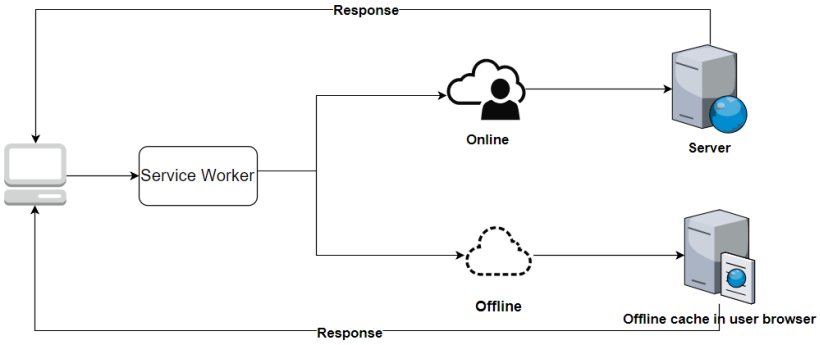


Рис 3.9. Взаємодія елементів верхнього рівня системи

Рис. 3.10. Алгоритм аутентифікації користувача

Робота програми буде вестися згідно із представленою схемою на рис. 3.11.

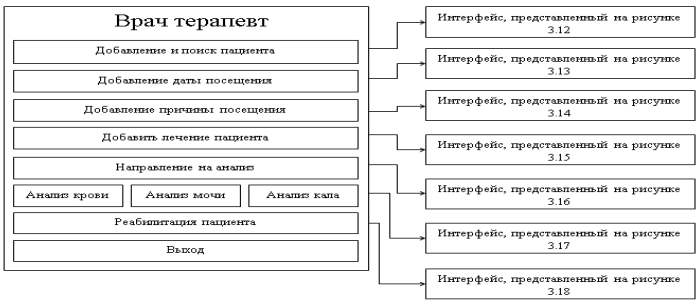


Рис. 3.11. Схема програми

3.2. Алгоритм роботи додатка

У ході процесу розробки програмної частини, був створений інтерфейс, що включає в себе кілька частин:

- Вікно авторизації користувача;
- Вікно роботи з базою пацієнтів (пошук, фільтрація);
- Вікно оформлення пацієнта (визначення статусу диспансеризації, друк облікової форми диспансеризації, оформлення медичного висновку);
- Вікно оформлення діагнозу пацієнта.

1) Вікно авторизації користувача;

Вікно авторизації користувача відповідає за ідентифікацію користувача в інформаційній системі [21 ,с. 76].

У даному модулі користувачеві необхідно коректно ввести дані привласнені йому (логін і пароль), а також після цього нажати на кнопку "Увійти", після чого ІС перевірить правильність уведених користувачем даних.

Після того, як ІС перевірить правильність уведених користувачем даних, то в користувача з'являється два шляхи подальших дій:

1) Інформація введена коректно (поява кнопок "Пацієнти" і "Вихід");
2) Інформація введена некоректно (поява повідомлення: "Помилка: перевірте правильність даних. ").

Право користуватися модулем "Терапевт" мають тільки фахівці медичної установи, яким привласнений певний вид доступу в базі даних.

У вікні авторизації розташовані стандартні компоненти Borland Delphi 7 такі як:

- Label (написи: авторизація, логін, пароль, користувач);
- Edit (форми для введення даних);
- Button (кнопка, відповідальна за перевірку введених користувачем даних).
- Image (компонент, що містить фоновий рис. вікна реєстрації)

2) Вікно роботи з даними пацієнтів (пошук, фільтрація);

Вікно роботи з даними пацієнтів відповідає за фільтрацію й пошук пацієнтів за встановленими критеріями (Додаток №1, Рис. 4). Так само в даному вікні розташовані кнопки "Вихід" і "Облікова форма диспансеризації пацієнта", відповідальні за повне закриття всієї програми й відкриття вікна з докладної інформації про проходження пацієнтом диспансеризації.

Дане вікно розділене на кілька частин [45, с. 80]:

Фільтрація (нижня частина вікна з маркерами для фільтрації даних);

- Відображення даних (верхня частина вікна з таблицею);

Кнопки вибору дій (підзаголовок вікна й нижній правий кут).

У частині фільтрації розташовані стандартні компоненти Borland Delphi 7 такі як:

- Groupbox (компонент, що містить Labelc посадою й ПІБ користувача, що працює в цей момент із IC).

- Label (мітки, відповідальні за назву маркерів фільтрації);

- Edit (графа для введення даних з метою фільтрації й пошуку);

У частині відображення даних розташовані стандартні компоненти Borland Delphi 7, такі як:

- Dbgrid (компонент, відповідальний за висновок даних з таблиці даних, що зберігаються в базі);

У частині кнопки, що містить, вибору дії, розташовані стандартні компоненти Borland Delphi 7, такі як:

- Button (кнопка, що відповідає перехід до докладної інформації користувача)

- Mainmenu (панель, що перебуває під заголовком вікна, що містить у собі кнопку виходу із програми)

3) Вікно оформлення пацієнта;

Вікно оформлення пацієнта відповідає, як за редагування, так і за одержання відомостей про проходження диспансеризації пацієнтом .

Дане вікно містить стандартні компоненти Borland Delphi 7, такі як:

- Label (мітки, відповідальні за назву граф);

- Edit (графи з виведеними в них даними пацієнта);

- Button (кнопка "Друк паспорта здоров'я" і кнопка "Висновок", відповідальна за відкриття шаблону з полем для введення мед. рекомендацій терапевта);

- Combobox (поля для вибору значень);

- Dbchart (діаграма з інформацією про пройдених мед. заходах).

Створення облікової форми диспансеризації пацієнта (Облікова форма № 131/у) проводиться за допомогою обробки програмою шаблону документа Microsoft Office Word. Після натискання на кнопку "Друк облікової форми диспансеризації", розташованої в правій частині вікна пацієнта, ИС відкриває каталог документів з переліком доступних шаблонів документів, розташованим у папці із програмою, у якому користувачеві необхідно вибрати підходящу форму документа. Після процедури вибору шаблону документа, ІС здійснить відкриття обраного шаблону й автоматичне заповнення граф даними про підсумки проходження диспансеризації обраним пацієнтом [10, .с 65].

Робота ІС із шаблоном документа Msword здійснюється за допомогою стандартного компонента Borlanddelphi 7 - Opendialog, який по заданому в коду шляху, відкриває каталог з потрібними документами й після вибору потрібного документа, заповнює його даними.

Зміст програмного модуля.

Програмний модуль ІС складається з одного файлу формату". exe" і папки з файлами - doc (містить шаблони документів) і images (містить іконки, фони вікон програми й логотипи).

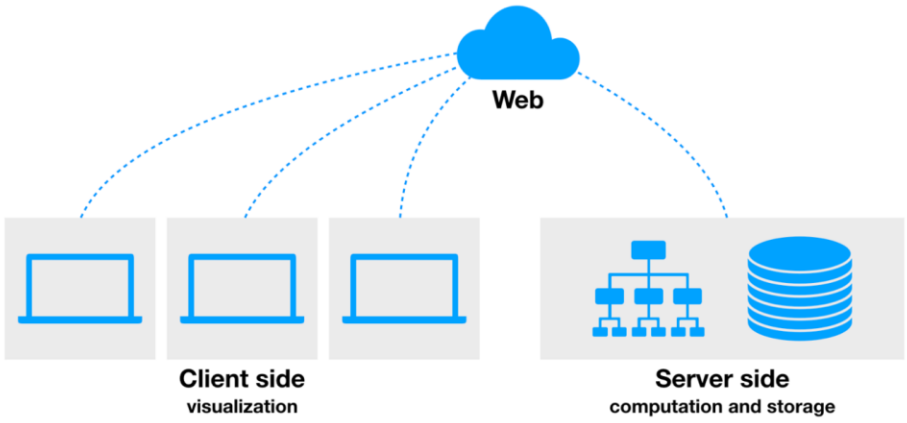


Рис. 3.12. Загальна структура додатку

Реалізація програми на першому витку спирали являє собою звичайні таблицю з елементами керування самого середовища розробки MS Access. Тут немає ніякого інтерфейсу, який міг би полегшити роботу користувача із програмою.

Основним ризиком на даному етапі розробки програми є виникнення проблем з електроживленням. При виникненні даної проблеми можливий збій роботи програми. Це може привести до відмови роботи бази даних, а так само не виключається можливість втрати яких-небудь даних. Для розв'язку даної проблеми потрібно мати додаткове джерело електроживлення/електрики.

Після того, як усі потрібні таблиці були створені, ми створюємо схему й зв'язку всіх використовуваних таблиць даних, необхідних для додатка. Схема даних, представлена на рис. 3.12, будувалася шляхом додавання таблиць даних, виділенням з них ключових полів, необхідних для створення зв'язків між ними. Усе це виконувалося в режимі "конструктора" [8, с. 37].

На даному етапі реалізовується можливість перегляду даних, а також можливість зміни, видалення, внесення й пошуку інформації про пацієнта, а також для печатки. Для чого створюється "Форма" - об'єкт, за допомогою якого користувачі можуть додавати, редагувати й відображати дані.

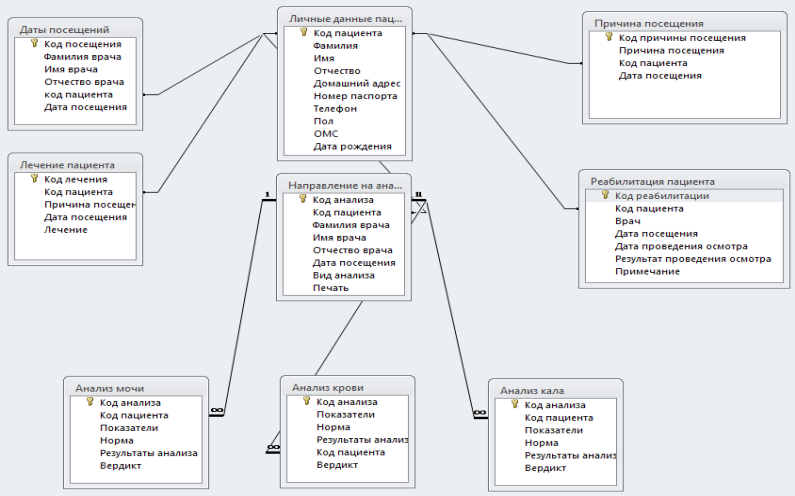


Рис. 3.13.. Схема даних

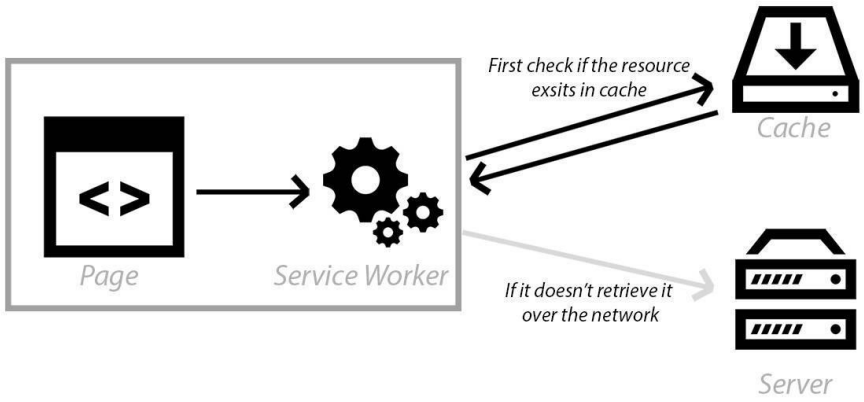


Рис. 3.14. Схема кешування

Основним ризиком на даному етапі розробки програми є людський фактор-неуважність лікаря при внесенні даних пацієнта. З боку користувача ПЗ - лікаря необхідно бути більш уважним при роботі з персональними даними пацієнтів.

3.3. Проектування бази данних

При розробці модуля був обраний канонічний підхід до проектування ІС, як оптимальний для розробки програмної частини, як окремого продукту. Даний вид підходу до проектування відповідає стандарту ДЕРЖСТАНДАРТ 34.601-90

"Інформаційна технологія. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення".

Відповідно до ДЕРЖСТАНДАРТ 34.601-90 розробка була розділена на стадії (таблиця 3.1.).

Таблиця 3.1.

Стадії й етапи створення ІС.

