Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ З ПІДБОРУ МЕДИЧНОГО ПЕРСОНАЛУ ДЛЯ ЛЮДЕЙ З ОБМЕЖЕНИМИ МОЖЛИВОСТЯМИ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-20-1
Мірошник Є.І.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Тітова О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Мірошнику Євгену Івановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та розробка інформаційної системи з підбору медичного персоналу для людей з обмеженими можливостями

затверджена наказом по університету від « 22 » жовтня 2021 року № 1574Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 26 листопада 2021 р.
3. Вихідні дані до роботи географічне розташування, критерії онлайн підтримки, мова програмування java, середовище розробки.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд невирішених питань пацієнтів з обмеженими можливостями.
2. Дослідження компоненту медичної інформаційної системи.
3. Розробка алгоритму онлайн підтримки пацієнта.
4. Обробка та аналіз отриманих результатів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність невирішених питань пацієнтів з обмеженими можливостями у медичній інформаційній системі, постановка задачі, опис алгоритмів, схема даних та функціональних можливостей, моделювання компонентів системи.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Бєлова Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	22.10.2021	
2	Аналіз завдання, підбір літератури	23.10.21-25.10.21	
3	Аналіз літератури з досліджуваної проблеми	25.10.21-31.10.21	
4	Аналіз технічних засобів	01.11.21-02.11.21	
5	Розробка методу	03.11.21-10.11.21	
6	Програмна реалізація	10.11.21-23.11.21	
7	Оформлення пояснювальної записки	23.11.21-02.12.21	
8	Перевірка на плагіат	03.12.2021	
9	Рецензування	04.12.2021	
10	Підготовка презентації та доповіді	05.12.2021	
11	Занесення роботи в електронний архів	06.12.2021	
12	Попередній захист кваліфікаційної роботи	06.12.2021	

Дата видачі завдання __22___ ___жовтня_____ 2021 р.

Студент _____
(підпис)

Керівник роботи _____ _____доц. Тітова О.В._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 57 с., 2 табл., 15 рис., 28 джерел.

МЕДИЧНА ІНФОРМАЦІЙНА СИСТЕМА, МОДЕЛЮВАННЯ, ПРОЕКТУВАННЯ, БАЗА ДАНИХ, SPRING, СИСТЕМА, ДЕКОМПОЗИЦІЯ, АРХІТЕКТУРА, МІКРОСЕРВІСИ, ПІДТРИМКА.

Об’єктом дослідження є автоматизація процесу надання допомоги пацієнтам з обмеженими можливостями.

Метою дослідження є розробка компоненту медичної інформаційної системи для вирішення частини питань людей з обмеженими можливостями.

Спроектовано та досліджено компонент медичної інформаційної системи підбору медичного персоналу для людей с обмеженими можливостями. Розроблено алгоритм онлайн підтримки пацієнта та виклику сиділки на дім. Розроблено інтерфейс користування додатком для пацієнта. Розглянуто частину невирішених питань обмежених пацієнтів у медичній інформаційній системі.

У результаті роботи здійснена розробка компоненту у вигляді Android застосунку.

MEDICAL INFORMATION SYSTEM, MODELING, DESIGN, DATA BASE, SPRING, SYSTEM, DECOMPOSITION, ARCHITECTURE, MICROSERVICES, SUPPORT.

The object of the study is to automate the process of providing care to patients with disabilities.

The study aims to develop a component of medical information system to address some of the issues of people with disabilities.

The component of the medical information system for selecting medical staff for people with disabilities is designed and investigated. Developed an algorithm for online support for the patient and home care provider. The patient interface with the add-on is developed. Part of the unresolved issues of disadvantaged patients in the medical information system is investigated.

The work has resulted in the development of a component in the form of an Android device.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

МІС – медична інформаційна система

ЕМК – електронна медична карта

HIS – Hospital Information System

ІС – інформаційна система

АРМ – автоматизоване робоче місце

МОЗ – Міністерство Охорони Здоров’я

БД – база даних

БЗ – база знань

ВСТУП

Медична інформаційна система (МІС) – комплексний програмний продукт, головне призначення якого автоматизація всіх основних процесів, пов’язаних із роботою медичних установ загальної і вузької спеціалізації. Автоматизовані медичні інформаційні системи дозволяють швидко й ефективно налагодити електронний документообіг, гнучко вибудовувати роботу з пацієнтами, вести оперативний облік роботи адміністративного персоналу, контролювати всі організаційні і фінансові питання.

Основні задачі які допомагає вирішувати медична інформаційна система – управління великими масивами даних про пацієнтів і результати діяльності медичної організації. Вся занесена в МІС інформація зберігається і доступна в будь-який час у будь-якій точці входу в систему. Таким чином уніфікується підхід до пацієнтів, а медична документація оформляється за одним зразком, МІС дозволяє створювати електронні структури для лікарень, їхніх філій і окремих кабінетів, об’єднувати кілька закладів у єдину електронну систему. Більшість МІС мають гнучкі алгоритми й інтуїтивно зрозумілі інструменти формування і ведення звітності. Вся інформація в МІС доступна для аналізу та обробки – це, по суті, величезний електронний архів. Система дозволяє надавати доступ до різних розділів різних груп користувачів (наприклад, підтримка окремого порталу для пацієнтів або внутрішнього порталу для лікарів із можливістю спілкування й обміну інформацією).

Актуальність дослідження полягає у тому що, на сьогодні в Україні достатньо немала кількість медичних інформаційних систем, але жодна з них не дає змогу якісно та швидко підтримувати людей с обмеженими можливостями, відкрите питання щодо підбору медичного персоналу для таких людей, розрахунок часу прибуття медичної сестри, зручний інтерфейс для користування.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Медичні інформаційні системи

За кордоном прийнято використовувати термін HIS (Hospital Information System) – госпітальна інформаційна система для комплексного управління всіма процесами медобслуговування, у тому числі юридичному аспекті. Доповненнями до неї можуть бути специфічні модулі, наприклад, RIS (Radiology Information System) – радіологічна інформаційна система або PACS (Picture Archiving and Communication System) – система збереження медичних зображень. Окремий вид МІС – лабораторні інформаційні системи (Laboratory Information Management Systems) й аптечні інформаційні системи (АІС). Вони можуть частково або повністю реалізовуватися у вигляді окремих компонентів комплексної медичної інформаційної системи [1].

Як правило, кожна МІС складається з блоків, що відповідають за автоматизацію різних складових діяльності медичної установи. Серед них:

– реєстратура та електронні медичні картки пацієнтів;
– дані медичних досліджень;
– робочі місця лікаря і медсестри;
– розподіл ресурсів установи, їхній розклад;
– управління фінансами й облік;
– адміністративна інформація і засоби комунікації співробітників;
– лікарські призначення, журнал призначень;
– стандарти надання медичної допомоги та багато іншого.

Основні функції медичної інформаційної системи (рис. 1.1).

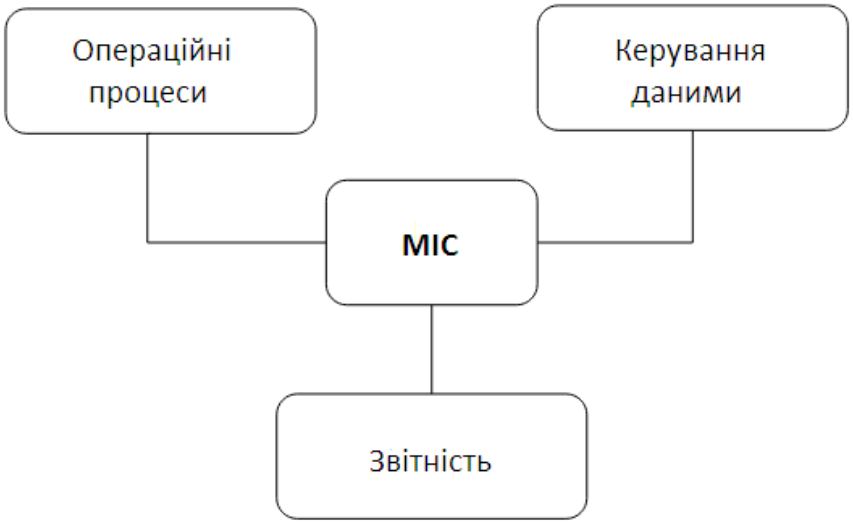

Рисунок 1.1 – Схема основних функцій МІС

Комплексні медичні інформаційні системи, як правило, складаються з модулів [2]. Це дозволяє зібрати і налаштувати МІС у потрібній конфігурації для установ різного типу і забезпечити необхідний функціонал із можливістю подальшого додавання/видалення модулів. Структура медичної інформаційної системи – окремі компоненти, що можна об’єднати у кілька великих груп:

Аналітичні й управлінські компоненти. Модулі та засоби ведення управлінського обліку, інструменти аналізу якості та ефективності медичних послуг. Ці складові МІС дозволяють проаналізувати стан вашої медичної організації, виявити проблемні місця й оптимізувати бізнес-процеси. На рівні користувача – пошук медичних записів по будь-яким критеріям, з урахуванням обмежень за рівнем доступу. Результати аналізу можна вивести на екран у вигляді графіків, таблиць або на друк.

Медичні компоненти. Усі модулі, пов’язані з реєстрацією пацієнтів, ведення реєстру електронних медичних карт, облік лікарняних листів, ведення протоколів лікування, інформаційний супровід лікування пацієнтів у

різних типах установ (амбулаторія, поліклініка, стаціонар), медична статистика й аналітика, історія хвороби та багато іншого.

Фінансово-економічні компоненти. До них відносяться інструменти ведення обліку медикаментів, управління запасами, розрахунок собівартості лікування і тарифів на надання медичних послуг, розрахунок надбавок лікарям, інструменти проведення економічного аналізу діяльності організації і т.д.

Компоненти обміну даними. Ведення уніфікованих реєстрів, каталогів і довідників, обмін даними в системі закладів охорони здоров'я, обробка отриманих даних [3].

Загально технічні компоненти. Контроль доступу користувачів і захист бази даних, а також підтримка можливостей інтеграції з іншими системами і програмами.

Якщо медична інформаційна система була вибрана вдало, її впровадження позитивно відобразиться на роботі організації. Але все буде залежати від виду МІС, її функціональних можливостей та специфіки конкретного мед закладу. Неможливо уявити наше життя без сучасної медицини – так які переваги використання МІС отримують різні учасники цього процесу? Користь для медичних закладів:

Позбавляє від заповнення паперів. Не потрібно дублювати записи і вносити інформацію в інші документи: лікарі і персонал, що мають доступ до карти пацієнта, можуть ознайомитися з історією хвороби, ходом лікування, результатами досліджень з єдиної бази даних.

Підвищення якості обслуговування і зниження впливу людського фактора. Автоматизований документообіг дозволяє зменшити кількість паперової роботи, успішно вести базу клієнтів, спираючись на актуальну інформацію про дослідження і надані послуги. Лікарям набагато простіше ставити діагноз, знижується ризик втрати важливих даних, як це часто буває з результатами аналізів у паперовому вигляді: їх можуть банально

прикріпити до чужої карті, через що доводиться проходити дослідження повторно [4].

Телемедицина. Лікарі отримують можливість проконсультуватися в реальному часі з колегами та іншими фахівцями стосовно правильності постановки діагнозу (особливо, в екстреній ситуації), призначення і корекції лікування, вести дистанційний моніторинг стану хворих і т.п.

Узгодженість роботи. Клінікам зручно мати онлайн-реєстратуру, вести загально лікарняні бази пацієнтів, розподіляти їх по філіях із урахуванням завантаженості і графіка фахівців, маючи при цьому можливість оцінити попит на ті чи інші послуги, а приватним центрам – формувати ціноутворення.