Стадії	Етапи робіт
1. Формування вимог до ІС	1.1 Обстеження об'єкта й обґрунтування необхідності створення ІС; (проведене спілкування з реєстраторами, вивчення відомчої документації (накази про проведення диспансеризації і т.д.). 1.2 Формування вимог користувача до ІС; (уявлення моделей бізнес- процесів (опис)
2. Технічне завдання	2.1 Розробка й твердження технічного завдання на створення ІС;
3. Технічний проект	3.1 Розробка ІС (модуль "Терапевт");
4. Запровадження в дію	4.1 Проведення попередніх випробувань;

Формування вимог:

Обстеження об'єкта й обґрунтування необхідності створення ІС;

На етапі обстеження об'єкта автоматизації було проведено:

- Спілкування з мед. персоналом;

- Вивчення посадових регламентів;

- Аналіз комп'ютерної техніки;

У результаті спілкування з фахівцями були вивчені основні робочі обов'язки мед. персоналу під час проведення диспансеризації, у які входить:

- первинний прийом пацієнтів;

- заповнення облікових форм диспансеризації пацієнтів;

- своєчасний добір і доставка медичної документації в кабінети лікарів, правильне ведення й зберігання картотеки;

- інформування терапевта про необхідність проходження диспансеризації пацієнтом [29, с. 80];

- забезпечення своєчасної адаптації поліклініки під нові нормативно-правові акти, що стосуються диспансеризації.

На даному етапі проектування модуля "Терапевт" був створений ряд моделей бізнес- процесів у вигляді діаграм процесів за допомогою нотації BPMN, які повною мірою відбивають послідовність дій у кожному процесі розроблювального модуля.

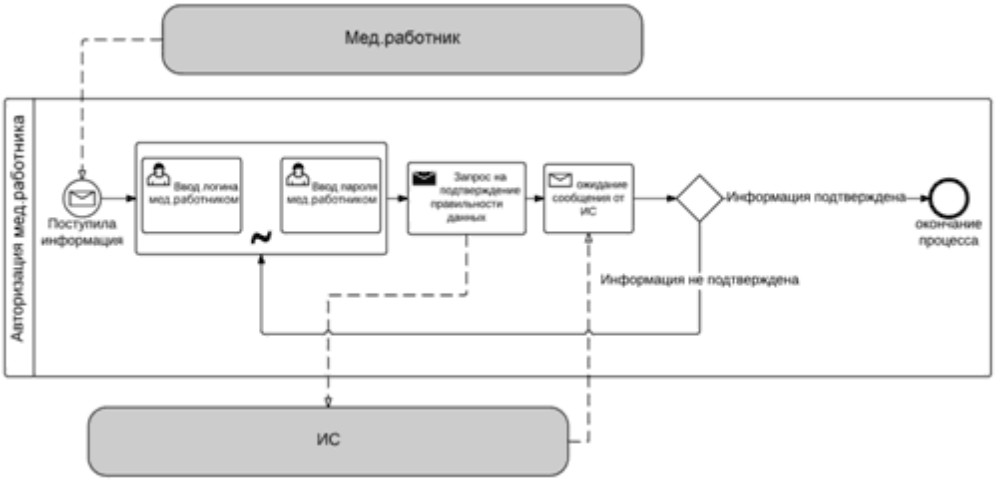


Рис. 3.15.. Мед. працівник. Авторизація

На даній діаграмі відображений бізнес- процес авторизації мед. працівника (у тому числі й терапевта) у програмній частині розроблювального модуля.

Першим етапом у процесі авторизації є введення логіна й пароля користувача, які має кожний мед. працівник медичної установи. Після введення логіна й пароля в ИС направляється запит на перевірку правильності даних співробітника, після чого існує два шляхи розвитку бізнес- процесу, а саме:

1. Дані мед. працівника підтверджені;
2. Дані мед. працівника не підтверджені;

Якщо введена інформація виявилася вірної, то даний процес завершується, і співробітник може продовжувати роботу із програмою. Якщо ж уведена інформація виявилася не правильної, то ІС повертає мед. працівника на етап уведення логіна й пароля, де він повинен спробувати знову ввести дані для авторизації [19, с. 54].

2) Первинний прийом пацієнта лікарем- терапевтом;

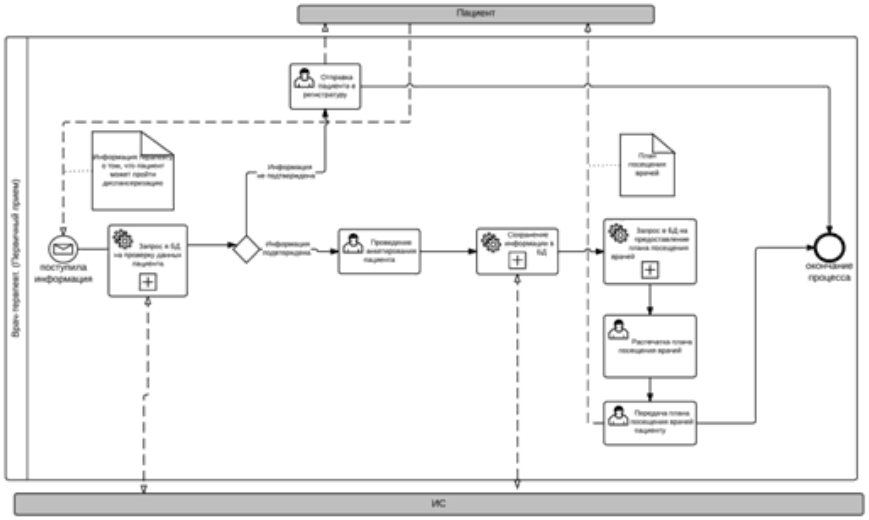


Рис. 3.16. Лікар- Терапевт. Первинний прийом пацієнта

На даній діаграмі відображений бізнес- процес первинного прийом пацієнта терапевтом із залученням модуля "Терапевт".

Першим етапом у процесі прийом пацієнта стає одержання терапевтом інформації про те, що пацієнт, може пройти диспансеризацію. Після цього відбувається запит у БД на перевірку правильності інформації повідомленої пацієнтом, після чого існує два шляхи розвитку бізнес- процесу:

1. Пацієнт може пройти диспансеризацію;
2. Пацієнт не може пройти диспансеризацію [23, с. 43];

Якщо виявилося, що пацієнт не може пройти диспансеризацію, то терапевт відправляє його в реєстратуру. Якщо ж він може пройти диспансеризацію, то терапевт переходить до анкетування пацієнта. Після занесення даних анкетування, терапевт відправляє в БД запит на одержання плану відвідування

лікарів пацієнтом, далі роздруковує й передає його пацієнтові. На цьому первинний прийом закінчується.

3) Вторинний прийом пацієнта лікарем- терапевтом;

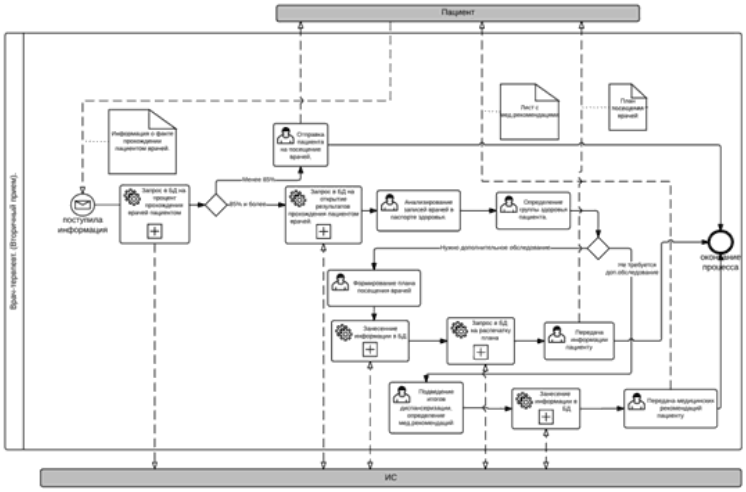


Рис. 3.17. Лікар- Терапевт. Вторинний прийом.

На даній моделі бізнес- процесів відображене вторинний прийом пацієнта терапевтом.

Першим етапом у представленій моделі бізнес- процесу є повідомлення пацієнтом інформації про те, що він пройшов усіх необхідних лікарів. Задіючи інформаційну систему, лікар- терапевт одержує з бази даних інформацію про відсоток завершення диспансеризації пацієнтом. Після цього існують два шляхи розвитку бізнес- процесу:

1. Якщо пацієнт пройшов достатню кількість необхідних мед. заходів.
2. Якщо пацієнт не пройшов достатня кількість необхідних мед. Заходів [19, с. 80];

Якщо пацієнт не пройшов досить мед. заходів, то він відправляється проходити те, що не пройшов. Якщо ж він пройшов досить мед. заходів, то терапевт відправляє в БД запит на одержання результатів проходження їх пацієнтом. Далі лікар аналізує записи, визначає групу здоров'я пацієнта, після чого знову ж існує два шляхи розвитку бізнес- процесів:

1. Додаткове обстеження потрібно;

2. Додаткове обстеження не потрібно;

У випадку якщо додаткове обстеження потрібно пацієнтові, терапевт знову формує план відвідування лікарів, затягає його в базу даних і відправляє на печатку, після передавши план пацієнтові. На це бізнес- процеси завершуються.

У випадку ж, якщо додаткове обстеження не потрібно (таке також можливо, якщо людей уже додатково обстежилася), те лікар- терапевт підводить підсумки диспансеризації, визначає медичні рекомендації, затягає свій висновок у БД, передає мед. рекомендації пацієнтові. На це бізнес- процеси завершуються.

4) Прийом пацієнта медичним працівником;

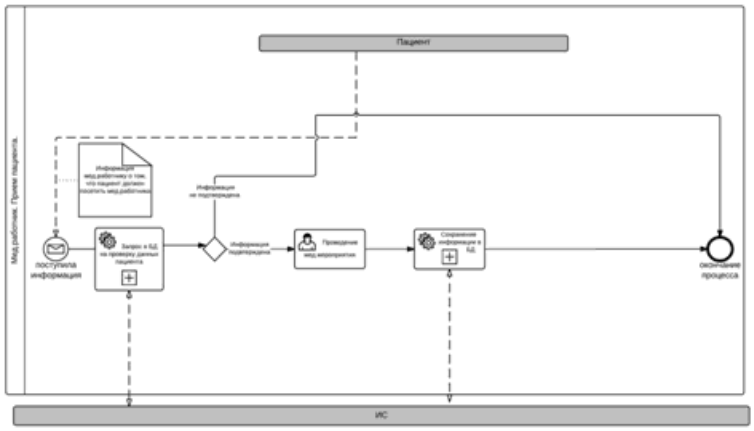


Рис. 3.18. Мед. працівник. Прийом пацієнта.

Дана модель описує бізнес- процес прийом мед. працівником пацієнта.

Першим етапом у представленій моделі бізнес- процесу, є повідомлення пацієнтом інформації про те, що терапевт направив його до цього медичного фахівця [20 ,с. 65].

Далі мед. працівник відправляє в БД запит на перевірку даних пацієнта. Після цього існують два шляхи розвитку бізнес- процесу:

Дані виявилися вірні;

Дані виявилися неправильні;

У випадку, дані пацієнта виявилися неправильними, то він відправляється в реєстратуру, а бізнес- процес закінчується. Якщо ж виявилися вірні, то

фахівцем проводиться медичний захід, а його підсумки зберігаються в базу даних. На це бізнес- процес закінчується.

5) Занесення даних у БД;

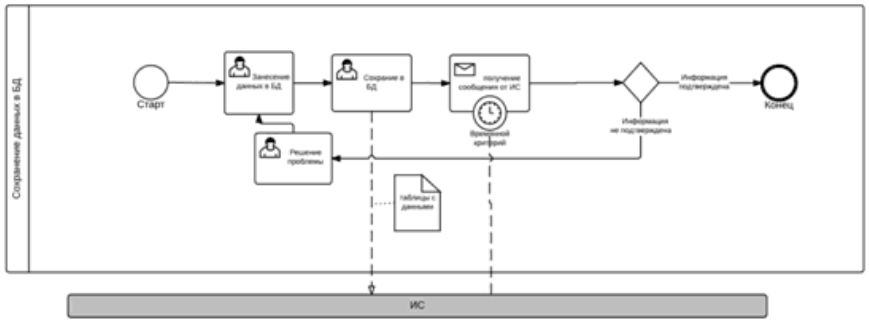


Рис. 3.19. Занесення даних у БД;

Дана модель описує бізнес- процес занесення даних у БД.

Першим етапом у представленій моделі бізнес- процесу, є занесення даних у необхідні таблиці, після чого користувач натискає кнопку збереження, а введені їм дані зберігаються в БД. Опираючись на часовий критерій, користувач очікує відповіді від БД. Після цього існують два шляхи розвитку бізнес- процесу:

1. Збереження даних пройшло успішно;
2. Зберегти дані не вдалося;

У випадку якщо від ИС зробила позитивна відповідь про збереження даних у БД, бізнес- процес завершується. А якщо ні, то, користувачеві необхідно розв'язати проблему перешкоджаючу збереженню даних, після чого повторити бізнес- процес спочатку [17, с. 87].

6) Запит у БД;

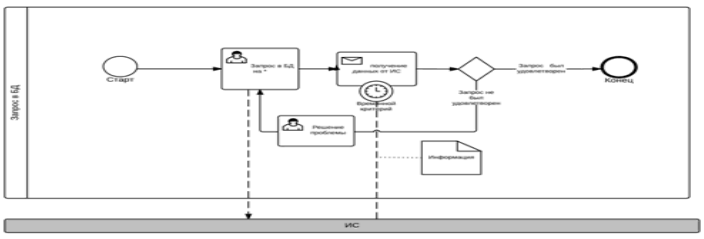


Рис. 3.20. Запит у БД

Дана модель описує бізнес- процес відправлення запиту в БД і одержання відповіді від неї.