Завдяки МІС пацієнти отримують доступ до своїх даних, можуть оперативно отримувати результати лабораторних аналізів і відстежувати їх разом із лікарем, записуватися до нього на прийом, підтримувати зворотний зв’язок і т.д. Знижується ймовірність підробки і втрати медичних даних, адже пацієнт їх моніторить самостійно. Система попередньої онлайн-запису дозволяє уникнути черги в лікарні.

1.2 Огляд існуючих МІС

Medods – платформа для організації роботи приватних медичних центрів і стоматологій, а також мережі клінік від російських розробників. Програма доступна в локальній і хмарній версії (Saas), підтримує всі необхідні модулі і дозволяє записувати пацієнтів на прийом, вести їхні електронні карти, налаштувати онлайн-запис із сайту, автоматично формувати договору та інші документи, виставляти рахунки, проводити і відстежувати платежі, вести облік запасів, планувати маркетингові акції, email і SMS-розсилки, отримувати зведену статистику роботи і багато

іншого. Medods – приклад вдалого поєднання елементів CRM-системи з підтримкою розкладу і записи пацієнтів, і інструментами бізнес-аналітики.

Плюси. Важливі переваги Medods – наявність онлайн-запису, робочий стіл керівника, вбудована інтеграція з телефонією UIS, інтеграція з іншими телефони по API, наприклад, з Asterisk, підтримка маркетингових інструментів, інтеграція з 1С та інше. Технічний і клієнтський супроводи включено у вартість придбання [5]. Зручний, інтуїтивно зрозумілий інтерфейс.

Мінуси. Не підтримується багатофакторна авторизація, резервне копіювання в декількох місцях, обмежені можливості вбудованої інтеграції, сховища даних. Пробний доступ не надається.

Оплата. Можливо два способи придбання ліцензії: абонентське використання (у хмарі) або одноразове придбання ліцензії. Оплата впроваждується за передплатою.

MedElement – медична інформаційна система, розроблена в Казахстані. Поєднання хмарних сервісів і потужної довідкової системи для лікарів, студентів-медиків і всіх, кому важлива турбота про здоров’я. Сфера застосування MedElement – автоматизація роботи клінік, клінік ДРТ, стоматологій, аптек, блоків живлення, приватних медичних практик [6]. Цікавою особливістю даної МІС є те, що крім підтримки основних модулів, вона є потужною довідковою системою. Тут містяться довідники захворювань, медичних термінів, лабораторних показників, лікарських засобів, розміщуються огляди світової періодики та ін. Більш того, вона має переваги хмарної МІС: підтримується автоматизація всієї медичної документації, формування звітів, збір маркетингової інформації, облік фінансів, послуг.

Плюси. Зручні вебсервіси, наявність мобільного застосування для швидкого пошуку лікаря, записи на прийом і ведення комунікації. У програму закладено технологія допомоги у прийнятті клінічних рішень – MedElement пов’язана з онлайн-базою інтерактивних медичних довідників.

Мінуси. Не надто зручний інтерфейс, підтримка «всього і відразу» не завжди є плюсом, проте, це стосується усіх програм.

Оплата. Оплата за підключення до системи MedElement здійснюється за принципом абонентської плати за місяць, півроку, рік.

Clinic365 – більш спеціалізоване CRM-рішення для комерційних клінік, ніж комплексна медична інформаційна система. Може розвиватися як у хмарі, так і на сервері. Clinic365 включає базові функції для обліку пацієнтів, управління розкладом, контролю фінансових взаємин із пацієнтами. Щоб приступити до роботи в системі і забезпечити функціонування інших процесів, потрібно ввести вручну або імпортувати довідкову інформацію по таким блокам: співробітники та графіки роботи, ресурси, каталог послуг, «єдине вікно» пацієнта, картотека пацієнтів. Ключова особливість МІС Clinic365 – можливість побудувати гнучкий алгоритм роботи з пацієнтом. Підтримується інтеграція телефонії з Телфин, Oktell [7].

Плюси. Наявність інтегрованої CRM-системи, опції, що налаштовують роботу з електронними медичними картками пацієнта, підтримка IP-телефонії/контакт-центру, маркетингових інструментів, звіти й управління доступом.

Мінуси. Пробний доступ не надається. Відсутня багатофакторна авторизація, не підтримуються повідомлення клієнтів.

Оплата і вартість. Оплата за передплатою. Є два типи ліцензій: безстрокові та тимчасові. Вартість безстрокової ліцензії буде залежати від загальної кількості робочих місць, установка на сервер клініки проводиться безкоштовно, оплата за ліцензію одноразова. Тимчасові ліцензії оплачуються раз на рік [8].

Doctor Eleks – комплексне рішення, що дозволяє оптимізувати роботу клінік будь-якого розміру і профілю (приватних і державних). Розробник – компанія Eleks (Львів, Україна). Doctor Eleks підтримує електронну медичну карту пацієнта, інструменти редагування шаблонів документів, особистий кабінет лікаря, модуль реєстратури та роботи зі звітністю, фінансами,

персоналом. Підсистема розкладів дозволяє формувати графіки роботи співробітників з урахуванням побажань лікарів і пацієнтів. Лабораторна інформаційна система Doctor Eleks може використовуватися як окремий програмний продукт [9]. Серед додаткових можливостей – повноцінний редактор для обробки відео і зображень, що можна додавати в документи. Гнучка технологія побудови звітів, є можливість проводити аудит медичних документів, підтримується модуль PACS, Web-клієнт і багато іншого.

Плюси. Потужний функціонал, наявність комунікаційного сервера для обміну даними у форматі HL7 із суміжними ІС, зовнішніми лабораторіями, страховими компаніями. Присутній інтеграція з Toshiba УЗД, підтримується імпорт DICOM-зображень, підключення DICOM-сумісного обладнання та багато іншого. Doctor Eleks підключений до системи eHealth, система пройшла перевірку і рекомендована до використання МОЗ України [10].

Мінуси. Навіть якщо вони є (наприклад, не підтримуються електронні напрямки), то ці недоліки компенсуються іншими можливостями.

Оплата. Ліцензування ПЗ проводиться на основі постійних і тимчасових ліцензій. Послуги з впровадження, інтеграції, техпідтримка по завершенні впровадження оплачуються окремо, як і вартість використання мобільного додатка, вебпослуг і роботи зі страховими компаніями.

EMCiMED – передова українська медична інформаційна система для медичних установ, приватних клінік і лабораторій, підключена до системи eHealth України. Складається з модулів, що легко можна збирати в потрібній конфігурації для кожного окремого закладу. Основні підтримувані модулі: реєстратура, управління персоналом, управління організацією, поліклініка, стаціонар, лабораторія, управління партнерськими відносинами. Можна придбати додаткові модулі: облік послуг, управління запасами, архів медичних зображень PACS та інші. За необхідності вони можуть поставлятися в складі пакетів EMCiMED-Поліклініка і EMCiMED-Стаціонар.

Плюси. Можливість вибирати модулі відповідно до вимог організації, гнучке налаштування, потужна функціональна складова. Система захищена

завдяки використанню USB-брелків та шифрування всієї інформації, підтримує інтеграцію з іншими продуктами, наприклад, 1С. Пройшла перевірку і рекомендована до використання МОЗ України [11].

Мінуси. Якщо присутні, то істотно не впливають на роботу з програмою.

Оплата та вартість. Ліцензія. Ліцензії безстрокові, припускають одноразову оплату на весь період використання. Більш докладно всі умови обговорюються з замовником при оформленні договору поставки.

1.3 Розробка та вартість МІС

Індивідуальна розробка МІС – це, в основному, окремі випадки для комерційних клінік. Більшість мед закладів використовують готові програмні рішення, їхній вибір залежатиме від типу закладу та повсякденних завдань, котрі належить оптимізувати за допомогою МІС. Перш, ніж приступити до вибору конкретної системи, необхідно звернути увагу на два моменти: формат поставки ПЗ і ваші обмеження у бюджеті [12].

Отримати доступ до можливостей МІС можна трьома способами:

Обираємо готове програмне рішення і працюємо на базовому функціоналі, необхідному для виконання всіх поточних завдань. Бажано вибирати системи, розробники яких пропонують скористатися пробним доступом до всіх функцій – так можна зрозуміти, чи дійсно потрібно стільки модулів, або можна обійтися стандартними, що входять у базовий комплект поставки.

Раціонально і з урахуванням потреб мед установи. Йдеться про інтеграцію готового серверного рішення з подальшим «Допилюванням» його під конкретні потреби організації. На практиці користувачі можуть виявити, що звичні алгоритми їхніх дій або прийняття рішень розходяться з тими, що пропонує система. Набагато простіше найняти розробників і

переписати/дописати функціонал, ніж перебудовувати роботу всієї організації під конкретне ПЗ [13].

Індивідуально, під нестандартні завдання. Тобто, замовити систему з нуля. Цей варіант може підійти установам з унікальною специфікою роботи. Сучасні комплексні МІС, як правило, пропонують інтеграцію або мають вбудовані модулі, заточені під специфічні завдання (управління лабораторіями, стаціонар, поліклініка, блоки живлення і т.д.). Можливо, необхідно доповнити вже існуючу систему новими можливостями, наприклад, додати портал для навчання персоналу або підключити до блоку з укладання договорів сервіс розпізнавання документів? Може виявитися, що постачальник вашого ПЗ не передбачив наявності таких модулів, а підключення сторонніх програм до системи, що містить вразливі персональні дані – досить ризикована затія. Розробка індивідуальної МІС для оптимізації рутинних операцій – рішення, де потрібно зважити всі за і проти. Однак у результаті необхідно отримати те, що необхідно.

Усі МІС поставляються у двох варіантах: хмарні (Saas) платформи та стаціонарні (з установкою на сервер компанії) [14]. Використання хмарних сервісів знижує вартість програм, вся підтримка, забезпечення безпеки лежить «на совісті» розробників, а принцип роботи простий: встанови і користуйся. Високий рівень конкуренції і наявність подібних продуктів на ринку сприяють тому, що майже всі пропонують максимально зручний інтерфейс, навчання і підтримку, тестовий доступ до функціонала. Мінус – вони не завжди здатні охопити всі процеси організації.

Перевагою розгортання МІС на базі власного серверного обладнання буде те, що треті особи повністю виключаються з процесу збору та управління даними, набір сервісів розширюється шляхом придбання додаткових модулів, роблячи саму систему і її налаштування більш гнучкими.

Вартість розробки МІС залежатиме від її функціоналу, а також рівня складності [15]. Як правило, функціонал МІС будується навколо створення

так званих АРМів – автоматизованих робочих місць, що повторюються в декількох версіях: АРМ лікаря, АРМ реєстратора, АРМ лаборанта і т.д. Створення спеціалізованої системи також передбачає написання специфічного для медичної галузі коду. Відповідальна і трудомістка частина роботи над МІС – розробка механізмів, що забезпечують цілісність даних, підсистеми безпеки, адміністрування, підтримки та зв'язку з медичним обладнанням і т.д. Тому вартість кожного проекту буде оцінюватися і розраховуватися індивідуально.

З огляду на великі обсяги медичної інформації, з якими щодня доводиться працювати медичним співробітникам, використання МІС дозволяє істотно знизити непрофільне навантаження на лікарів і персонал, допомагає керівництву в управлінні, прийнятті рішень, налагодженні стосунків із клієнтами і партнерами.

В Україні використання МІС є однією зі складових сучасної електронної системи охорони здоров'я eHealth. МІС дають можливість автоматизувати роботу медичних установ із Центральної базою даних (ЦБД). Підключення та обмін даними з ЦБД можливі тільки за допомогою МІС. Повний перелік МІС, підключених до ЦБД eHealth, а також короткий огляд їхніх функціональних модулів можна подивитися тут. Варто зазначити, що українське законодавство дозволяє використання тільки тих систем, що пройшли перевірку МОЗ і були підключені до системи eHealth [16].