Першим етапом у представленій моделі бізнес- процесу, є відправлення в ИС необхідного запиту. Запит може кожним, єдиний критерій - він не повинен перевищувати функціональні можливості ИС. Потім, опираючись на часовий критерій, користувач очікує одержання даних від ИС. Після цього існують два шляхи розвитку бізнес- процесу [24, с. 48]:

Запит був удоволений;

Запит не був удоволений;

У випадку, якщо запит користувача вдоволений не був, він усуває проблему, що перешкодила йому одержати необхідну інформацію, після чого повторює свій запит. У випадку, якщо запит був удоволений, бізнес- процес завершується.

7) Запит до пацієнта;

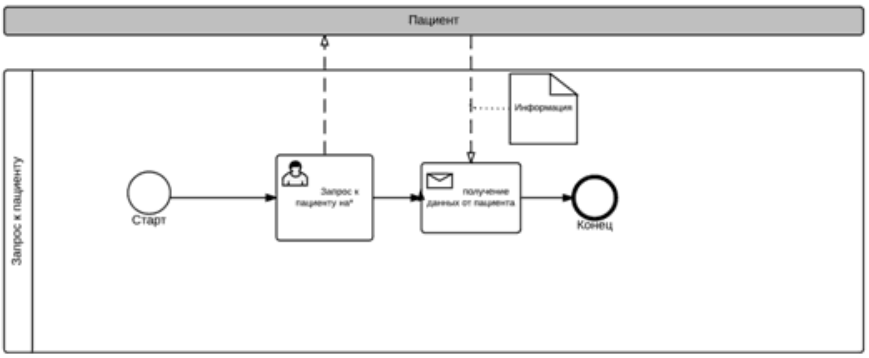


Рис. 3.21. Запит до пацієнта

Дана модель описує бізнес- процес відправлення запиту до пацієнта й одержання відповіді від нього.

У представленій моделі бізнес- процесу, медичним фахівцем відправляється запит до пацієнта на одержання необхідної інформації, або виконання якої-небудь дії, після чого очікує він відповіді. Незалежно від того, у

якому ступені був удоволений запит, бізнес- процес завершується. При необхідності він може бути повторений медичним фахівцем.

Таблиця 3.3.

Опис типів полів БД

Найменування таблиці	Найменування стовпця	Тип поля
Таблиця "Маршрутна карта" містить дані з результатами приймань пацієнтів у кожного з фахівців медичної установи.	id_pz	INT
	Опитування терапевтом	TEXT
	Вимір ПЕКЛО	TEXT
	ЭКГ	TEXT
	Флюорографія легенів	TEXT
	ЭФГДС	TEXT
	Огляд офтальмолога	TEXT
	Огляд невролога	TEXT
	Огляд лікаря хірурга	TEXT
	УЗИ черевної порожнини	TEXT
	Other medical events (20)	TEXT
Таблиця "Пацієнти" містить особисті дані пацієнтів, що пройшли процедуру реєстрації в медичній установі.	id_pz	INT
	Прізвище	VARCHAR

	Ім'я	VARCHAR
	По батькові	VARCHAR
	Підлога	VARCHAR
	Дата_народження	VARCHAR
	СНИЛС	VARCHAR
	Поліс_ОМС	VARCHAR
	Адреса_фактичний	VARCHAR
	Телефон	VARCHAR
	Other personal information (10)	VARCHAR
Таблиця "Діагнози" містить дані з діагнозами, призначеними пацієнтам, а також дату постанови діагнозу, дані лікаря, що поставив діагноз і оцінку показників здоров'я пацієнта.	id	INT
	id_pz	INT
	Медмероприятие	CHAR
	Діагноз	VARCHAR
	Дата	DATE
	Лікар	VARCHAR
	Оценкапоказателей	VARCHAR
Таблиця "Медперсонал" містить дані про персонал медичної установи.	id_sotr	INT

	id_dolzj	INT
	id_bol	INT
	ПІБ	VARCHAR
Таблиця "Користувачі ИС" містить дані користувачів для роботи з ИС.	id_sotr	INT
	ПІБ	VARCHAR
	login	CHAR
	password	CHAR
	dostup	CHAR
	Должность	VARCHAR
Таблиця "Мед. Установа" містить дані про медичні установи.	id_adr	INT
	id_pz	INT
	Адреса	VARCHAR
Таблиця "Посади" містить дані про посади співробітників медичної установи	Найменування	CHAR
	id_sotr	INT
	id_medmer	INT
Таблиця " Медмероприятия-Віку" містить дані з віками пацієнтів	id_medmer	INT
	Медмерпориятие	CHAR
	Список віків	INT

Таблиця "Організації" містить дані з місцями роботи пацієнтів	Id_org	INT
	Id_pz	INT
	Місце роботи	CHAR

3.4. Реалізація інтерфейсу користувача

Наступним кроком було потрібно захистити дані пацієнтів. Для цього випливало поставити пароль на базу даних. Основним ризиком на даному етапі розробки програми є введення занадто простого пароля, а також недбалого відносини з боку користувачів за дотриманням умов роботи в ПЗ. Це може привести до входу в базу даних сторонніх осіб. Для розв'язку даної проблеми потрібно ретельно підходити уводити, увести до ладу вибору пароля, а лікареві-користувачеві уважно підходити уводити, увести до ладу зберіганню й використанню службової інформації (паролі, коди) і персональних даних пацієнтів [35, с. 64].

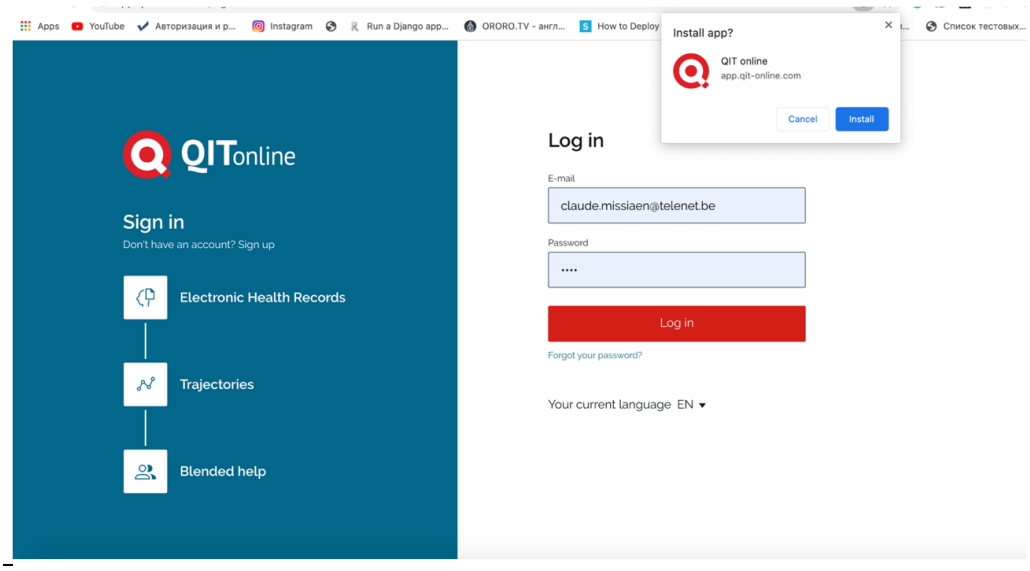


Рис. 3.22. Вікно для введення пароля

У випадку якщо пароль буде введений неправильно, те програма не дасть користувачеві можливість увійти в базу даних і виведе на екран вікно оповіщення, представлене на рис. 4.23.

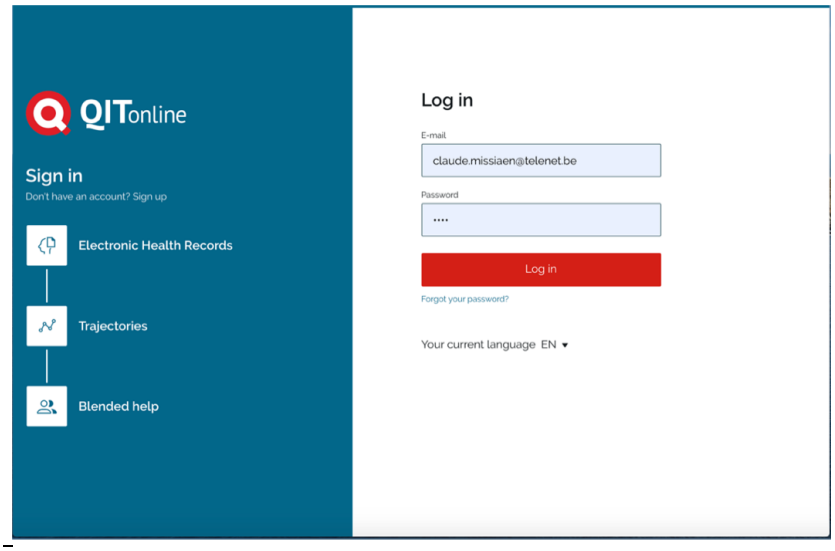


Рис. 3.23. Вікно, що сповіщає про неправильне введення пароля

Так само слід створити інтерфейс для прискорення прийом пацієнтів і спрощення роботи користувача (рис. 3.23.) [23, с. 76].

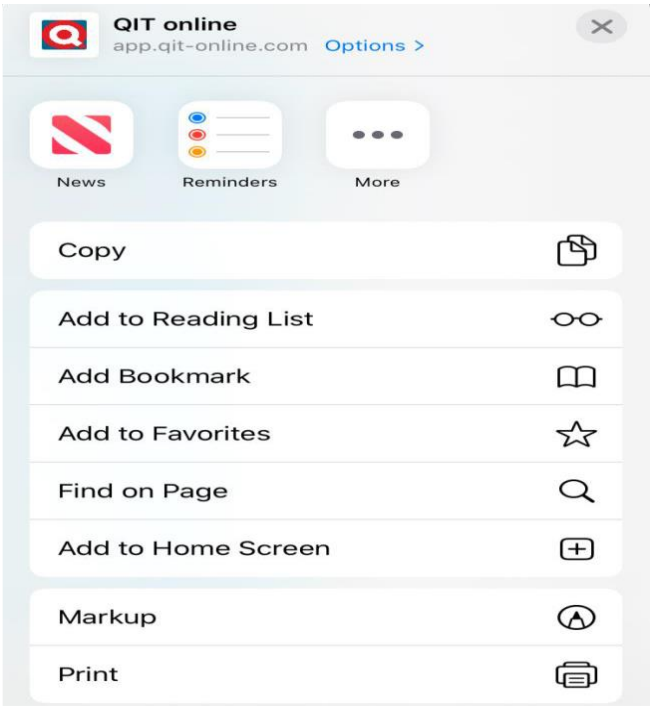


Рис. 3.24. Головне меню додатка

Перша кнопка - "Додавання й пошук пацієнта", при натисканні на цю кнопку відбувається перехід у форму з інформацією про пацієнтів. Форма представлено на рис. 3.24.. У ній можна зробити пошук інформації про потрібний нам пацієнтові, додати нового при натисканні на кнопку "Додати пацієнта", вилучити пацієнта, або відредагувати дані, які вже існують у цій формі.

Друга кнопка - "Додавання дати відвідування". При натисканні на неї відбувається перехід у форму з датами відвідувань пацієнтів. У ній можна подивитися дати відвідувань пацієнтів і додати нові. Дана форма представлено на рис. 3.24.

Третя кнопка - "Додавання причини відвідування". При натисканні на неї відбувається перехід у форму із причинами відвідувань усіх пацієнтів. У ній можна подивитися історію хвороб пацієнтів, а також додати нові. Дана форма представлено на рис. 3.25. [41, .с 87].

Четверта кнопка - "Додати лікування пацієнта". При натисканні на неї відбувається перехід у форму з усією інформацією про історію лікування пацієнтів. У ній можна подивитися історію лікування пацієнта, а також додати нове, якщо буде потреба. Дана форма представлено на рис. 3.25..

П'ята кнопка - "Напрямок на аналіз". У цій формі можна зробити напрямок пацієнтові на який-небудь аналіз, роздрукувати його й поставити друк. Дана форма представлено на рис. 3.25..

Шоста - восьма кнопки - "Аналіз крові/сечі/калу". При натисканні на ці кнопки буде відбуватися перехід у форми результатами аналізів пацієнтів. У ній можна подивитися інформацію про результати аналізів для ухвалення рішення про те, чим робити лікування пацієнта.

Дев'ята кнопка - "Реабілітація пацієнта". При натисканні на цю кнопку буде відбуватися перехід у форму, що містить інформація про те, як відбуватися реабілітація пацієнта. У ній можна подивитися як відбувається реабілітація пацієнта. У випадку ускладнення/погіршення стану призначити нові ліки. Дана

форма представлено на рис. 3.25.. Десята кнопка - "Вихід". При натисканні на неї відбувається виходу з додатка.

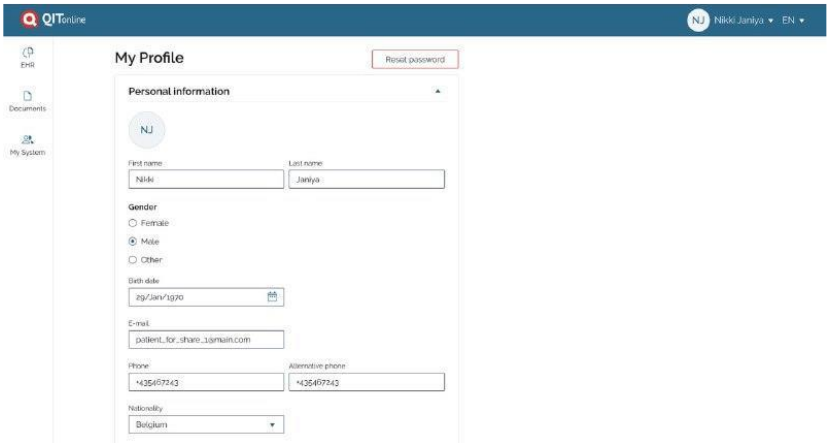


Рис. 3.25. Інтерфейс особистого кабінету терапевта

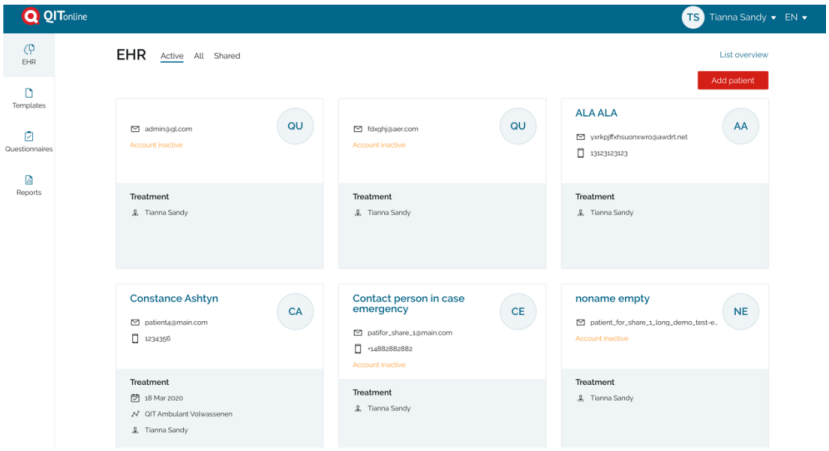


Рис. 3.26. Інтефейс терапевта

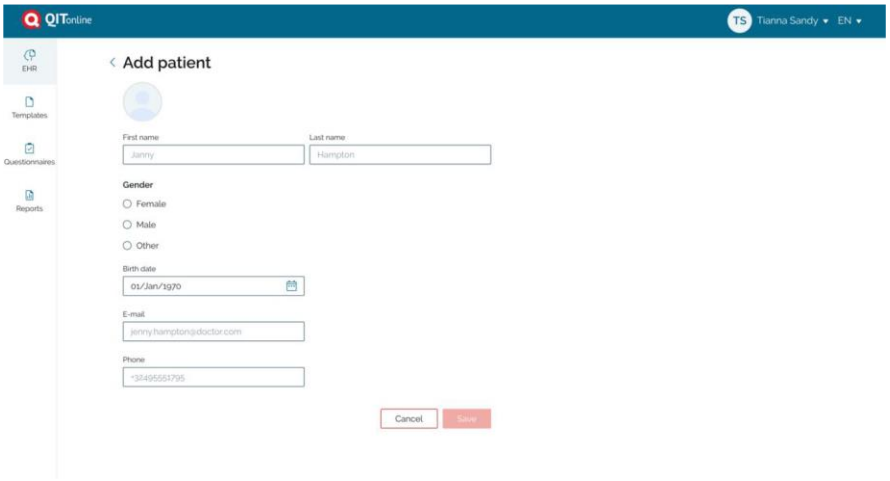


Рис. 3.27. Процес створення облікового запису пацієнта

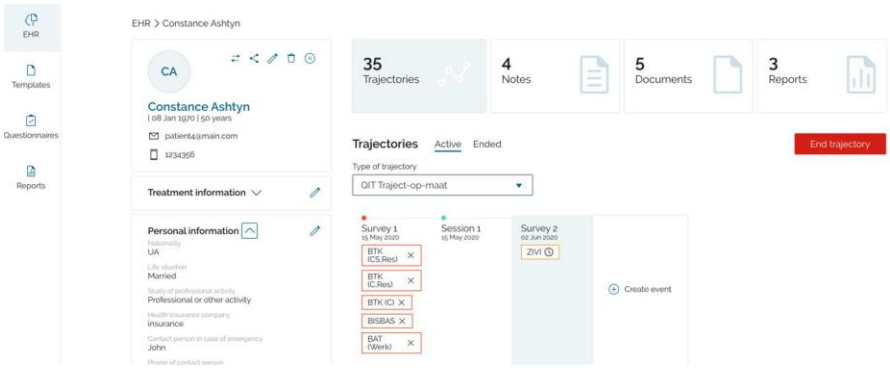


Рис. 3.28. Сторінка взаємодії з пацієнтом

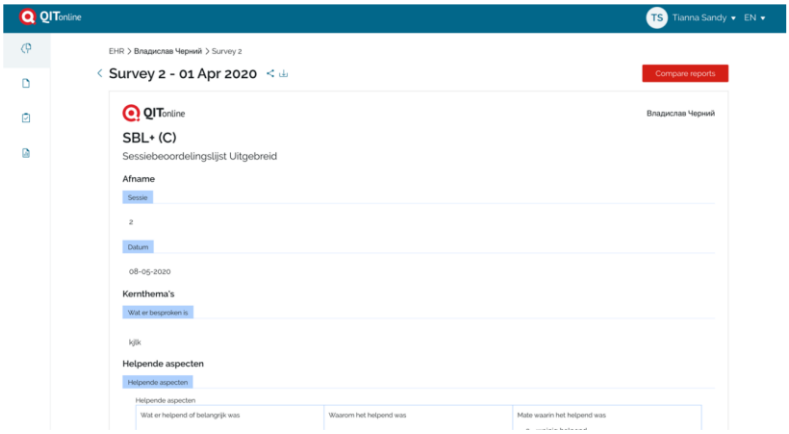


Рис. 3.29. Сторінка репорту

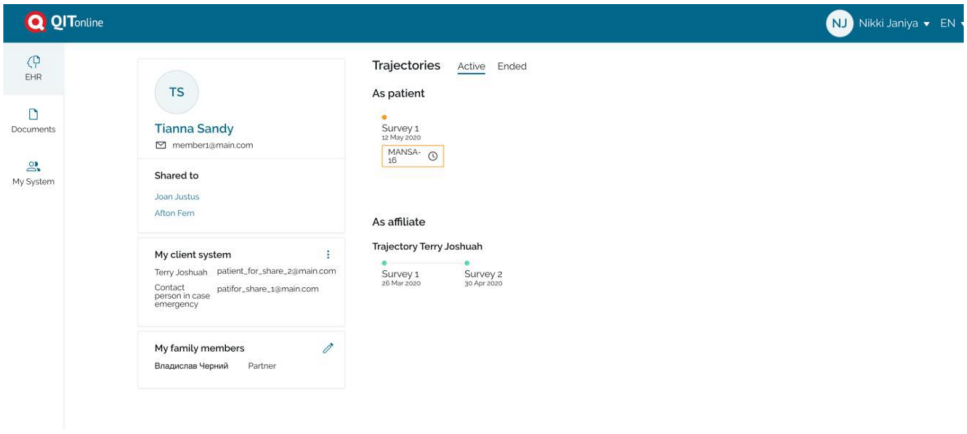


Рис. 3.30. Інтерфейс пацієнта

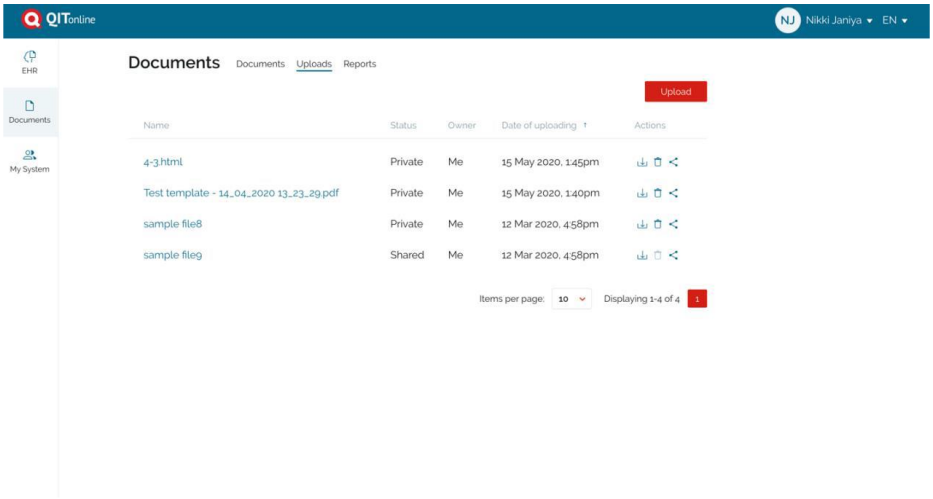


Рис. 3.31. Вкладка “Документи”

Для проектування системи була використана діаграма класів, яка включала ключові компоненти системи, такі як лікар терапевт, пацієнт, історія медичного обстеження, симптоми, рецепти, база даних, інтерфейс користувача та система прийняття рішень. Це забезпечило структурованість системи та розподіл функціональності між різними класами.

Основні функції системи включали зберігання та управління медичною інформацією, вибір симптомів та визначення діагнозу, а також виписування рецептів для пацієнтів. Інтерфейс користувача забезпечував зручну взаємодію лікаря з системою шляхом відображення інформації та отримання вибору від користувача [29, .с 54].

Для реалізації системи було використано підходи та методи проектування програмного забезпечення, зокрема принципи SOLID, які сприяли гнучкості та розширюваності системи. Також було важливо врахувати особливості домену медицини та забезпечити безпеку та конфіденційність медичних даних.

Реалізація інформаційної системи підтримки прийняття рішень лікаря терапевта може значно полегшити його роботу та покращити якість надання медичних послуг. Система може сприяти швидкому та точному визначенню

діагнозів, ефективному лікуванню пацієнтів та забезпеченню збереження та доступу до медичної інформації

Проте, варто зазначити, що проектування та реалізація інформаційної системи підтримки прийняття рішень лікаря терапевта є складним процесом, який вимагає глибокого розуміння домену медицини, уважного аналізу вимог та потреб користувачів, а також дотримання вимог безпеки та конфіденційності.

В цілому, розробка інформаційної системи підтримки прийняття рішень для лікаря терапевта може принести значну користь в медичній галузі, полегшивши роботу лікарів та покращивши результати лікування пацієнтів. Проектування та реалізація такої системи вимагає систематичного та комплексного підходу, але може привести до значних переваг для медичного співтовариства та пацієнтів.

3.5. Тестування та налагодження програми

Тестування програмного забезпечення - процес аналізу програмного продукту й супутньої документації з метою виявлення недоліків у роботі й підвищення його якості. Звичайно для перевірки працездатності розробленої програми проводять три види тестування: функціональне, інтеграційне й навантажувальне.

Функціональне тестування (Functional testing) - вид тестування, спрямований на перевірку коректності роботи функціональності додатка (коректність реалізації функціональних вимог). Опис розроблених модулів програми, відповідних до вимог. Функціональне тестування даних модулів показало їхню повну працездатність і відсутність помилок, тобто всі функціональні вимоги були успішно реалізовані [11, с. 80].

Інтеграційне тестування (integration testing) спрямоване на перевірку взаємодії між декількома частинами додатка. Дане тестування проводиться у

випадку, коли відбувається взаємодія модулів програми (II виток спирали). Дана перевірка виявилася успішною. Зокрема , у процесі запису пацієнта на прийом відбувається додавання в таблицю "Особисті дані пацієнтів" даних, уведених у форму для додавання пацієнта Даний процес виконується без збоїв у роботі програми. Аналогічно можна розглянути процес додавання причини відвідування пацієнта. У процесі додавання причини відвідування для пацієнта за допомогою потрібної форми запис, що відповідає, з'являється в таблиці "Причини відвідування". Даний процес також працює без збоїв.

Навантажувальне тестування (load testing, capacity testing) - дослідження здатності додатка зберігати задані показники якості при навантаженні в припустимих межах і деякому перевищенні цих меж (визначення "запасу міцності") [14, с. 54].