1.4 Класифікація МІС

Медична інформаційна система призначена для управління даними, які збираються та зберігаються в медичній установі. Це включає в себе лікарні, приватні та державні клініки, а також лікарні. Ці заклади збирають, зберігають, керують та надсилають електронні медичні записи пацієнтів. Цифрові системи охорони здоров'я покращують лікування пацієнтів, маючи

найновіші дані про пацієнтів. Дані пацієнтів дуже чутливі, тому будь-яка медична інформаційна система, що використовується, повинна забезпечувати точність зібраних даних і конфіденційність пацієнтів. Інші види використання даних пацієнтів, окрім індивідуального лікування клієнта, включають медичні дослідження, дані щодо формування політики, аналіз циклу доходів та інформацію про прийняття рішень. Інформаційні системи охорони здоров’я регулярно отримують доступ, обробляють або зберігають великі обсяги конфіденційних даних пацієнтів. В результаті безпека комп’ютерних систем має вирішальне значення [17].

Медичні інформаційні системи доступні для медичних працівників. До них належать ті, хто безпосередньо має справу з пацієнтами, клініцистами та посадовими особами охорони здоров’я. Медичні працівники збирають дані та збирають їх, щоб приймати рішення щодо охорони здоров’я для окремих пацієнтів, груп клієнтів та широкої громадськості. Інформаційні системи охорони здоров'я включають:

– електронна медична карта та електронна медична карта. Системи EHR замінюють паперові записи EMR та записи. Медична інформація про кожного пацієнта збирається та зберігається [18]. Ці записи включають інформацію про стан здоров’я відвідувачів, результати аналізів, лікарів та спеціалістів, медичне лікування. Багато медичних даних використання для хмарних сховищ для підвищення безпеки. Однак це не може бути варіантом для критичного доступу лікаря, який має проблеми з основними системами;

– програмне забезпечення управління практикою. Інформаційні системи допомагають закладам охорони здоров’я та персоналу керуючою щоденною роботою закладу. Це включає планування прийому та виставлення рахунків для медичних послуг. Незалежно від їх розміру, від лікарів з однієї лікарем до величезних багатоцентрових лікарів, усі постачальники медичних послуг використовують системи управління практикою. Мета – автоматизувати адміністративні завдання, що виконуються в рамках ведення бізнесу на об’єкті;

– головний індекс. Програмне інформаційне забезпечення охорони здоров’я включає в себе записи з більшою системою базових даних. MPI містить записи для будь-якого хворого, зареєстровані в організації охорони здоров’я [19]. MPI створює індекс для всіх записів для цього пацієнта. MPI зменшує кількість дублікатів записів та неточної кількості інформації про пацієнтів, що може призвести до відмови від лікування у заявах;

– портали. Ця інформаційна система дає пацієнтам переглядати стан про своє здоров’я. Через Інтернет вони можуть отримати доступ до інформації про прийом, ліків, які вони можуть отримати, та результати лабораторних досліджень. Деякі портали спілкуються також сприяють активному зв’язку з медичними працівниками, включаючи лікарів, фармацевтів, щодо їх запитів на доповнення рецептів та планування прийомів;

– відданий моніторинг лікаря (RPM). Також повідомляє telehealth надає медичні працівники, які можуть передавати дані RPM як медичні працівники. RPM може контролювати рівень глюкози в крові та артеріальний тиск. Це корисно для серцево-судинної системи із такими особливими захворюваннями, як цукровий діабет 2 типу, гіпертонії або захворювання. Дані, зібрані та передані через RPM, можуть бути медичним працівником або командою охорони здоров’я для виявлення медичних подій, як інсульт або серцевий напад, які потребують потреби та агресивного медичного засобу. Зібрані можуть бути використані як частина дослідницького проекту або дослідження здоров’я. RPM – яка рятує це системи життя у віддалених районах, які не можуть отримати медичну допомогу вічна-віч;

– підтримка клінічних рішень (CDS). CDS аналізує клінічні та адміністративні системи. Мета – допомога медичним працівникам обґрунтовують клінічні рішення. Доступні можуть надати інформацію медичним фахівцям, які знають або прогнозують медичні стани, як-от взаємодія та реакція на ліки. Інструменти CDS фільтрують інформацію, щоб допомогти медичним працівникам доглядати за окремими клієнтами;

– лабораторна інформаційна система (LIS). Програмне забезпечення LIS дозволяє лікарям і лаборантам координувати стаціонарні та амбулаторні дослідження з мікробіології, гематології, хімії та імунології для отримання клінічних даних. Стандартна інформаційна система для лабораторій дає демографічні показники керуючих, відображення даних про реєстрацію та обробку зразків, а також результати [20].

Ще один найпоширеніший поділ – це поділ на клінічні та адміністративні системи. Однак, якщо замислитись, то в принципі неможливо розробити будь-яку клінічну систему без того, щоб вона не залежала від будь-якого типу адміністративних даних. Наприклад, найбільш базова з клінічних систем має дозволяти готувати листи лікарям загальної практики або пацієнтам для подальшого спостереження. для цього потрібна інформація про лікаря загальної практики та адресу. Питання, чи є такі дані адміністративною чи клінічною інформацією? Якщо розглядати спрощено, то ядро інтегрованої клінічної інформаційної системи лікарні є нічим іншим, як головний індекс, що складається з основних даних про пацієнта (адміністративна інформація), що забезпечує зв’язок з різними клінічних систем [21]. Потім кожна відомча клінічна система дозволяє окремим особам створювати додаткові дослідні набори даних для конкретної діяльності. Можна стверджувати, кожна клінічна система містить електронну карту пацієнта чи віртуальне об'єднання кожної з них для конкретного пацієнта є ЕПР [22]. Усі лікарні у Великій Британії мають систему PAS (Patient Administration System) для надання даних у вигляді звітів HES (Hospital Hospital Episode Statistics) звітів до Міністерства охорони здоров’я [23]. Це ймовірно, визначається як адміністративна система оскільки вона була розроблена для забезпечення можливості ретроспективного введення даних введення даних (тобто інформація про пацієнта зазвичай вводилася вводилася після виписки, коли записи потрапляли до медичні карти) та надавала детальну інформацію про кожного епізоду лікування. Тим не менш, цікаво відзначити, що звіти, з мінімальними змінами були називалися

наборами даних за контрактами. Подібним чином набір даних також містить інформацію про діагноз, процедури та результати, всі з яких можуть бути класифікуються як клінічні.

1.5 Постановка задачі дослідження

На сьогодні усі існуючі в Україні чимало МІС направлені на допомогу лікарям та хворим [24]. МІС для комунікації та допомоги пацієнтам з обмеженими можливостями наразі є невирішеним питанням не тільки у нашій країні але і у багатьох країнах світу. Задачі підбору лікаря, або сиділки для інвалідів.

Тобто, зараз є багато відкритих задач щодо створення та вдосконалення МІС для допомоги пацієнтам з обмеженими можливостями. Для якісної та швидкої комунікації пацієнта з лікарем необхідна наявність МІС, яка направлена не тільки на допомогу лікарям, а також й на вирішення проблем самого пацієнта с обмеженими можливостями. Тому створення інформаційної системи для підбору медичного персоналу людям с обмеженими можливостями є дуже актуальною.

Об’єктом дослідження є автоматизація процесу надання допомоги пацієнтам з обмеженими можливостями.

Метою дослідження є розробка компоненту медичної інформаційної системи для вирішення частини питань людей з обмеженими можливостями.

Для цього необхідно вирішити такі завдання:

- моделювання компонентів системи;
- проектування схеми бази даних;
- моделювання діаграм;
- компонент онлайн зв’язку із лікарем;
- компонент виклику медсестри.

Результатом роботи є компоненти медичної інформаційної системи для підтримки людей с обмеженими можливостями, підбір необхідного медичного персоналу за заданими критеріями та виклик медсестри або сиділки за місцезнаходженням користувача.

2 МОДЕЛЮВАННЯ МІС ПІДБОРУ ПЕРСОНАЛУ ДЛЯ ЛЮДЕЙ З ОБМЕЖЕНИМИ МОЖЛИВОСТЯМИ

Компонент МІС призначений для автоматизації пошуку та підбору медичного персоналу для людей з обмеженими можливостями.

При проектуванні компоненту медичної інформаційної системи необхідно провести аналіз цілей цієї системи і виявити вимоги до неї користувачів. Інформація для побудови моделі інформаційної системи береться на основі проведення всебічного обстеження організації, для якої виконується розробка інформаційної системи. Збір даних починається з вивчення сутностей предметної області, процесів, що використовують ці сутності, опису схеми даних.

2.1 Концепція компонентів МІС

Основна концепція будь-якої інформаційної системи полягає в її поетапному проектуванні та моделюванні кожного компонента системи. Конпепція базується на функціональних вимогах та бізнес вимогах системи.

2.1.1 Основні поняття

Глосарій предметної області медичної інформаційної системи підбору медичного персоналу (табл. 2.1).

Таблиця 2.1 – Глосарій МІС для підбору персоналу

Термін	Визначення
Користувач	Людина котра зареєструвалась у МІС.
Лікарня	Медичний заклад який містить доступних лікарів.

Продовження таблиці 2.1

Термін	Визначення
Лікар	Людина з медичною освітою зареєстрована у МІС
Сиділка	Людина з медичною освітою зареєстрована у системі з ціллю допомоги інвалідам
База даних	Сукупність даних, організованих відповідно до концепції МІС
МІС	Медична інформаційна система, представляє собою мобільний додаток для пошуку лікаря та виклику швидкої допомоги
Місцезнаходження	Географічне розташування користувача для виклику сиділки або медичної сестри
Інвалід	Користувач МІС з обмеженими можливостями

2.1.2 Функціональні та бізнес вимоги МІС

Для формування функціональних та бізнес вимог до розроблюваного компоненту медичної інформаційної систерилми підбору медичного персоналу для людей з обмеженими можливостями необхідно побудувати діаграму прецедентів, згідно стандартів мови уніфікованого моделювання (Unified Modeling Language UML) [25].

Розробка функціональних та бізнес вимог планується на етапі описування бізнес процесів розроблюваної медичної інформаційної системи та етапів розробки програмних компонентів додатку системи. На рисунку 2.1 зображена діаграма прецедентів до функціональних вимог компоненту інформаційної системи.

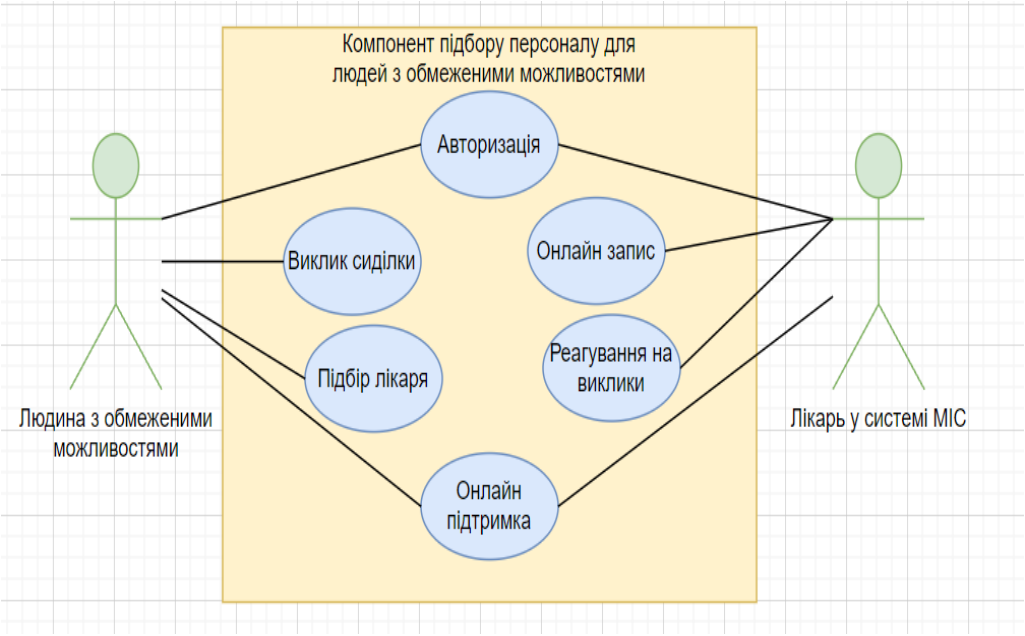

Рисунок 2.1 – Діаграма прецедентів до вимог MIC

Прецедентами діаграми являються поставлені функціональні та бізнес задачі до компоненту MIC підбору медичного персоналу для людей з обмеженими можливостями. Безпосередня робота системи та її основні функції, описані на діаграмі, базується на взаємодії інваліда з MIC та лікаря або медичної сестри/сиділки самої системи. Відображені ті аспекти, які в користувач з обмеженими можливостями зможе реалізувати за допомогою мобільного програмного забезпечення.

2.1.3 Контекстна модель компонентів MIC

При виконанні процесів функцій компоненту MIC здійснюється взаємодія з іншими процесами і з зовнішнім середовищем по входу, виходу, управління і механізмів та даних. Для опису концептуальної моделі необхідно побудувати модель IDEF0 згідно стандартів.

Діаграма IDEF0 представляє інтегровану картину входів, керування, виходів і механізмів (ICOM) для декомпозиції функції. Діаграма IDEF0, яка є частиною набору подання поведінкової (логічної архітектури), відображає

велику кількість контекстної інформації про взаємозв’язки декомпозиції без відображення фактичної логіки управління/структури декомпозиції. Спочатку визначена стандартом FIPS-183 Національного інституту стандартів і технологій (NIST), діаграма IDEF0 використовується рідше, ніж інші поведінкові уявлення, але все ще є цінною частиною інтегрованого набору представлень.

На діаграмі IDEF0 підфункції показані на головній діагоналі. Порядок функцій автоматично визначається CORE, який проходить через базову структуру батьківської функції (як показано графічно на діаграмі діяльності або EFFBD). Структура проходить зліва направо паралельно, а вибрані конструкції переходять по одній гілці за раз. Для кожного функціонального вузла: Входи вводяться зліва. Вони можуть виходити з краю діаграми (зовнішні входи) або з іншої функції на схемі. Елементи керування (дані запуску) вводяться зверху. Вони можуть виходити з краю діаграми (зовнішні тригери) або з іншої функції на діаграмі. Вихід праворуч. Виходи можуть або підключатися до іншої функції на схемі, виходити на край діаграми, або обидва (представляючи вихід, який є вхідним для / запускає як внутрішні, так і зовнішні функції). Механізми (розподіл компонентів) входять знизу (рис. 2.2).

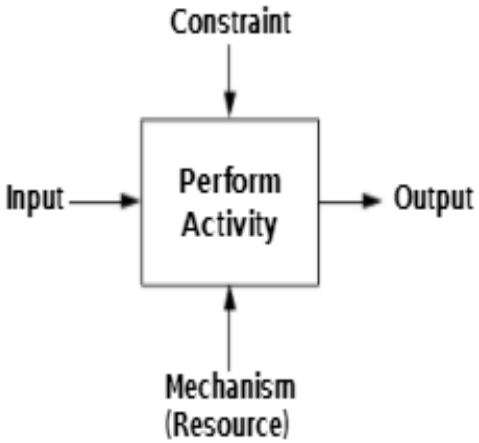

Рисунок 2.2 – Структура діаграми IDEF0

Процес моделювання будь-якої інформаційної системи в IDEF0 починається з визначення основого контексту, абстрактного рівня опису системи в цілому. У контекст входить визначення суб’єкта моделювання, цілі і точки зору на модель.

Вхідними даними будуть дані людини з обмеженими можливостями, географічне розташування мед закладів та саме користувача.

Механізмами здійснення процесу є компоненти обробки даних МІС, медичний персонал зареєстрований у МІС, користувач зареєстрований у системі.

На рисунку 2.3 зображена контекстна діаграма, що описує роботу компоненту МІС підбору медичного персоналу для людей з обмеженими можливостями. На цій діаграмі дається загальне уявлення про роботу, а також взаємозв’язок із зовнішнім середовищем або іншими процесами.

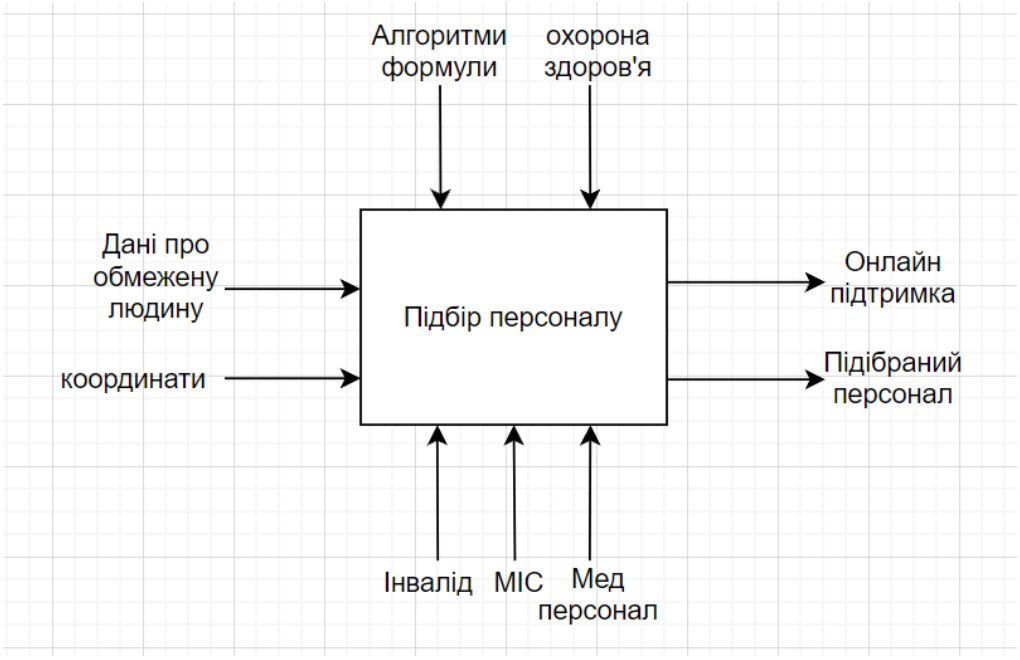

Рисунок 2.3 – Контекстна діаграма IDEF0 підбору персоналу

2.1.4 Склад та опис даних системи

Для опису сутностей даних необхідно побудувати таблицю з описами полів які буде мати кожна сутність компоненту МІС. В якості бази даних системи використовується нереляційна база MongoDB, тому сутності представлені у вигляді JSON – документа згідно стандартів бази даних (табл. 2.2).

Таблиця 2.2 – Опис Json-сутностей бази даних

Сутність	Опис сутності	Склад полів
Інвалід	Документ сутності користувача з обмеженими можливостями	– ФІО – Пошта – Пароль – Телефон – Діагноз – Адреса – Група інвалідності
Лікар	Сутність лікар відображення зареєстрованого у МІС лікаря який готовий надати онлайн допомогу користувачу з обмеженими можливостями	– ФІО – Спеціалізація – Досвід – Лікарня – Телефон – Пошта – Рейтинг
Лікарня	Локація відображає реальну зареєстровану лікарню у МІС	– Назва – Адреса
Сиділка	Документ сутності сиділка описує людину яка виїзжає на виклик до людини інваліда	– ФІО – Адреса – Телефон

Продовження таблиці 2.2

Сутність	Опис сутності	Склад полів
Спеціалізація	Сутність відображає доступні спеціалізації лікарів у МІС	– Назва
Реанімаційна бригада	Сутність описує зареєстрованих у МІС бригад невідкладної медичної допомоги для користувача	– Назва бригади – Лікарня
Локація	Сутність яка зберігає географічні координати(довготу та широту) для зареєстрованих лікарень МІС	– Довгота – Широта – Назва

У таблиці 2.2 описана повна модель зберігання даних у системі МІС та їх детальний опис і склад. Для кожної сутності необхідно створити колекції які будуть відповідати за збереження відповідних даних.

У колекції може зберігатися ряд документів. Колекція аналогічна таблиці RDBMS.

Колекція може зберігати документи тих, хто за структурою не однаковий. Це можливо, тому що MongoDB – це база даних без схем та зв’язків між сутностями, що дає змогу робити структуру більш гибкою та незалежною [26].

2.1.5 Модель функціональних можливостей МІС

Моделювання можливостей системи є важливим етапом у проектуванні будь якої інформаційної системи. Для МІС підбору персоналу людям з обмеженими можливостями необхідно побудувати схему функціональних можливостей для наступних вимог:

– онлайн підтримка обмеженого пацієнта;

– розрахунок часу прибуття медсестри або сиділки;

– аналіз місцезнаходження пацієнта;

– можливість зв’язку із лікарем.

На рисунку 2.4 зображена схема функціональних можливостей розроблюваного компоненту МІС для людей з обмеженими можливостями.

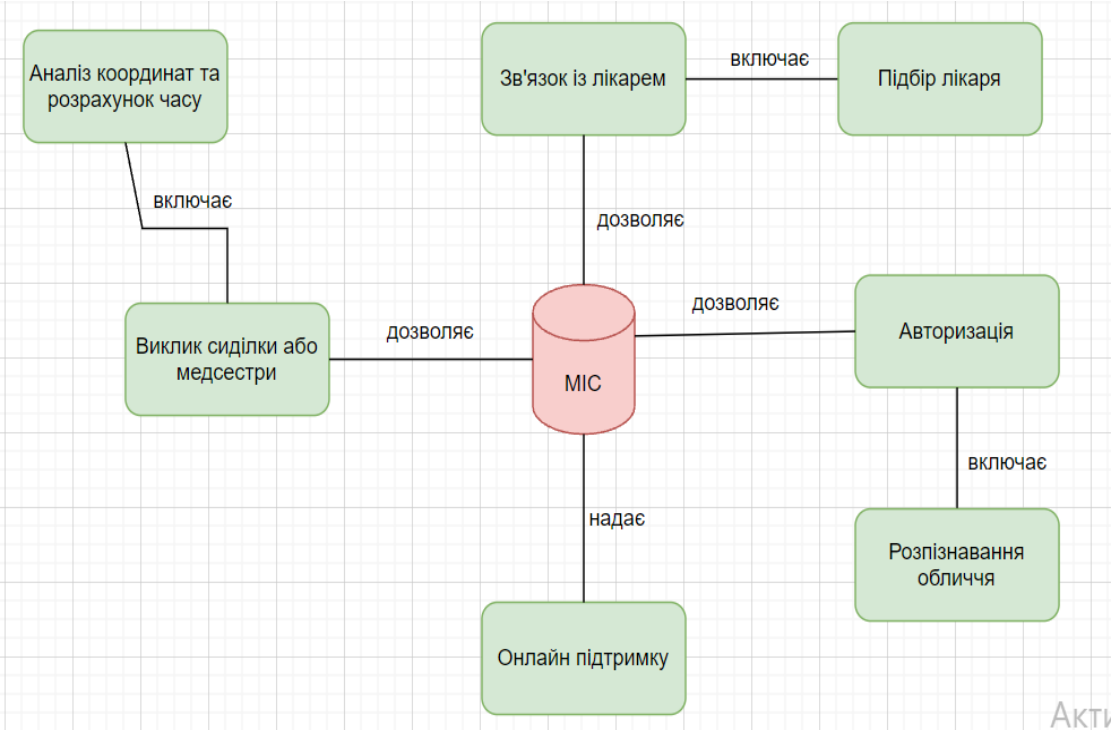

Рисунок 2.4 – Схема функціональних можливостей МІС

2.2 Моделювання компонентів системи

МІС підбору медичного персоналу для людей з обмеженими можливостями включає:

– компонент виклику сиділки або медсестри до інваліду;

– компонент онлайн підтримки та зв’язку із лікарем.