Навантажувальне тестування проводилося для дослідження швидкодії таких процесів, як, наприклад, запис пацієнта на прийом (оскільки цей процес покликано зменшити час очікування пацієнтів у чергах, то передбачається, що його виконання не повинне займати багато часу) і пошук записи. Для тестування була обрана наступна кількість записів: 5 (у середньому відвідувачів за годину роботи), 50 (у середньому відвідувачів за день) і 500 (у середньому відвідувачів на 2 тижні). Результати навантажувального тестування наведені в табл.6.

Спочатку 5 раз проводиться перевірка відгуку системи на різні дії, її результати вносяться у відповідні рядки стовпців t1-t5. Далі розраховується середнє арифметичне всіх вимірів по формулі (1):

$$t_{\text{ср.ариф.}} = \frac{\sum_{i=1}^{n} t_i}{n}$$

Результат заноситься в стовпець "Середній час відгуку" таблиці. У наступному стовпці перебувають відповідні середні квадратические відхилення, розраховані по формулі (2):

$$\sigma = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} \left(t_i - t_{\text{ср.ариф.}}\right)^2}$$

Погрішність вимірів розраховувалася по формулі (3) і заносилася в стовпець "Погрішність вимірів", де n - число вимірів, - довірчий коефіцієнт Стьюдента, рівний 0.95, - абсолютна погрішність електронного секундоміра рівна 0,005.

$$\Delta t = \sqrt{\left(\frac{\sigma}{n} \cdot t_{\alpha(N-1)}\right)^2 + \Delta t_p^{\,2}}$$

У стовпці "Час відгуку" - підсумковий час відгуку. Воно розраховувалося по формулі (4):

$$t_{\text{откл}} = t_{\text{ср.ариф.}} \pm \Delta t$$

Таблиця 3.5.

Результати навантажувального тестування

Кількість записів	Дія	t1, с.	t2, с.	t3, с.	t4, с.	t5, с.	Середній час відгуку, с.	Середн. квадр. откл., с.	Погрішність вимірів, с.	Час відгуку, с.
5	Додавання	0,21	0,23	0,21	0,23	0,2	0,216	0,0235	0,0254	0,216±0,025
	Пошук	0,2	0,22	0,19	0,21	0,2	0,206	0,0225	0,0243	0,206±0,024
50	Додавання	0,3	0,29	0,31	0,31	0,3	0,302	0,0341	0,0369	0,302±0,037

	Пошук	0,27	0,26	0,28	0,29	0,3	0,28	0,0339	0,0358	0,28±0,036
500	Додавання	0,39	0,38	0,39	0,37	0,4	0,385	0,0376	0,0394	0,385±0,04
	Пошук	0,36	0,35	0,36	0,34	0,4	0,362	0,0369	0,0389	0,362±0,039

Як видне з табл. 3.5., час відгуку не перевищує 0,5 з і майже не зауважується користувачем. Таким чином, можна зробити висновок, що навантажувальне тестування проведене успішно.

Метою даної роботи було створення автоматизованого робочого місця для лікаря-терапевта. Необхідно було оптимізувати робочий процес із метою економії часу співробітника й зменшення часу прийом пацієнтів. Таким чином, у рамках даної роботи були виконання наступні дії:

- У процесі роботи були проаналізовані користувацькі вимоги (19 вимог), отримані шляхом опитування й вивчення документації [25, .с 61].

- Були визначені пріоритети отриманих вимог.

- На основі користувацьких вимог були складені функціональні вимоги, які повинні бути реалізовані в розроблювальному додатку.

- Складений список вимог, що зв'язує разом користувацькі, функціональні вимоги, їх пріоритет, а також програмні компоненти, відповідальні за реалізацію даних вимог;

- Після формування списку вимог відбувся розподіл усіх вимог на 1-3 рівні спирали.

- Після того, як вимоги були проаналізована, проводилося складання моделей AS- IS і TO-BE ключових бізнес- процесів організації в нотаціях ARIS VACD (0-1 рівні) і UML Activity Diagram (2-3 рівень), і проведене їхнє

зіставлення на кожному рівні. Це дозволило визначити структуру розроблювального додатка [13, с. 85].

Розробка додатка проводилася в середовищі MS Access, де поетапно були реалізовані всі зазначені вимоги, виправлені недоліки й помилки в роботі програми. Ця програма є простий і зрозумілої в обігу; є повністю сумісною із системою Windows і має величезні можливості по імпорту й експорту даних.

Після кожного етапу (витка спирали) розробки проводилося функціональне тестування розроблених модулів програми, що показало їхня працездатність і відповідність заявленим вимогам.

При завершенні заключного етапу розробки проведене інтеграційне тестування для перевірки взаємодії програмних модулів, що також показало їх повну працездатність.

Потім було проведено навантажувальне тестування при різній кількості записів у БД, що свідчить про високу продуктивність програми.

ВИСНОВКИ

Таким чином, дослідження показало, що інформаційні системи підтримки прийняття рішень мають великий потенціал для поліпшення медичної практики та підвищення якості надання медичної допомоги.

Переваги використання таких систем включають доступ до актуальних медичних даних, клінічних протоколів та експертних рекомендацій, забезпечення систематизації та організації медичної інформації, аналіз даних та надання рекомендацій щодо оптимального лікування. Ці системи можуть допомагати лікарям терапевтам впроваджувати стандартизовані підходи до діагностики та лікування, сприяти прийняттю обґрунтованих рішень та зменшувати ризики помилок.

Однак, важливо зазначити, що інформаційні системи підтримки прийняття рішень не можуть замінити професійний досвід та експертизу лікарів. Вони слугують інструментом, що сприяє ухваленню обґрунтованих рішень, але кінцеве рішення має бути зроблене лікарем, враховуючи унікальні особливості кожного пацієнта.

Загалом, впровадження інформаційних систем підтримки прийняття рішень в медичну практику може принести значну користь. Вони сприяють збільшенню ефективності лікарської роботи, покращенню якості надання медичної допомоги та зменшенню ризиків помилок. Проте, необхідно забезпечити надійний захист медичної інформації та враховувати індивідуальні потреби та професійний досвід лікарів під час розробки та використання таких систем.

У майбутньому, подальші дослідження в цій області можуть спрямовуватись на розробку інноваційних інформаційних систем, які враховуватимуть реальний контекст медичної практики та індивідуальні потреби

лікарів терапевтів. Такі системи можуть стати важливим інструментом підтримки лікарських рішень та сприяти подальшому покращенню медичної допомоги.

Крім того, розвиток технологій штучного інтелекту та машинного навчання відкриває нові перспективи для інформаційних систем підтримки прийняття рішень у медицині. Вони можуть використовувати алгоритми машинного навчання для аналізу великих обсягів даних, виявлення кореляцій та патернів, а також прогнозування результатів лікування. Це може допомогти лікарям терапевтам у виявленні ранніх ознак захворювань, визначенні оптимальних терапевтичних режимів та прогнозуванні результатів лікування з вищою точністю.

Однак, важливо забезпечити етичність та надійність використання таких систем. Збір та аналіз медичних даних має відповідати нормам конфіденційності та захисту приватності пацієнтів. Також, необхідно враховувати можливі обмеження та помилки алгоритмів машинного навчання, що можуть вплинути на точність рекомендацій системи.

Загалом, інформаційні системи підтримки прийняття рішень є важливим елементом сучасної медицини, які сприяють покращенню якості діагностики та лікування. Впровадження таких систем може забезпечити лікарям терапевтам доступ до актуальної інформації, підтримку у прийнятті обґрунтованих рішень та покращення результатів лікування пацієнтів. При цьому, важливо зберігати баланс між використанням технологій та професійним досвідом лікарів, а також забезпечити надійний захист медичної інформації та етичність використання систем.

Зокрема, інформаційні системи підтримки прийняття рішень можуть виявитися особливо корисними для лікарів терапевтів в умовах зростаючого обсягу медичної інформації. Завдяки цим системам лікарі можуть отримати швидкий доступ до актуальних клінічних настанов, досліджень та баз знань, що значно полегшує їх завдання з обробки та аналізу інформації. Вони можуть

скористатися алгоритмами та моделями машинного навчання, які допомагають у виявленні складних залежностей та патернів, що більшій мірою випереджається людським сприйняттям.

Інформаційні системи підтримки прийняття рішень також можуть забезпечити індивідуалізований підхід до лікування. За допомогою аналізу даних про пацієнта, його історії хвороби, лікування та результатів обстежень, система може надати рекомендації, що враховують унікальні особливості та потреби конкретного пацієнта. Це сприяє покращенню результатів лікування, зменшенню небажаних побічних ефектів та підвищенню задоволення пацієнта від наданої медичної допомоги.

Однак, при використанні інформаційних систем підтримки прийняття рішень необхідно враховувати певні виклики та обмеження. Наприклад, потрібно забезпечити високу якість та достовірність даних, а також правильність функціонування алгоритмів. Крім того, необхідно вирішити питання етичності, конфіденційності та захисту персональних даних пацієнтів.

У подальшому розвитку інформаційних систем підтримки прийняття рішень для лікарів терапевтів можливе поєднання з іншими технологіями, такими як розширена реальність або інтернет речей. Це може забезпечити більш інтуїтивний та зручний інтерфейс для взаємодії з системою, а також розширені можливості у візуалізації та обробці даних.

У підсумку, інформаційна система підтримки прийняття рішень для лікаря терапевта є актуальним та перспективним напрямом розвитку в сфері медицини. Вона надає лікарям доступ до актуальної медичної інформації, аналізу даних та рекомендацій, що сприяє покращенню діагностики, лікування та результатів медичної практики.

Завдяки використанню технологій штучного інтелекту та машинного навчання, інформаційні системи можуть аналізувати великі обсяги даних, виявляти патерни та надавати індивідуалізовані рекомендації для конкретного

пацієнта. Це сприяє підвищенню ефективності лікування, зменшенню помилок та покращенню задоволення пацієнтів.

Проте, при розробці та використанні таких систем необхідно враховувати етичні, юридичні та конфіденційність. Забезпечення надійного захисту персональних даних пацієнтів, дотримання етичних норм та збереження балансу між технологіями та професійним досвідом лікарів є надзвичайно важливими аспектами.

Загалом, інформаційна система підтримки прийняття рішень для лікаря терапевта може стати потужним інструментом у покращенні медичної практики. Вона сприяє зростанню якості діагностики та лікування, підвищує ефективність роботи лікарів та забезпечує кращі результати для пацієнтів. Проте, важливо враховувати етичні та конфіденційність, забезпечуючи безпеку та захист персональних даних пацієнтів, а також зберігати гуманістичний підхід та експертність медичних фахівців.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. Абдуліна І. Роблячи ставки на розробку програмного забезпечення [Текст] /І. Абдуліна, В. Березанська// Інтелетуальна власність.–2012.–№9.–С8-12.
2. Алгоритми і структура даних: Навчальний посібник / В.М.Ткачук. - ІваноФранківськ: Видавництво Прикарпатського національного університету імені Василя Стефаника, 2016. 286 с.
3. Алгоритми та структури даних. Навчальний посібник / Т. О. Коротєєва. Львів: Видавництво Львівської політехніки, 2014. - 280 с.
4. Алексенко О. В.. Технології програмування та створення програмних продуктів: конспект лекцій. – Суми : Сумський державний університет, 2013. – 133 с.
5. Бадд Т. Об'єктно-орієнтоване програмування в дії / Перекл. з англ. - Спб .: Питер, 1997. - 464 с.
6. Баркмейер Едвард Дж. (2003). Концепції автоматизації системної інтеграції NIST 2003. – 435 с.
7. Безменов М. І. Основи програмування у середовищі Delphi : навч. посіб. – Харків : НТУ «ХПІ», 2010. – 608 с.
8. Глоба Л. С. Розробка інформаційних ресурсів та систем [Електронний ресурс]: конспект лекцій / Л. С. Глоба, Т. М. Кот. - Київ : НТУУ "КПІ", 2014. - 318 с.
9. Грязнова В. О., Єфіменко С. В. Основи методології програмування. - К.: ВПЦ "Київський університет", 2010.
10. Інженерія якості програмного забезпечення: навч. посібник / Г.В Табунщик, Р.К. Кудерметов, Т.І. Брагіна. - Запоріжжя: ЗНТУ, 2013. - 180 с.
11. Комп’ютерні Мережі А.Г. Микитишин, М.М. Митник, П.Д. Стухляк, В.В. Пасічник Режим доступу: http://www.dut.edu.ua/ua/lib/1/category/739/view/1739