Для опису компонентів системи необхідно використовувати діаграми декомпозиції IDEF3, блок-схеми алгоритмів компонентів.

Метод захоплення опису процесу IDEF3 описує послідовності діяльність. Його головна мета – надати структурований метод опису функціонування певної системи чи організації. IDEF3 був розроблений для:

– збирання, описування, зберігання, керування та повторного використовування інформаційного процесу;

– використовування в інженерії, виробництві, логістиці, бізнес-системах і навіть районах урядових операцій;

– змоделювання в дрібно та великомасштабних системах як на високому абстрактному, так і на високому рівні;

– інтегрування з іншими методами IDEF;

– бути легким у навчанні та використанні зацікавленими сторонами.

Перевагами IDEF3 є економія витрат, скорочення часу, підвищення якості, підвищення організаційних можливостей і постійне вдосконалення організаційний механізм і бізнес-процес [27].

IDEF3 звик:

– визначення бізнес-процесів в різних сферах;

– надання незалежної від реалізації специфікації для людської системи взаємодій;

– визначення управління бізнес-процесами та управління змінами;

– задокументованих процедур прийняття рішень підприємства чи уряду;

– бути корисним інструментам підтримки інтерв'ю;

– розробляти програмне забезпечення для керування в режимі реального часу, забезпечуючи чіткий механізм визначити факти, моменти прийняття рішень і класифікацію посад;

– визначати поведінку систем і програм керування робочими процесами.

2.2.1 Компонент виклику медсестри

Компонент виклику медсестри або сиділки для інваліду є одним з найважливіших компонентів даної МІС. Він повинен розраховувати час прибуття медсестри до пацієнта за рахунок розрахунків відстані між їх географічним положенням.

Для розрахунку відстані між географічними точками координат необхідно використати формулу гаверсинусів з модифікацією для антиподів:

$$d = 2r \arcsin(\sqrt{\sin^2(\frac{\varphi 2 - \varphi 1}{2}) + (\varphi 1)\cos(\varphi 2)\sin^2(\frac{\lambda 2 - \lambda 1}{2})}),$$

де $\varphi 1, \varphi 2$ – широта першої та другої точки у радіанах;

$\lambda 1, \lambda 2$ – довгота першої та другої точки у радіанах;

d – відстань.

Компонент виклику медсестри має опрацьовувати вхідні географічні координати користувача. Наступним етапом є розрахунок приблизного часу прибуття найближчої медсестри або сиділки до початкових координат за допомогою формули гаверсинусів з модифікацією антиподів. Після отримання необхідних розрахунків, інформація передається користувачу.

Для описання роботи компоненту виклику медсестри, необхідно побудувати діаграму декомпозиції рівня IDEF3 (рис. 2.5).

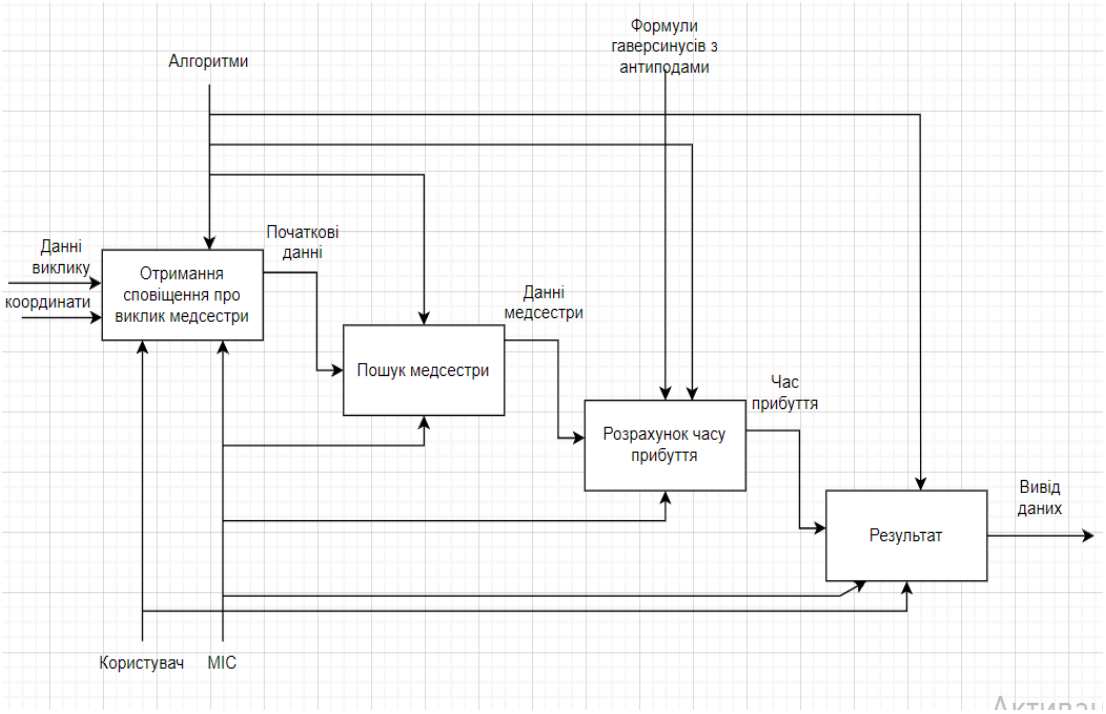

Рисунок 2.5 – Діаграма декомпозицій компоненту виклику медсестри

2.2.2 Компонент онлайн підтримки

Компонент онлайн підтримки пацієнта дозволяє отримати зв’язок із лікарем не виходячи з дому, що є дуже важливим для людей з обмеженими можливостями [28].

Для описання роботи компоненту онлайн підтримки пацієнта, необхідно побудувати блок-схему (рис. 2.6).

Блок-схема – це схема, яка зображує алгоритм процесу, системи або комп’ютера. Вони широко використовуються в багатьох сферах для документування, вивчення, планування, вдосконалення та спілкування часто складних процесів за чіткими, зрозумілими діаграмами. Блок-схеми, іноді написані як діаграми потоку, використовують прямокутники, овали, алмази та потенційно численні інші фігури для визначення типу кроку, а також сполучні стрілки для визначення потоку та послідовності. Вони можуть відрізнятися одна від одної, починаючи від простих, намальованих вручну

діаграм до великих та схематичних комп’ютерних діаграм, що зображують кілька ітерацій чи циклів.

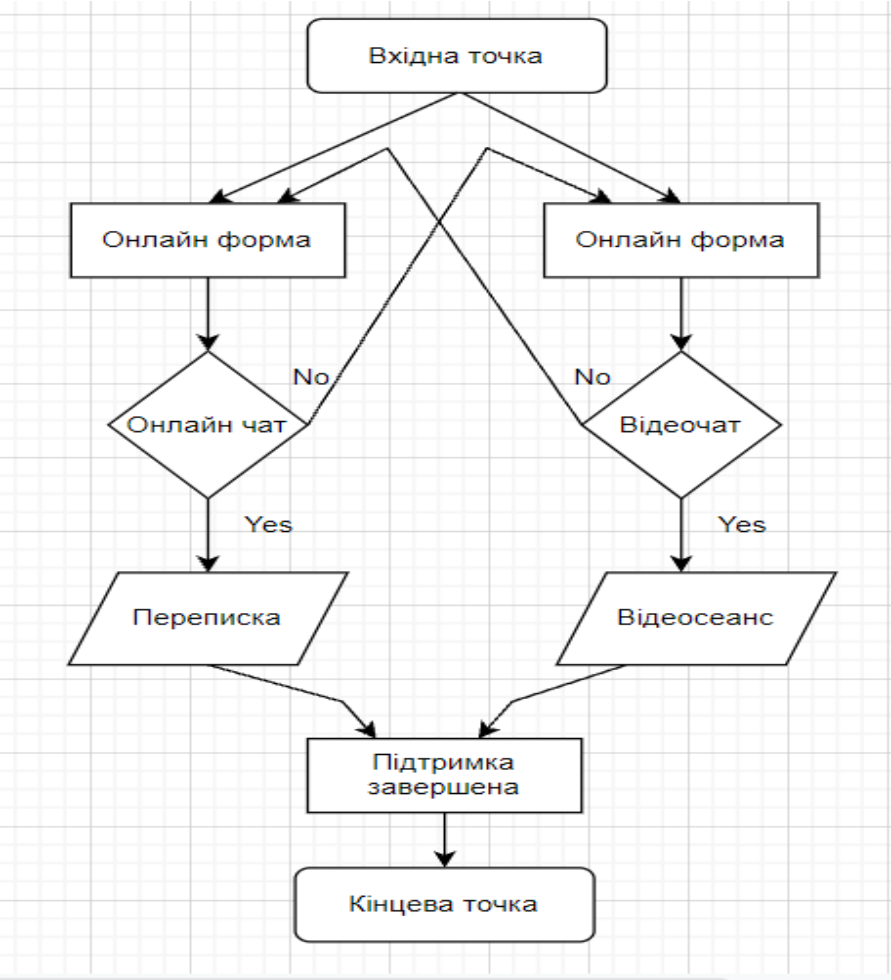

Рисунок 2.6 – Блок-схема компоненту онлайн підтримки

Блок-схема описує послідовність роботи компоненту онлайн зв’язку з лікарем, що дозволяє пацієнту з обмеженими можливостями отримати контакт с персоналом не виходячи із дому за допомогою переписки з лікарем або ж онлайн сеансу з ним.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ КОМПОНЕНТУ МЕДИЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Технології програмної реалізації

Для гнучкого зв’язку даних клієнта та back-end частини компононенту було обрану нову технологію GraphQl.

GraphQL – це мова запитів і середовище виконання на стороні сервера для інтерфейсів програмування прикладних програм (API), які надають клієнтам саме ті дані, які вони запитують, і не більше того. GraphQL розроблено, щоб зробити API швидкими, гнучкими та зручними для розробників. Його навіть можна розгорнути в інтегрованому середовищі розробки (IDE), відомому як GraphiQL. Як альтернатива REST, GraphQL дозволяє розробникам створювати запити, які витягують дані з кількох джерел даних за один виклик API. Крім того, GraphQL дає супроводжувачам API гнучкість додавати або забороняти поля, не впливаючи на існуючі запити. Розробники можуть створювати API за допомогою будь-яких методів, які вони віддають перевагу, а специфікація GraphQL забезпечить їх передбачувану роботу для клієнтів.

Розробники API використовують GraphQL для створення схеми для опису всіх можливих даних, які клієнти можуть запитувати через цю службу. Схема GraphQL складається з типів об’єктів, які визначають, який тип об’єкта ви можете запитати та які поля він має. Коли надходять запити, GraphQL перевіряє запити на відповідність схемі. Потім GraphQL виконує перевірені запити. Розробник API приєднує кожне поле в схемі до функції, яка називається резольвером. Під час виконання викликається резольвер для отримання значення. Окрім визначення та перевірки синтаксису для запитів API (викладеного в репозиторії graphql-spec), GraphQL залишає більшість інших рішень на розробку API. GraphQL не надає жодних вказівок щодо

того, як зберігати дані або яку мову програмування використовувати – розробники можуть використовувати PHP (graphql-php), Scala (Sangria), Python (Graphene Python), Ruby (graphql-ruby), JavaScript (graphql. js) та багато іншого. GraphQL не пропонує жодних вимог до мережі, авторизації або розбиття на сторінки. З точки зору клієнта, найпоширенішими операціями GraphQL, ймовірно, є запити та мутації. Якби думати про них з точки зору моделі створення, читання, оновлення та видалення (CRUD), запит був би еквівалентним прочитанню. Усі інші (створювати, оновлювати та видаляти) обробляються мутаціями.

В якості мови програмування було обрано Java версії 11.

Java – це мова програмування, розроблена для одночасного використання, на основі класів та об’єктно-орієнтованої, а також обчислювальної платформи, вперше випущеної Sun Microsystems у 1995 році. Величезна кількість програм і вебсайтів не працюватимуть, якщо у вас не встановлено Java, і щодня створюється більше. Відмовляти собі в Java – це те саме, що забороняти собі доступ до технологічної інфраструктури. Java рекламується і цінується за швидку продуктивність, безпеку та надійність.

Jenkins – це інструмент автоматизації з відкритим вихідним кодом, написаний на Java з плагінами, створеними для цілей безперервної інтеграції. Jenkins використовується для створення та постійного тестування ваших програмних проектів, що полегшує розробникам інтеграцію змін у проект і полегшує користувачам отримання нової збірки. Це також дозволяє вам постійно постачати програмне забезпечення шляхом інтеграції з великою кількістю технологій тестування та розгортання. Завдяки Jenkins організації можуть прискорити процес розробки програмного забезпечення за допомогою автоматизації. Jenkins інтегрує всі процеси життєвого циклу розробки, включаючи збірку, документацію, тестування, пакування, стадію, розгортання, статичний аналіз та багато іншого. Дженкінс досягає безперервної інтеграції за допомогою плагінів. Плагіни дозволяють інтегрувати різні етапи DevOps. Якщо необхідно інтегрувати певний

інструмент, вам потрібно встановити плагіни для цього інструменту. Наприклад, Git, проект Maven 2, Amazon EC2, HTML publisher тощо.

Jenkins Pipeline має настроювану та масштабовану систему автоматизації, яка дозволяє створювати сценарії конвеєра розповсюдження – також названі «Pipeline as Code». Написані як звичайний текст мовою Groovy Domain Specific Language (DSL), ці сценарії називаються JenkinsFile, і їх напрочуд легко написати та зрозуміти. JenkinsFile зберігає весь процес CI/CD як код на локальній машині. Таким чином, файл можна переглянути та перевірити на платформі керування вихідним кодом (SCM) (будь то Git чи SVN) разом із вашим кодом. Звідси термін «конвеєр як код». Існує два різних типи синтаксису, які можна використовувати для побудови конвеєра Дженкінса, кожен із яких кардинально відрізняється за підходом від іншого: декларативний та сценарний. Скриптовий синтаксис конвеєра За допомогою конвеєра Scripted JenkinsFile записується в екземплярі інтерфейсу користувача Jenkins. Потім він запускається на майстрі Дженкінса за допомогою виконавця (агента). Код визначається всередині блоків вузлів, тому для перетворення сценарного конвеєра в окремі команди потрібно відносно небагато ресурсів. Декларативний синтаксис конвеєра Декларативний конвеєр – це новий підхід до створення конвеєрів, що дозволяє читати та записувати код конвеєра. Оскільки він містить заздалегідь визначену ієрархію для проектування конвеєрів, він полегшує організацію та контроль усіх аспектів виконання конвеєра. За допомогою декларативного синтаксису конвеєра код записується в JenkinsFile у блоках конвеєра, які потім можна перевірити в SCM за вашим вибором.

В якості контролю версій додатку було обрано GIT.

Git є найбільш часто використовуваною системою контролю версій. Git відстежує зміни, які вносяться у файли, тож у вас є запис про те, що було зроблено, і можна повернутися до певних версій, якщо вам знадобиться. Git також спрощує співпрацю, дозволяючи об’єднувати зміни, внесені кількома людьми, в одне джерело.

Репозиторій Git (або скорочено репозиторій) містить усі файли проекту та всю історію редакцій. Звичайну папку з файлами (наприклад, кореневу папку вебсайту) і скажете Git зробити її сховищем. Це створює підпапку .git, яка містить усі метадані Git для відстеження змін. В операційних системах на базі Unix, таких як macOS, файли та папки, які починаються з крапки, приховані, тому не можливо побачити папку .git у Finder macOS, якщо не відобразите приховані файли, але вона є. Необхідно побачити це в деяких редакторах коду. На рисунку 3.1 зображено принцип роботи Git [28].

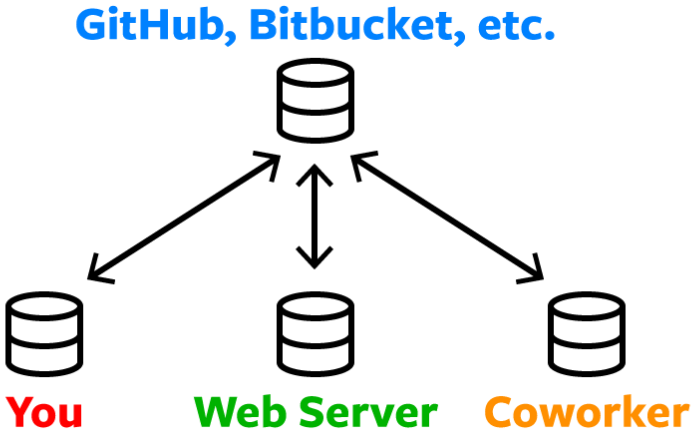

Рисунок 3.1 – принцип роботи Git

Pull-запити – це спосіб обговорення змін перед об’єднанням їх у свою кодову базу. Скажімо, керування проектом. Розробник вносить зміни в нову гілку і хоче об’єднати цю гілку з головною. Вони можуть створити запит на витяг, щоб сповістити вас про перегляд їх коду. Можна обговорити зміни та вирішити, чи необхідно їх об’єднати чи ні.

Для розгортання застосунку було обрано контейнер Docker.

Docker – це програмна платформа для створення програм на основі контейнерів – невеликих і легких середовищ виконання, які спільно використовують ядро операційної системи, але в інших випадках працюють ізольовано один від одного. Хоча контейнери використовувалися в системах Linux і Unix протягом певного часу, Docker, проект із відкритим кодом, запущений у 2013 році, допоміг популяризувати цю технологію, полегшивши

розробникам, ніж будь-коли, упакувати своє програмне забезпечення, щоб будувати один раз і запускати де завгодно.

Запуск додатків у контейнерах дає ряд переваг:

– портативність. Після того, як протестовано контейнеризовану програму, можно розгорнути її в будь-якій іншій системі, де працює Docker, і можно бути впевненим, що програма працюватиме так само, як і під час тестування;

– продуктивність. Хоча віртуальні машини є альтернативою контейнерам, той факт, що контейнери не містять операційної системи (тоді як віртуальні машини містять), означає, що контейнери мають набагато менші розміри, ніж віртуальні машини, їх швидше створювати та швидше запускатиє;

– спритність. Переваги в продуктивності, які пропонують контейнери, можуть допомогти зробити процес розробки більш гнучким і чуйним. Покращення безперервної інтеграції та безперервної доставки, щоб скористатися перевагами контейнерів і технологій, таких як Enterprise Developer Build Tools для Windows, полегшує доставку потрібного програмного забезпечення в потрібний час. Інструменти збірки Enterprise Developer для Windows це компонент Enterprise Developer, який надає всі функції Enterprise Developer, щоб можна було компілювати, створювати та тестувати код COBOL, але без накладних витрат на IDE. Ізоляція Контейнер Docker, який містить одну з ваших програм, також містить відповідні версії будь-якого допоміжного програмного забезпечення, яке вимагає ваша програма. Якщо інші контейнери Docker містять програми, які вимагають різних версій одного і того ж допоміжного програмного забезпечення, це не проблема, оскільки різні контейнери Docker повністю незалежні один від одного. Це також означає, що, проходячи через різні етапи свого життєвого циклу розробки, можна бути впевнені, що зображення, яке створиться під час розробки, буде працювати точно так само, як воно проходить тестування та потенційно для користувачів;

– масштабованість. Можна швидко створювати нові контейнери, якщо їх вимагає попит на програми. Використовуючи кілька контейнерів, можна скористатися різними параметрами керування контейнерами.

3.2 Мікросервісна архітектура back-end

Для написання back-end сервісу була обрана мікросервісна архітектура. Принцип її дії – це розбиття програмного коду на сервіси, де кожен сервіс має свій функціонал та роль у системі.

Мікросервіси – також відомі як архітектура мікросервісів – це архітектурний стиль, який структурує додаток як набір служб, які:

– високі до ремонтопридатності і тестування;

– самостійно розгортуються;

– організовані навколо можливостей бізнесу.

Мікросервісна архітектура забезпечує швидку, часту та надійну доставку великих складних програм. Це також дозволяє організації розвивати свій набір технологій.

На рисунку 3.2 зображено основну мікросервісну архітектуру.

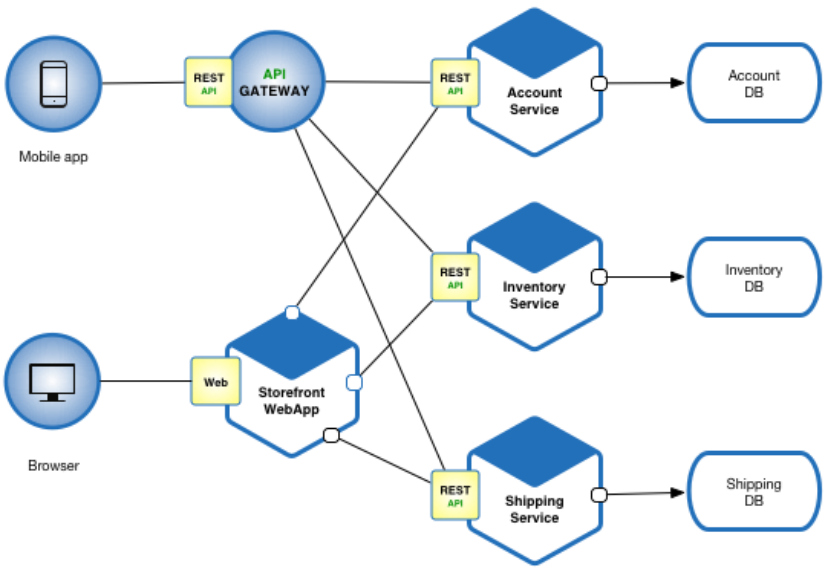

Рисунок 3.2 – Основна мікросервісна архітектура

На рисунку 3.3 зображена мікросервісна архітетура back-end частини компоненту MIC для людей з обмеженими можливостями.