12. Кравець П. Об'єктно-орієнтоване програмування: навч. посібник / П.О. Кравець. – Львів: Видавництво Львівської політехніки, 2012. – 624 с.
13. Кузьменко, Н.Г. Комп'ютерні мережі і мережеві технології / Н.Г. Кузьменко. - СПб .: Наука і техніка, 2013. - 368 с.
14. Кульгин М. Технології корпоративних мереж. Енциклопедія. - СПб .: Пітер, 2000. - 704 с.
15. Лавріщева К. М. Програмна інженерія / К. М. Лавріщева. – К. : Академперіодика, 2008. – 319 с
16. Леонов И.В. Введення в методологію розробки програмного забезпечення за допомогою Rational Rose // Ескейп, 2004. – 301 с.
17. Локальні мережі. Повне керівництво. / Под ред. В.В. Самойленко. - К .: Століття К .: НТІ, СПб .: КОРОНА принт, 2002. - 400 с.
18. Николайчук Я. М. Проектування спеціалізованих комп'ютерних систем : навч. посібник / Я. М. Николайчук, Н. Я. Возна, І. Р. Пітух. - Тернопіль : ТзОВ "Терно-граф", 2010. - 392 с.
19. Одом, Уенделл. Офіційне керівництво Cisco по п одготовке до сертифікаційних іспитів CCENT / CCNA ICNDl 100- 101, акад. изд. : Пер. з англ. - М. : ТОВ "І .Д. Вільямові", 2015. - 912 с. : Ил.
20. Отеро, Карлос. "Виклики дизайну програмного забезпечення". Покращення продуктивності IT. ТОВ "Тейлор і Френсіс". Отримано 19 жовтня 2017. – 280 с.
21. Пол Р. Сміт і Річард Сарфаті (1993). Створення стратегічного плану управління конфігурацією за допомогою засобів автоматизованого програмного забезпечення (CASE). Папір для 1993 року Національна група користувачів DOE / Підрядники та об'єкти CAD / CAE.
22. Порєв Г.В. Архітектура сучасних комп’ютерних мереж: Методичний посібник.—Вінниця: УНІВЕРСУМ-Вінниця, 2008— 98 с

23. Сидорова Н. М. Навчання інженерії програмного забезпечення – систематичний огляд літератури/ Н. М. Сидорова // Матеріали міжнародної науково- практичної конференції аспірантів і студентів «Інженерія програмного забезпечення 2011». [Електрон.ресурс].-Спосіб доступу: [http://www.nbuv.gov.ua/portal/ natural/Ipz/2011_2/Sidorova.pdf].
24. Системне програмування. Основні поняття [Electronic resource]. – Mode of access:WWW/URL : https://owncloud.kspu.kr.ua/index.php/s/69e091 5776e e3225579f734e986ad017.
25. Скотт Б., Нейл Т.: Проектування веб-інтерфейсів 2016 — 352 с
26. Столлінгс, В. Комп'ютерні мережі, протоколи і технології Інтернету / В. Столлінгс. - СПб .: BHV, 2005. - 832 с.
27. Таненбаум Е., Уезеролл Д.Т18 Комп'ютерні мережі. 5-е изд. - СПб .: Пітер, 2012. - 960 с .
28. Тарасенко Р.О. Інформаційні технології: навч. посіб. / Тарасенко Р.О., Гаріна С.М., Робоча Т.П. – К.: Алефа, 2008. – 312 с.
29. Технології створення програмних продуктів та інформаційних систем : навч. посібник / М. Ю. Карпенко, Н. О. Манакова, І. О. Гавриленко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. - Харків : ХНУМГ ім. О. М. Бекетова, 2017. - 93 с.
30. Технологічні мережі і системи зв'язку: навчальний посібник / М.А Смичкёк. - 2-е вид. - Москва; Вологда: Инфа-Інженерія, 2019.-400с
31. Цвіркун Л.І. Глобальні комп’ютерні мережі. Програмування мовою PHP: навч. посібник / Л.І. Цвіркун, Р.В. Липовий, під заг. ред. Л.І. Цвіркуна. – Д.: Національний гірничий університет, 2013. – 239 с.
32. Цвіркун Л.І. Розробка програмного забезпечення комп’ютерних систем. Програмування: навч. посіб. [Електронний ресурс] / Л.І. Цвіркун, А.А. Євстігнєєва, Я.В. Панферова ; під заг. ред. Л.І. Цвіркуна. – Електрон. дані та прогр. – М-во освіти і науки України, Нац. техн. ун-т «Дніпровська

політехніка». – 1 електрон. опт. диск (CD-ROM) ; 12 см. – Систем. вимоги: Процесор 32-розрядний (x86) 1 ГГц ; MS Windows 7, 8.1, 10 ; CD-ROM drive. – Назва з титул. екрана. – Дніпро: НТУ «ДП», 2019.

33. Шаховська Н. Б. Алгоритми та структури даних / Н. Б. Шаховська, Р.О. Голощук. – Львів : Магнолія-2006. – 2009. – 216 с.
34. Шевчук І. Б. Інформаційні технології в регіональній економіці: теорія і практика впровадження та використання : монографія. Львів : Видавництво ННВК "АТБ", 2018. 448 с.
35. Шеховцов В.А. Операційні системи / В.А. Шеховцов. – К. : Видавнича група BHV, 2005. – 576 с.
36. Company S. Maven: The Definitive Guide / Sonatype Company. – Sebastopol: O'Reilly Media, 2008. – 470 с. – (first edition).
37. Craig C. Learn Android Studio: Build Android Apps Quickly and Effectively / C.
38. Craig, A. Gerber., 2015. – 486 с.
39. Hans M. Appium Essentials / Manoj Hans., 2015. – 188 с.
40. Rahman M. BROWSER-BASED AUTOMATION TESTING USING SELENIUM WEBDRIVER / Md.Foyzur Rahman., 2014. – 94 с.
41. Sharan K. Beginning Java 8 Fundamentals: Language Syntax, Arrays, Data Types, Objects, and Regular Expressions / Kishori Sharan., 2014. – 828 с. – (first edition).
42. Siminiuc A. Improve Selenium Code with Automation Patterns: Page Object Model Page Factory Page Elements Base Page Loadable Component / Alex Siminiuc., 2017. – 112 с.
43. Raskin J. The Humane Interface: New Directions for Designing Interactive Systems / Jef Raskin. – Denver: Addison-Wesley Professional, 2000. – 233 с. – (1).

44. Galitz W. O. The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques / Wilbert O. Galitz. – New York: Wiley, 2007. – 888 с.
45. Documenting Software Architectures: Views and Beyond / F.Bachmann, L. Bass, D. Garlan, J. Ivers. – Boston: Addison-Wesley Professional, 2010. – 592 с.
46. Perry D. E. Foundations for the Study of Software Architecture / D. E. Perry, A. L. Wolf. – Boulder: ACM SIGSOFT, 1992. – 13 с. – (SOFTWARE ENGINEERING NOTES). 6. What is your definition of software architecture?. // Carnegie Mellon University. – 2017. – №3854. – С. 6.
47. Gero J. S. Design Prototypes: A Knowledge Representation Schema for Design / John S. Gero. – New York: AI Magazine, 1990. – 26 с.
48. van Drongelen M. Android Studio Cookbook: Design, test, and debug your apps using Android Studio / Mike van Drongelen. – Лондон: Packt Publishing, 2015. – 232 с.
49. Brewer C. Designing Interfaces / C. Brewer, J. Tidwell, A. Valencia. – New York: 'Reilly Media, 2020. – 600 с.
50. Why Is a GUI Important? [Електронний ресурс] // Reference. – 2018. – Режим доступу до ресурсу: https://www.reference.com/world-view/gui-important95d42a64e0c41332.
51. Eckel Bruce. Thinking in Java, – Prentice Hall, 2006. – 1 видання, 1150с.
52. Meier Reto. Professional Android, – Wrox, 2018. – 4 видання, 984с.
53. Mew Kyle. Mastering Android Studio 3: Build Dynamic and Robust Android applications, – Packt Publishing, 2017. – 220с.
54. Mew Kyle. Android Design Patterns and Best Practices, – Packt Publishing, 2017. 534с.

55. Architecture of JavaScript Applications [Електронний ресурс] // Oracle. – 2010. – Режим доступу до ресурсу: https://docs.oracle.com/cd/E19957-01/816-641110/getstart.htm.
56. Sullivan D. NoSQL for Mere Mortals / Dan Sullivan. – New York: Addison-Wesley Professional, 2015. – 542 с.
57. Marrs T. JSON at Work: Practical Data Integration for the Web / Tom Marrs., 2017. – 376 с.
58. Privacy and Security in Firebase [Електронний ресурс] // Firebase. – 2019. – Режим доступу до ресурсу: https://firebase.google.com/support/privacy.
59. Murphy M. The Busy Coder Guide to Advanced Android Development / Mark Murphy. – United States of America: CommonsWare, 2009. – 509 с.
60. Wiegers K. Software Requirements (Developer Best Practices) / Karl Wiegers. – Washington: Microsoft Press, 2010. – 670 с.
61. Beatty J. Software Requirements / Joy Beatty. – Washington: Microsoft Press, 2013. – 455 с.
62. Eck D. J. Introduction to Computer Graphics / David J. Eck. – New York: Hobart and William Smith Colleges, 2018. – 411 с.
63. Kaner C. Testing Computer Software / Cem Kaner. – Vancouver: Wiley, 1999. –480 с.
64. J. Myers G. The Art of Software Testing / Glenford J. Myers. – New Jersey: John Wiley & Sons, Inc., 2004. – 375 с.
65. Crispin L. Agile Testing / L. Crispin, J. Gregory. – Toronto: Addison-Wesley, 2009. – 274 с.
66. Kaner C. Lessons Learned in Software Testing: A Context-Driven Approach / Cem Kaner. – Vancouver: Wiley, 2001. – 227 с.

ДОДАТОК А.

Лістинг програми

) DataModule;

unit Unit3;, Classes, DB, ADODB;= class (TDataModule): TADOConnection;: TADOQuery;: TADOQuery;: TADOQuery;_Karta: TADOQuery;: TADOQuery;: TADOQuery;: TADOQuery;: TADOQuery;: TADOQuery;: TDataSource;: TDataSource;: TDataSource;_Karta: TDataSource;: TDataSource;: TDataSource;: TDataSource;: TDataSource;: TDataSource;: TADOQuery;: TDataSource;: TADOQuery;: TDataSource;_pz: TAutoIncField;: TWideStringField;: TWideStringField;: TWideStringField;: TWideStringField;: TDateField;: TWideStringField;: TWideStringField;: TWideStringField;: TWideStringField;: TWideStringField;: TWideStringField;: TWideStringField;: TWideStringField;: TWideStringField;: TWideStringField;: TDateField;: TIntegerField;DataModuleCreate (Sender: TObject);

{ Private declarations }

{ Public declarations };: TDM1;

*{$R *. dfm}.*

2) Головна форма терапевта;

unit Unit1;, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,, Menus, ExtCtrls, Grids, DBGrids, Unit7, TeeProcs, TeEngine,, DbChart, StdCtrls, DateUtils, DBCtrls, Series, Mask;= class (TForm): TDBGrid;: TMainMenu;: TMenuItem;: TMenuItem;: TMenuItem;: TOpenDialog;: TPanel;: TRadioGroup;: TGroupBox;: TLabel;: TLabel;: TLabel;: TLabel;: TPanel;: TPanel;: TRadioGroup;: TPanel;: TPanel;: TGroupBox;: TEdit;: TButton;N4Click (Sender: TObject);DBGrid2Enter (Sender: TObject);DBGrid2CellClick (Column: TColumn);DBGrid1DrawColumnCell (Sender: TObject; const Rect: TRect;: Integer; Column: TColumn; State: TGridDrawState);

{ procedure N1Click (Sender: TObject);N2Click (Sender: TObject);N3Click (Sender: TObject); }N6Click (Sender: TObject);

{ procedure FormCreate (Sender: TObject); }DBGrid1DblClick (Sender: TObject);Button2Click (Sender: TObject);

{ Private declarations }

{ Public declarations };: TForm1;, s,ClWhere: string;: integer;, Unit3, Unit4, Unit5, Unit6, Unit8;

*{$R *. dfm}*

// Функція записує новий текст для закладки з іменем abmname.

// Функція повертає значення True, якщо закладка знайдена і її текст змінений

// і False - якщо закладка не знайдена.

// Переустановка тексту закладки виплоняется так:

// - Одержуємо посилання на об'єкт- діапазон, який містить текст закладки.

// - Видаляємо закладку.

// - Установлюємо новий текст для об'єкта-діапазону.

// - Створюємо нову закладку з діапазоном, який містить новий текст.

function Setbmtext (var abms: Variant; const abmname, atext: String): Boolean;

Bm, Rng: Variant;

begin

// Перевіряємо - чи існує закладка із заданим іменем.

Result: = abms. Exists (abmname);

// Якщо закладка не знайдена - виходимо.

if not Result then Exit;

// Посилання на закладку.

Bm: = aBms. Item (aBmName);

// Посилання на діапазон, пов'язаний із закладкою.

Rng: = Bm. Range;

// Видалення закладки.

Bm. Delete;

// Заміняємо текст у діапазоні.

Rng. Text: = atext;

// Додаємо нову закладку з таким же іменем.

// aBms. Add (aBmName, Rng);;TForm1. N4Click (Sender: TObject);. Terminate;;TForm1. DBGrid2Enter (Sender: TObject);

*{DM1. QueryDiagnozy. Close;. QueryDiagnozy. SQL. Text: = ('SELECT DISTINCT * FROM diagnozy WHERE diagnozy. id_pz='+clWhere);. QueryDiagnozy. Open; };TForm1. DBGrid2CellClick (Column: TColumn);, i,d,n, KolvoStrokDiag: integer;,x,ClwhereOcenka,ClWherePZ, NowYear: string;,x2,x3,x4,x5,proverka: real;: array [1.27] of integer;. Series1. Clear;: =0;: =0;*

// Label3. Caption: =DBText2. Caption+' '+DBText3. Caption+' '+DBText4. Caption;. QueryVozrastMedmer. Close;

// Label2. Font. Color: =clRed;

*// Label4. Visible: =true;: =DBGrid2. DataSource. DataSet. Fields [0]. AsString;. QueryDiagnozy. Close;. QueryDiagnozy. SQL. Text: = ('SELECT DISTINCT * FROM diagnozy WHERE diagnozy. id_pz='+clWhere);. QueryDiagnozy. Open;: =DBGrid2. DataSource. DataSet. Fields [0]. AsString;. QueryDiagnozy. Close;. QueryDiagnozy. SQL. Text: = ('SELECT DISTINCT **

*FROM diagnozy WHERE diagnozy. id_pz='+clWhere);. QueryDiagnozy. Open;;TForm1. DBGrid1DrawColumnCell (Sender: TObject; const Rect: TRect;: Integer; Column: TColumn; State: TGridDrawState);: TDateTime;: string;,KolvoStrok,proverka: integer;,DateM,DateD,DateHour,DateMin,DateSec: Word;: string;: =DBGrid2. DataSource. DataSet. Fields [0]. AsString;: =Form8. DBGrid1. DataSource. DataSet. Fields [1]. AsString;. ZQuery7. Close;. ZQuery7. SQL. Text: = ('SELECT DISTINCT * FROM diagnozy WHERE diagnozy. id_pz='+clWhere+' and diagnozy. МедМероприятие='+#39+ClWhere2+#39);. ZQuery7. Open;: = FormatDateTime ('yyyy', Now);: =Form5. DBGrid1. DataSource. DataSet. RecordCount;KolvoStrok=0 then exiti: =1 to KolvoStrok do: =0;(StrToDate (Form5. DBGrid1. DataSource. DataSet. FieldByName ('Дата'). AsString),DateY, DateM, DateD);*

// DiagnozDate: =StrToDate (Form5. DBGrid1. DataSource. DataSet. Fields [3]. AsString);

// DecodeDate (DiagnozDate, DateY, DateM, DateD);: =IntToStr (DateY);DiagnozYear=NowYear{ условие }Begin. DBGrid1. Canvas. Brush. Color: =clGreen;. DBGrid1. Canvas. Font. Color: =clWhite;. DBGrid1. Canvas. FillRect (Rect);: =1;. DBGrid1. Canvas. Brush. Color: =clTeal;. DBGrid1. Canvas. Font. Color: =clWhite;. DBGrid1. Canvas. FillRect (Rect);;

// якщо рядок був виділений, залишаємо "підсвічені" кольори

{IF gdSelected IN StateBegin(Sender). Canvas. Brush. Color: = clHighLight;(Sender). Canvas. Font. Color: = clHighLightText;. Canvas. FillRect (Rect);; }(Sender). DefaultDrawColumnCell (Rect,DataCol,Column,State);proverka=1 then break else. DBGrid1. DataSource. DataSet. Next;;;

{TForm1. N1Click (Sender: TObject);. Visible: =false;. visible: =true;. Visible: =false;;TForm1. N2Click (Sender: TObject);. Visible: =false;. visible: =true;. Visible: =false;;TForm1. N3Click (Sender: TObject);. Visible: =true;;

}TForm1. N6Click (Sender: TObject); // ОТЧЕТ, wdDocs, wdDoc, wdBms: Variant;: TOpenDialog;: = OpenDialog1;Od. InitialDir = '' then. InitialDir: = ExtractFilePath (Application. ExeName)

;

Od. Title: = 'Виберіть шаблон, на основі якого буде створений новий документ';

if not Od. Execute then Exit;not Fileexists (Od. Filename) then begin (

'Файл із заданим іменем не знайдений. Дія скасована. '

,mtwarning, [mbok], 0

);;;: = Createoleobject ('Word. Application');('Не вдалося запустити MS Word. Дія скасована. ');

Exit;

end;

// Робимо видимим вікно MS Word.

wdapp. Visible: = True;

// Посилання на колекцію документів.

wddocs: = wdapp. Documents;

// Спроба відкрити обраний файл.

wddoc: = wddocs. Open (Filename: =Od. Filename);

// Підключаємося до колекції закладок.

wdbms: = wddoc. Bookmarks;

// Шукаємо закладки з потрібними іменами й змінюємо їх текст, у відповідність

// с даними, уведеними на формі.

try(wdBms, 'id_pz', DM1. QueryPacienty. FieldByName ('id_pz'). AsString);(wdBms, 'data_ofrml',DM1. QueryPacienty. FieldByName ('Дата оформления'). AsString);(wdBms, 'n_bol',DM1. QueryBolnica. FieldByName ('Наименование'). AsString);(wdBms, 'adr_bol',DM1. QueryBolnica. FieldByName ('Адрес'). AsString);(wdBms, 'kod_bol',DM1. QueryBolnica. FieldByName ('Код ОГРН'). AsString);(wdBms, 'FIO', DM1. QueryPacienty. FieldByName ('Фамилия'). AsString + ' ' + DM1. QueryPacienty. FieldByName ('Имя'). AsString + ' ' + DM1. QueryPacienty. FieldByName ('Отчество'). AsString);(wdBms, 'Data_rozhd', DM1. QueryPacienty. FieldByName ('Дата рождения'). AsString);(wdBms, 'pol', DM1. QueryPacienty. FieldByName ('Пол'). AsString);(wdBms, 'pasport', DM1. QueryPacienty. FieldByName ('Паспорт'). AsString);(wdBms, 'adres_pac', DM1. QueryPacienty. FieldByName ('Адрес'). AsString);(wdBms, 'telephon', DM1. QueryPacienty. FieldByName ('Телефон'). AsString);(wdBms, 'nomer_polis', DM1. QueryPacienty. FieldByName ('Полис ОМС'). AsString);

{QueryOtchet}

// Дописати поле Datatime у запиті

*DM1. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Заключение врача терапевта' + #39;*

*DM1. QueryOtchet. Open;(wdBms, 'terapevt', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Антропометрия' + #39;. QueryOtchet. Open;(wdBms, 'antropometria', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Измерение АД' + #39;. QueryOtchet. Open;(wdBms, 'izmerenie_ad', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Уровень общего холестерина в крови' + #39;. QueryOtchet. Open;(wdBms, 'holesterin', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Определение уровня глюкози в крови' + #39;. QueryOtchet. Open;(wdBms, 'glukoza', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Определение суммарного сердечно-сосудистого риска' + #39;. QueryOtchet. Open;(wdBms, 'serdechny_risk', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие =*

*' + #39 + 'ЭКГ (мужчини) ' + #39;. QueryOtchet. Open;(wdBms, 'ekg', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Осмотр фельдшера' + #39;. QueryOtchet. Open;(wdBms, 'osm_feldsher', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Флюорография легких' + #39;. QueryOtchet. Open;(wdBms, 'flurogramma', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Маммография' + #39;. QueryOtchet. Open;(wdBms, 'mammographia', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Клінічний аналіз крові' + #39;*

*DM1. QueryOtchet. Open;(wdBms, 'klin_analiz_krovi', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Клинический анализ крови развернутий' + #39;*

*DM1. QueryOtchet. Open;(wdBms, 'klin_analiz_krovi_razv', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Анализ крови биохимический общетерапевтический' + #39;. QueryOtchet. Open;(wdBms, 'bio_analiz_krovi', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Общий анализ мочи' + #39;. QueryOtchet. Open;(wdBms, 'mocha', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Исследование кала на скритую кровь' + #39;. QueryOtchet. Open;(wdBms, 'kal', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Определение уровня ПСА в крови' + #39;. QueryOtchet. Open;(wdBms, 'psa', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'УЗИ органов брюшной полости' + #39;. QueryOtchet. Open;(wdBms, 'uzi', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Измерение внутриглазного давление' + #39;. QueryOtchet. Open;(wdBms, 'glaz_davl', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Дуплексное сканирование брахиоцефальних артерий' + #39;. QueryOtchet. Open;(wdBms, 'dsb_art', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'ЭФГДС' + #39;. QueryOtchet. Open;(wdBms, 'efgds', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Осмотр врача хирурга' + #39;. QueryOtchet. Open;(wdBms, 'osm_vrach_hirurg', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Осмотр врача хирурга-проктолога' + #39;. QueryOtchet. Open;(wdBms, 'osm_vrach_hirurg_proct', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 +*

*ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Колоноскопия (ректороманоскопия) ' + #39;. QueryOtchet. Open;(wdBms, 'kolonoskopia', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Определение липидного спектра крови' + #39;. QueryOtchet. Open;(wdBms, 'spectr_krovi', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Осмотр врача акушера-гинеколога' + #39;. QueryOtchet. Open;(wdBms, 'akusher', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Определение концентрации гликированного Hb в крови' + #39;. QueryOtchet. Open;(wdBms, 'konc_hb_krovi', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Осмотр врача офтальмолога' + #39;. QueryOtchet. Open;(wdBms, 'oftalmolog', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);. QueryOtchet. SQL. Text: = 'Select * from diagnozy where id_pz = '+ #39 + ClWhere + #39 + ' and ' + 'МедМероприятие = ' + #39 + 'Осмотр врача невролога' + #39;. QueryOtchet. Open;(wdBms, 'nevrolog', DM1. QueryOtchet. FieldByName ('Диагноз'). AsString);*

finally

// Зберігати документ випливає під іншим іменем, щоб не перезаписати шаблон.

// wdapp. Displayalerts: = False; // Відключаємо режим показу попереджень.

// wddoc. Saveas (Filename: =.);

// wdapp. Displayalerts: = True; // Включаємо режим показу попереджень.

// Закриваємо документ.

// wddoc. Close;

// Закриваємо MS Word.

*// wdApp. Quit;;; {TForm1. FormCreate (Sender: TObject);. Visible: =false;. Visible: =false;; }TForm1. DBGrid1DblClick (Sender: TObject);: string;: =DM1. QueryPacienty. FieldByName ('id_pz'). AsString;: =DM1. QueryVozrastMedmer. FieldByName ('МедМероприятие'). AsString;. ZQuery8. Close;. ZQuery8. SQL. Text: = ('SELECT DISTINCT * FROM diagnozy WHERE diagnozy. id_pz='+clWhere+' and diagnozy. МедМероприятие='+#39+ClWhere2+#39);. ZQuery8. Open;. Show;;TForm1. Button2Click (Sender: TObject);. RadioGroup1. ItemIndex: =-1;. RadioGroup1. ItemIndex: =0;. ShowModal;;.*

3) Головна форма мед. працівника;

unit Unit5;, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,, Menus, ExtCtrls, Grids, DBGrids, TeeProcs, TeEngine,, DbChart, StdCtrls, DateUtils, DBCtrls, Series, Mask, jpeg;= class (TForm): TDBGrid;: TDBGrid;: TMainMenu;: TMenuItem;DBGrid2DblClick (Sender: TObject);N1Click (Sender: TObject);

{ Private declarations }

{ Public declarations };: TForm2;, Unit3, Unit4, Unit6, Unit7;

*{$R *. dfm}TForm2. DBGrid2DblClick (Sender: TObject);, id_pz: string;_pz: =DM1. QueryPacienty. FieldByName ('id_pz'). AsString;: =DM1. QueryVozrastMedmer. FieldByName ('МедМероприятие'). AsString;. ZQuery8. Close;. ZQuery8. SQL. Text: = ('SELECT DISTINCT * FROM diagnozy WHERE diagnozy. id_pz='+id_pz+' and diagnozy. МедМероприятие='+#39+ClWhere2+#39); DM1. ZQuery8. Open;. Show;;TForm2. N1Click (Sender: TObject);. terminate;;.*

4) Форма редагування діагнозу;

unit Unit6;, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, DBCtrls, Mask, Menus;

type= class (TForm): TDBEdit;: TDBEdit;_pz: TDBEdit;: TDBMemo;: TEdit;: TLabel;: TLabel;: TLabel;: TLabel;: TButton;: TButton;: TLabel;Button1Click (Sender: TObject);Button2Click (Sender: TObject);

{ Private declarations }

{ Public declarations };: TForm3;, Unit3, Unit4, Unit5, Unit7;

*{$R *. dfm}TForm3. Button1Click (Sender: TObject);dm1. QueryDiagnozy. Modified then. QueryDiagnozy. Post;;TForm3. Button2Click (Sender: TObject);. Close;;.*

5) Форма список діагнозів;

unit Unit7;, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,, Menus, Grids, DBGrids;= class (TForm): TDBGrid;: TMainMenu;: TMenuItem;: TDBGrid;N1Click (Sender: TObject);FormShow (Sender: TObject);

{ Private declarations }

{ Public declarations };: TForm5;, Unit3, Unit4, Unit5, Unit6;

*{$R *. dfm}TForm5. N1Click (Sender: TObject);. QueryDiagnozy. Close;. QueryDiagnozy. Open;*

DM1. QueryDiagnozy. Append;. QueryDiagnozy. FieldByName ('Дата'). AsString: =DateToStr (Date);. QueryDiagnozy. FieldByName ('МедМероприятие'). AsString: =DM1. QueryVozrastMedmer. FieldByName ('МедМероприятие'). AsString;. QueryDiagnozy. FieldByName ('id_pz'). AsString: =DM1. QueryPacienty. FieldByName ('id_pz'). AsString;. Edit1. Text: =DM1. QueryPacienty. FieldByName ('Фамилия'). AsString+' '+DM1. QueryPacienty. FieldByName ('Имя'). AsString+' '+DM1. QueryPacienty. FieldByName ('Отчество'). AsString;. QueryDiagnozy. Post;. show;;TForm5. FormShow (Sender: TObject);dostup <> 1 then. Visible: =False;;.