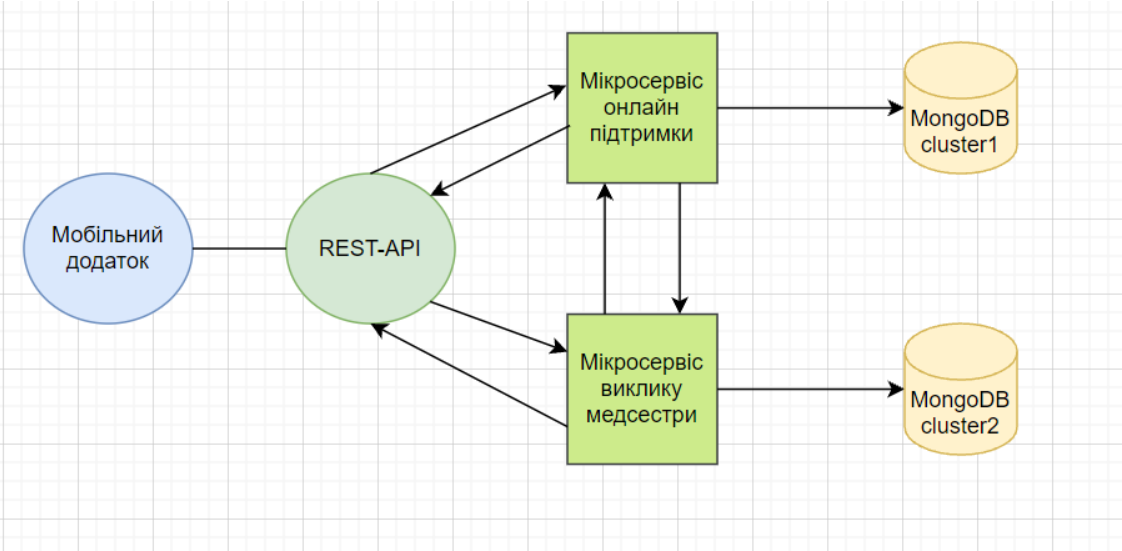

Рисунок 3.3 – Основна мікросервісна архітектура

Два мікросервіси які отримують данні з REST-API мають нелазежні між собою кластери бази даних. У такому наборі архітектури кожний сервіс має свою логіку та відповідає за свою частину функціоналу.

Для розробки повноцінного REST-API було обрано фреймворк Spring Boot. Перш за все, він використовує Java, яка є однією з найпопулярніших мов програмування у світі.

Spring Boot – це дивовижний інструмент, який допомагає швидко запускати і запускати програми корпоративного рівня, не турбуючись про правильне та безпечне налаштування програми. Крім того, спільнота користувачів величезна. Якщо вам потрібні безкоштовні навчальні матеріали та курси, можна знайти їх багато. Доступність освіти мала великий вплив на популярність фреймворка. Деякі додаткові переваги включають:

– скорочує час розробки та підвищує загальну продуктивність команди розробників;

– допомагає автоматично налаштувати всі компоненти для виробничої програми Spring;

– полегшує розробникам створення та тестування програм на основі Java, надаючи налаштування за замовчуванням для модульних та інтеграційних тестів;

– уникає написання великої кількості шаблонного коду, анотацій та конфігурації XML;

– постачається з вбудованими серверами HTTP, такими як Tomcat або Jetty, для тестування вебзастосунків;

– додає багато плагінів, які розробники можуть використовувати для легкої роботи з вбудованими базами даних і базами даних у пам’яті;

– spring дозволяє легко підключатися до бази даних і служб черги, таких як Oracle, PostgreSQL, MySQL, MongoDB, Redis, Solr, ElasticSearch, Rabbit MQ, ActiveMQ та багато інших;

– дозволяє підтримку адміністратора, тобто ви можете керувати за допомогою віддаленого доступу до програми.

3.3 Програмна реалізація front-end

Front-end частина компоненту MIC створена для ОС Anroid та розроблена мовою програмування Java. Для поєднання цієї частини з back-end використовується бібліотека Volley.

Volley – це бібліотека HTTP, яка робить роботу в мережі для додатків Android простішою і, головне, швидшою. Volley доступний на GitHub.

Volley пропонує наступні переваги:

– автоматичне планування мережевих запитів;

– кілька одночасних мережевих підключень;

– прозоре кешування відповідей на диск і пам’ять із стандартною узгодженістю кешу HTTP;

– підтримка визначення пріоритетів запитів;

– API запиту на скасування. Можливо скасувати один запит або встановити блоки чи діапазони запитів для скасування;

– простота налаштування, наприклад, для повторної спроби та повернення;

– надійне впорядкування, яке полегшує правильне заповнення інтерфейсу користувача даними, які асинхронно завантажуються з мережі;

– інструменти налагодження та трасування.

Volley відмінно виконує операції типу RPC, які використовуються для заповнення інтерфейсу користувача, наприклад отримання сторінки результатів пошуку як структурованих даних. Він легко інтегрується з будь-яким протоколом і поставляється з підтримкою необроблених рядків, зображень і JSON. Надаючи вбудовану підтримку необхідних функцій, Volley звільняє вас від написання шаблонного коду і дозволяє зосередитися на логіці, характерній для програми.

3.4 Опис бази даних

В якості системи сховища даних використовувся нереліаційних підхід збереження даних, тому було обрано базу данних MongoDB для компоненту МІС людей з обмеженими можливостями.

Нереліаційна база даних – є нетабличною базою даних і зберігає дані інакше, ніж реляційні таблиці. Бази даних NoSQL бувають різних типів на основі їх моделі даних. Основними типами є документ, ключ-значення, широкий стовпець і графік. Вони забезпечують гнучкі схеми і легко масштабуються з великими обсягами даних і високим навантаженням на користувачів.

MongoDB – програма керування базами даних NoSQL з відкритим вихідним кодом. NoSQL використовується як альтернатива традиційним

реляційним базам даних. Бази даних NoSQL досить корисні для роботи з великими наборами розподілених даних.

MongoDB – це інструмент, який може керувати документально-орієнтованою інформацією, зберігати або отримувати інформацію. MongoDB підтримує різні форми даних. Це одна з багатьох технологій нереляційних баз даних, які виникли в середині 2000-х під прапором NoSQL – зазвичай для використання в програмах для великих даних та інших роботах з обробки даних, які погано вписуються в жорстку реляційну модель.

Замість використання таблиць і рядків, як у реляційних базах даних, архітектура MongoDB складається з колекцій і документів. Організації можуть використовувати Mongo DB для спеціальних запитів, індексування, балансування навантаження, агрегації, виконання JavaScript на стороні сервера та інших функцій.

MongoDB використовує записи, які складаються з документів, які містять структуру даних, що складається з пар полів і значень. Документи є основною одиницею даних у MongoDB. Документи подібні до нотації об’єктів JavaScript, але використовують варіант під назвою Binary JSON (BSON). Перевага використання BSON полягає в тому, що він вміщує більше типів даних. Поля в цих документах подібні до стовпців у реляційній базі даних. Містяться значення можуть бути різних типів даних, включаючи інші документи, масиви та масиви документів, відповідно до посібника користувача MongoDB.

Документи також включатимуть первинний ключ як унікальний ідентифікатор. Набори документів називаються колекціями, які функціонують як еквівалент таблиць реляційної бази даних. Колекції можуть містити будь-які типи даних, але обмеження полягає в тому, що дані в колекції не можуть бути розподілені між різними базами даних. Оболонка mongo є стандартним компонентом відкритих дистрибутивів MongoDB. Після встановлення MongoDB користувачі підключають оболонку mongo до своїх запущених екземплярів MongoDB.

Оболонка mongo діє як інтерактивний інтерфейс JavaScript для MongoDB, що дозволяє користувачам запитувати й оновлювати дані, а також виконувати адміністративні операції.

Вся система MongoDB може представляти не тільки одну базу даних, що знаходиться на одному фізичному сервері. Функціональність MongoDB дозволяє розташувати кілька баз даних на декількох фізичних серверах, і ці бази даних зможуть легко обмінюватися даними і зберігати цілісність.

Для зберігання в MongoDB застосовується формат, який називається BSON (Бісон) або скорочення від binary JSON. Тому усі сутності медичної інформаційної системи зберігаються у форматі JSON.

На рисунку 3.4 зображений приклад зберігання сутності користувача з обмеженими можливостями у базі даних розроблюваного компоненту медичної інформаційної системи.

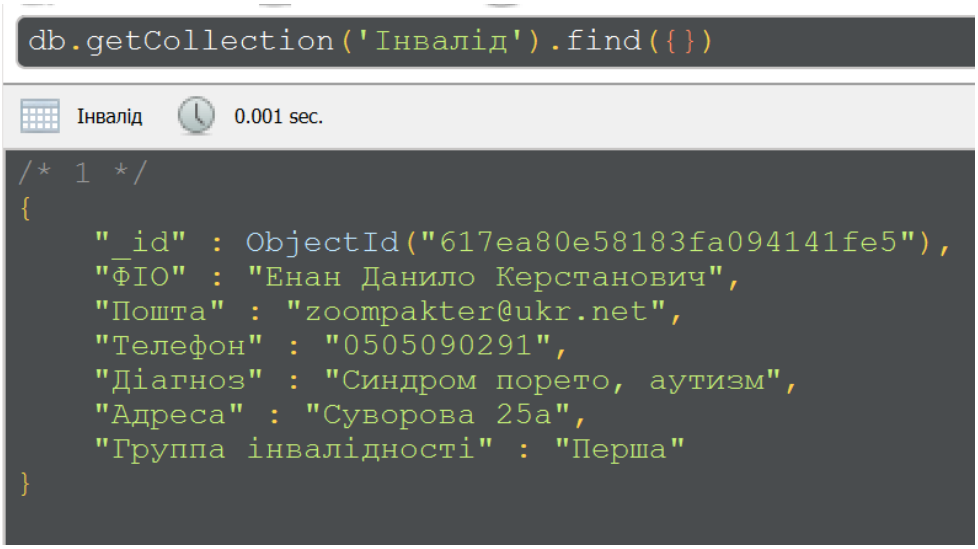

Рисунок 3.4 – Json-документ бази даних МІС

Оскільки документ зберігається у форматі Json, тому сутність відповідає його стандартам, має ключ та значення. Для кожної сутності MongoDB генерує ідентифікатор ключа який є унікальним для кожної сутності в базі даних та слугує індексом для швидкого відпрацювання запитів до бази.

На рисунку 3.5 зображено приклад зберігання сутності медсестри у базі даних.

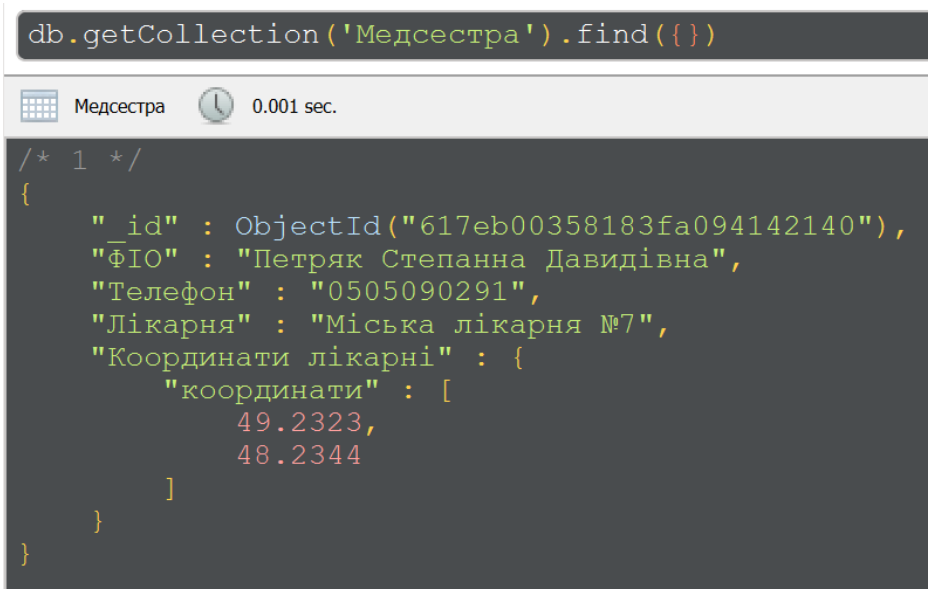

Рисунок 3.5 – Json-документ сутності медсестра

В координатах зберігаєтья довгота та широта заданої точки. Такий формат дуже легко дозволяє поєднати програмний компонент та використувувати запити до бази даних з метою опрацювання географічних координат.