6) Форма пацієнта;

unit Unit8;, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,, Menus, ExtCtrls, Grids, DBGrids, Unit7, TeeProcs, TeEngine,, DbChart, StdCtrls, DateUtils, DBCtrls, Series, Mask, ComCtrls;= class (TForm): TPanel;: TDBChart;: TPieSeries;: TLabel;: TLabel;: TPanel;: TDBEdit;: TDBEdit;: TDBEdit;: TBevel;: TLabel;: TLabel;: TBevel;: TLabel;: TDBEdit;: TLabel;: TDBEdit;: TPanel;: TPageControl;: TTabSheet;: TTabSheet;: TDBGrid;: TButton;: TRadioGroup;: TRadioGroup;: TDBGrid;: TButton;: TButton;RadioGroup1Click (Sender: TObject);DBGrid1CellClick (Column: TColumn);

{ Private declarations }

{ Public declarations };: TForm8;Unit3, Unit1;

*{$R *. dfm}TForm8. RadioGroup1Click (Sender: TObject);, i,d,n, KolvoStrokDiag: integer;,x,ClwhereOcenka,ClWherePZ, NowYear: string;,x2,x3,x4,x5,proverka: real;: array [1.27] of integer;. Series1. Clear;: =0;: =0;*

// Label3. Caption: =DBText2. Caption+' '+DBText3. Caption+' '+DBText4. Caption;. QueryVozrastMedmer. Close;. Font. Color: =clRed;

// Label4. Visible: =true;[1]: =21;i: =2 to 27 do[i]: =mas [i-1] +3;: = YearsBetween (Date,DM1. QueryPacienty. FieldByName ('Дата рождения'). AsDateTime);. Caption: =IntToStr (years);i: =1 to 27 doyears=mas [i] then

*// Label4. Visible: =false;. Font. Color: =clBlack;: =IntToStr (years);RadioGroup1. ItemIndex = 0 then. QueryVozrastMedmer. SQL. Text: = ('SELECT id_medmer, МедМероприятие FROM vozrast_medmer WHERE vozrast_medmer. '+ x + '=1' + ' and vozrast_medmer. Этап=1');. QueryVozrastMedmer. Open;: =Form1. DBGrid2. DataSource. DataSet. Fields [0]. AsString;: =DBGrid1. DataSource. DataSet. Fields [1]. AsString;. ZQuery8. Close;. ZQuery8. SQL. Text: = ('SELECT DISTINCT * FROM diagnozy WHERE diagnozy. id_pz='+clWhere+' and diagnozy. МедМероприятие='+#39+ClWhere2+#39);. ZQuery8. Open;: =1;. QueryVozrastMedmer. SQL. Text: = ('SELECT id_medmer, МедМероприятие FROM vozrast_medmer WHERE vozrast_medmer. '+ x + '=1');. QueryVozrastMedmer. Open;: =Form1. DBGrid2. DataSource. DataSet. Fields [0]. AsString;: =DBGrid1. DataSource. DataSet. Fields [1]. AsString;. ZQuery8. Close;. ZQuery8. SQL. Text: = ('SELECT DISTINCT * FROM diagnozy WHERE diagnozy. id_pz='+clWhere+' and diagnozy. МедМероприятие='+#39+ClWhere2+#39);. ZQuery8. Open;: =1;;;;*

{Диаграмма}proverka=1 then: = FormatDateTime ('yyyy', Now);: =Form1. DBGrid2. DataSource. DataSet. Fields [0]. AsString;

{оценка Норма}: ='Норма';

*DM1. QueryDiagnozyGraphik. Close;. QueryDiagnozyGraphik. SQL. Text: = ('SELECT DISTINCT * FROM diagnozy WHERE diagnozy. id_pz='+clWherePZ+' and diagnozy. ОценкаПоказателей='+#39+ClWhereOcenka+#39+' and YEAR (Дата) =' + NowYear);. QueryDiagnozyGraphik. Open;: =DM1. QueryDiagnozyGraphik. RecordCount;*

{оцінка Невеликі відхилення}

Clwhereocenka: ='Невеликі відхилення';

*DM1. QueryDiagnozyGraphik. Close;. QueryDiagnozyGraphik. SQL. Text: = ('SELECT DISTINCT * FROM diagnozy WHERE diagnozy. id_pz='+clWherePZ+' and diagnozy. ОценкаПоказателей='+#39+ClWhereOcenka+#39+' and YEAR (Дата) =' + NowYear);. QueryDiagnozyGraphik. Open;: =DM1. QueryDiagnozyGraphik. RecordCount;*

{оцінка Серйозні відхилення}

ClwhereOcenka: ='Серйозні відхилення';

*DM1. QueryDiagnozyGraphik. Close;. QueryDiagnozyGraphik. SQL. Text: = ('SELECT DISTINCT * FROM diagnozy WHERE diagnozy. id_pz='+clWherePZ+' and diagnozy.*

ОценкаПоказателей='+#39+ClWhereOcenka+#39+' and YEAR (Дата) =' + NowYear);. QueryDiagnozyGraphik. Open;: =DM1. QueryDiagnozyGraphik. RecordCount;

{Не пройдено}: =DM1. QueryVozrastMedmer. RecordCount-x2-x3-x4;

{передача параметрів}

if x1>=1 then Form8. Series1. Add (x1, 'Не пройдене', clwhite);

if x2>=1 then Form8. Series1. Add (x2, 'Норма', clgreen);x3>=1 then Form8. Series1. Add (x3, 'Невеликі відхилення', clyellow);

if x4>=1 then Form8. Series1. Add (x4, 'Серйозні відхилення', clred);

end;

// Відсоток проходження мед. заходів

try

Form8. Label9. Visible: =True;: =0;

*x5: =x1+x2+x3+x4;: =100-x1/x5*100;. label10. Caption: = (Formatfloat ('0.0', x5));;*

*// завершення;Tform8. Dbgrid1Cellclick (Column: Tcolumn);: string;: =DM1. Querypacienty. Fieldbyname ('id_pz'). Asstring;: =DM1. Queryvozrastmedmer. Fieldbyname ('МЕДМЕРОПРИЯТИЕ'). Asstring;. Zquery8. Close;. Zquery8. SQL. Text: = ('SELECT DISTINCT * FROM diagnozy WHERE diagnozy. id_pz='+clwhere+' and diagnozy. Медмероприятие='+#39+Clwhere2+#39);. Zquery8. Open;. Show;;.*

7) Форма авторизації;

unit Unit4;Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,, Menus, ExtCtrls, Grids, DBGrids, TeeProcs, TeEngine,, DbChart, StdCtrls, DateUtils, DBCtrls, Series, Mask, jpeg;= class (TForm): TImage;: TLabel;: TButton;: TButton;: TEdit;: TEdit;: TLabel;: TLabel;: TImage;: TDBText;: TButton;btnOKBtnClick (Sender: TObject);btnCancelBtnClick (Sender: TObject);Image1Click (Sender: TObject);Button1Click (Sender: TObject);

{ Private declarations }

{ Public declarations };: TPasswordDlg;: integer;Unit1, Unit7, Unit5, Unit3;

*{$R *. dfm}TPasswordDlg. btnOKBtnClick (Sender: TObject);,x2: string;. QueryUsers. Close;. QueryUsers. SQL. Text: = 'Select * from users where login = '*

+ #39 + edt1. Text + #39+ ' and password = ' + #39 + edt2. Text + #39;. QueryUsers. Open;: =0;: =DM1. QueryUsers. FieldByName ('password'). AsString;: =DM1. QueryUsers. FieldByName ('login'). AsString;x1=Edt1. Text thenx2=Edt2. Text then: =DM1. QueryUsers. FieldByName ('dostup'). AsInteger;dostup of

: begin('Помилка: перевірте правильність даних. ');;

: // Терапевт. Show;. Caption: =DM1. QueryUsers. FieldByName ('Должность'). AsString + ': ' + DM1. QueryUsers. FieldByName ('ПІБ'). AsString;. Visible: =False;;

: // Главврач. Show;. Caption: =DM1. QueryUsers. FieldByName ('Должность'). AsString + ': ' + DM1. QueryUsers. FieldByName ('ПІБ'). AsString;. Visible: =False;;

: // Акушер/Гинеколог. Caption: =DM1. QueryUsers. FieldByName ('Должность'). AsString + ': ' + DM1. QueryUsers. FieldByName ('ПІБ'). AsString;. Show;. QueryVozrastMedmer. SQL. Text: = ('Select МедМероприятие from vozrast_medmer where id_medmer IN (11,29,9) ');. QueryVozrastMedmer. Open;. Visible: =False;;

: // Лаборант. Caption: =DM1. QueryUsers. FieldByName ('Должность'). AsString + ': ' + DM1. QueryUsers. FieldByName ('ПІБ'). AsString;. Show;. QueryVozrastMedmer. SQL. Text: = ('Select МедМероприятие from vozrast_medmer where id_medmer IN (2,3,4,5,12,13,14,15,16,17) ');. QueryVozrastMedmer. Open;. Visible: =False;;

: // Диагност. Caption: =DM1. QueryUsers. FieldByName ('Должность'). AsString + ': ' + DM1. QueryUsers. FieldByName ('ПІБ'). AsString;. Show;. QueryVozrastMedmer. SQL. Text: = ('Select МедМероприятие from vozrast_medmer where id_medmer IN (10,18,23,24,28,30) ');. QueryVozrastMedmer. Open;. Visible: =False;;

: // Невролог. Caption: =DM1. QueryUsers. FieldByName ('Должность'). AsString + ': ' + DM1. QueryUsers. FieldByName ('ПІБ'). AsString;. Show;. QueryVozrastMedmer. SQL. Text: = ('Select МедМероприятие from vozrast_medmer where id_medmer IN (32) ');. QueryVozrastMedmer. Open;. Visible: =False;;

: // Окулист. Caption: =DM1. QueryUsers. FieldByName ('Должность'). AsString + ': ' + DM1. QueryUsers. FieldByName ('ПІБ'). AsString;. Show;. QueryVozrastMedmer. SQL. Text: = ('Select МедМероприятие from vozrast_medmer where id_medmer IN (31, 19) ');. QueryVozrastMedmer. Open;. Visible: =False;;

: // Хирург. Caption: =DM1. QueryUsers. FieldByName ('Должность'). AsString + ': ' + DM1. QueryUsers. FieldByName ('ПІБ'). AsString;. Show;. QueryVozrastMedmer. SQL. Text: = ('Select МедМероприятие from vozrast_medmer where id_medmer IN (25,26,27) ');. QueryVozrastMedmer. Open;. Visible: =False;;

*: // Кардиолог. Caption: =DM1. QueryUsers. FieldByName ('Должность'). AsString + ': ' + DM1. QueryUsers. FieldByName ('ПІБ'). AsString;. Show;. QueryVozrastMedmer. SQL. Text: = ('Select МедМероприятие from vozrast_medmer where id_medmer IN (6,7,8) ');. QueryVozrastMedmer. Open;. Visible: =False;;;;TPasswordDlg. btnCancelBtnClick (Sender: TObject);. Terminate;;TPasswordDlg. Image1Click (Sender: TObject);. Terminate;;TPasswordDlg. Button1Click (Sender: TObject);x1,x2: string;: integer;. QueryUsers. Close;. QueryUsers. SQL. Text: = 'Select * from users where login = '*

+ #39 + edt1. Text + #39+ ' and password = ' + #39 + edt2. Text + #39;. QueryUsers. Open;: =0;: =DM1. QueryUsers. FieldByName ('password'). AsString;: =DM1. QueryUsers. FieldByName ('login'). AsString;x1=Edt1. Text thenx2=Edt2. Text then: =DM1. QueryUsers. FieldByName ('dostup'). AsInteger;dostup > 0 then. Visible: =False;. Visible: =False;. Visible: =False;. Visible: =True;. Visible: =False;. Visible: =False;. Visible: =True;. Visible: =True;. Visible: =False;('Доступа нет');;

end.