3.5 Автоматизоване тестування компоненту

Для тестування компоненту MIC було використано технологію BDD. Розробка, орієнтована на поведінку (BDD) – це філія розробки, орієнтованої на тестування (TDD). BDD використовує зрозумілі людиною описи вимог користувача до програмного забезпечення як основу для тестування програмного забезпечення. Як і Domain Driven Design (DDD), раннім кроком у BDD є визначення спільного словника між зацікавленими сторонами, експертами в області доменів та інженерами. Цей процес включає визначення сутностей, подій і результатів, які турбують користувачів, і надання їм імен, з

якими кожен може погодитися. Професіонали BDD потім використовують цей словниковий запас, щоб створити специфічну для домену мову, яку вони можуть використовувати для кодування системних тестів, таких як тести приймання користувача (UAT).

Підхід до розробки, орієнтованої на поведінку – BDD – зосереджується навколо історій, написаних всюдисущою мовою, які описують очікувану поведінку програми. Використання зрозумілої для людини мови Gherkin дозволяє технічним та нетехнічним зацікавленим сторонам проекту брати участь у створенні описів функцій і, отже, тестів. Ці описи служать основою для роботи як розробників (описи специфікацій та функцій), так і тестувальників (кроки тестування). На рисунку 3.6 зображено основний принцип роботи BDD тестування у системах.

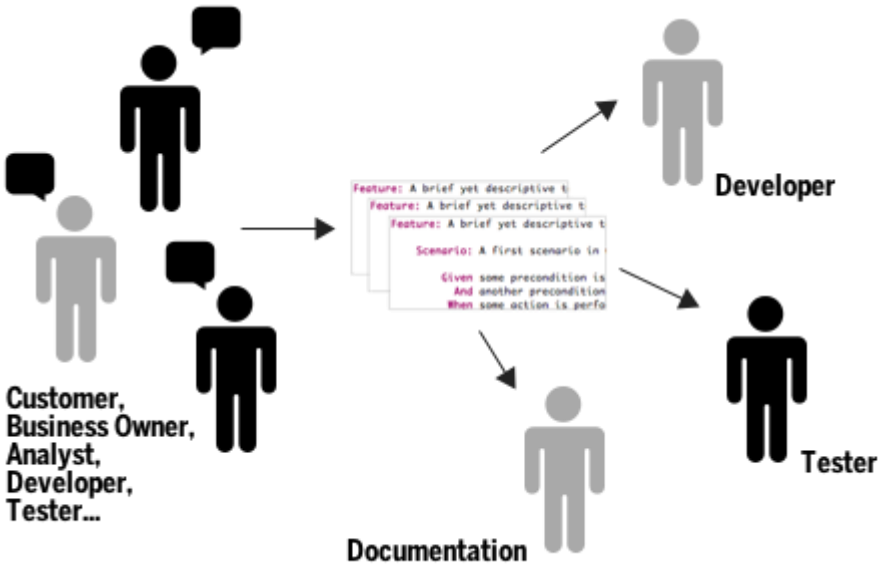

Рисунок 3.6 – Принцип технології BDD

В якості тестувального фреймворку було обрано Cucumber. Cucumber – це інструмент тестування, який підтримує розвиток, орієнтований на поведінку (BDD). Він пропонує спосіб писати тести, які можуть зрозуміти будь-хто, незалежно від їхніх технічних знань. У BDD користувачі (бізнес-

аналітики, власники продуктів) спочатку пишуть сценарії або приймальні тести, які описують поведінку системи з точки зору клієнта, для перегляду та підписання власниками продуктів, перш ніж розробники напишуть свої коди.

На рисунку 3.7 зображено сценарій виклику сиділки до пацієнта с обмеженими можливостями, який описує увесь бізнес процес компоненту MIC.

```
@integration
Scenario: Invalid patient should call nurse to home and get approximate arrival time
  Given I have patient with name DanielIvanov and with diagnosis Invalid and address Suborova57
  And I have invalid tender with name AnnaPavlovna , working in hospital 7 and phone number 380502791283
  And I have hospital with number 7 and with address Nauki7b
  When patient with name DanielIvanov press button NURSE CALL
  Then Nurse with name AnnaPavlovna should get notification
  And Patient should get information about arriving nurse in this table
    | callId | NurseName    | HospitalName | Approximate arrival time | phone       |
    | 1      | AnnaPavlovna | Hospital 7   | 10 minutes 30 seconds    | 380502791283 |
```

Рисунок 3.7 – Сценарій виклику сиділки до пацієнта

Сценарій описує послідовність бізнес процесів та функцій які необхідно виконати пацієнту для виклику медсестри на дім. Результат виконання сценарію зображено на рисунку 3.8. Якщо кожний окремий шаг був валідний то отримуємо зелений маркер, в разі усіх зелений маркерів можна вважати що тест пройденним, та компонент працює коректно згідно з функціональних вимог.

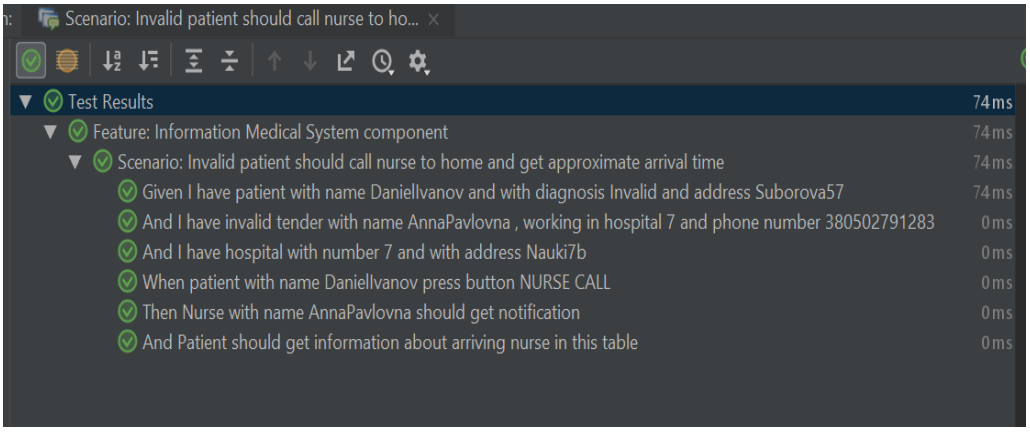

Рисунок 3.8 – Результат виконня сценарію

Плюсами автоматизованого тестування є:

– збережння коштівж;

– швидка розробка;

– збільшення продуктивності;

– висока якість розроблюваного продукту;

– швидкий відгук замовника.

Саме тому було вирішено використовувати автоматизоване тестування для компоненту MIC.

В якості тестування front-end частини застосунку MIC було обрано Selenium.

Першу версію Selenium створив Джейсон Хаггінс у 2004 році. Він втомився витрачати час і енергію на тестування веб-додатків і придумав бібліотеку Javascript, яка дозволила йому автоматично запускати тести для кількох браузерів. Таким чином, Selenium став першим інструментом, який дозволив користувачам керувати браузером за допомогою будь-якої мови програмування. Але хоча Selenium дозволяв своїм користувачам автоматизувати багато речей, він не обійшовся без недоліків. Оскільки він був заснований на Javascript, деякі речі було неможливо зробити. Крім того, вебпрограми з часом ускладнювалися, що спричиняло ще більше обмежень у інструменті. Через пару років інженер, який користувався інструментом у Google на ім’я Саймон Стюарт, набридли обмеженням Selenium. Він хотів створити інструмент, який би спілкувався безпосередньо з браузером, використовуючи його рідну мову та операційну систему. Так народився WebDriver. Знадобилося кілька років, щоб Selenium об’єднався з WebDriver, але коли вони все-таки об’єднали зусилля, це означало взяти найкраще з обох світів і зібрати під одним дахом величезну спільноту найяскравіших умів у автоматизації тестування. Відтоді сталося набагато більше, і на додаток до Selenium WebDriver, проект Selenium перетворився на набір інструментів, який складається з Selenium WebDriver, Selenium IDE та Selenium Grid. Пульт дистанційного керування Selenium (RC) також був частиною інструментарію,

але з тих пір він був знецінений, головним чином тому, що він був неймовірно повільним.

Selenium є найбільш популярним безкоштовним програмним забезпеченням і інструментом автоматизації з відкритим кодом. Переваги Selenium для автоматизації тестування величезні. Важливо, що він дозволяє записувати та відтворювати для тестування вебзастосунків і може запускати кілька сценаріїв у різних браузерах. Переваги Selenium Test Automation актуальні для різних сегментів бізнесу. Як згадувалося раніше, найбільшою перевагою Selenium є те, що він є безкоштовним і портативним інструментом. Він не має авансових прямих витрат. Інструмент можна безкоштовно завантажити, а його підтримка на основі спільноти доступна безкоштовно. Підтримка мови: Selenium підтримує ряд мов, включаючи Java, Perl, Python, C#, Ruby, Groovy, JavaScript тощо. У нього є власний сценарій, але він не обмежений цією мовою. Він може працювати з різними мовами – незалежно від того, що зручно розробникам/тестерам. Підтримує операційні системи: Selenium може працювати та підтримуватися в кількох операційних системах (ОС), таких як Windows, Mac, Linux та UNIX. За допомогою набору рішень Selenium можна створити спеціальний пакет тестування на будь-якій платформі, а потім виконати на іншій. Наприклад, можна створювати тестові приклади за допомогою ОС Windows і легко запускати його в системі на базі Linux. Підтримка в різних браузерах: Selenium підтримує кілька браузерів, а саме Internet Explorer, Chrome, Firefox, Opera та Safari. Це стає дуже винахідливим під час виконання тестів і тестування в різних браузерах одночасно. Підтримка мов програмування та фреймворку Selenium інтегрується з мовами програмування та різними фреймворками. Наприклад, він може інтегруватися з фреймворком ANT або Maven для компіляції вихідного коду. Крім того, він може інтегруватися з платформою TestNG для тестування програм і звітності. Він може інтегруватися з Jenkins або Hudson для безперервної інтеграції (CI) і навіть може інтегруватися з іншими інструментами з відкритим кодом для підтримки інших функцій. Тестування

на різних пристроях Selenium Test Automation можна реалізувати для автоматизації мобільних вебзастосунків на Android, iPhone та Blackberry. Це може допомогти отримати необхідні результати та вирішувати проблеми на постійній основі. Постійні оновлення. Підтримка Selenium заснована на спільноті, що забезпечує постійні оновлення та оновлення. Ці оновлення легко доступні і не вимагають спеціальної підготовки. Це також робить Selenium винахідливим і економічно ефективним.

Завантажені люкси Selenium Selenium – це не просто окремий інструмент або утиліта, це завантажений пакет різних інструментів тестування, тому його називають пакетом. Кожен інструмент розроблено для задоволення різних потреб тестування та вимог тестового середовища. Крім того, Selenium має можливості підтримки Selenium IDE, Selenium Grid і Selenium Remote Control (RC).

Простота виконання Selenium пропонує зручний інтерфейс, який допомагає легко та ефективно створювати та виконувати тести. Його функції з відкритим кодом допомагають користувачам створювати сценарії власних розширень, які полегшують їх розробку, налаштовують дії та навіть маніпулюють на просунутому рівні. Тести виконуються безпосередньо в браузерах, і користувачі можуть дивитися, поки вони виконуються. Крім того, можливості звітування Selenium є однією з причин його вибору, оскільки він дозволяє тестувальникам отримувати результати та виконувати наступні дії.

Повторне використання та доповнення платформа Selenium Test Automation використовує скрипти, які можна протестувати безпосередньо в кількох браузерах. Одночасно з Selenium можна виконувати кілька тестів, оскільки він охоплює майже всі аспекти функціонального тестування, реалізуючи додаткові інструменти, які розширюють сферу тестування. Існує інша школа думки, яка говорить про деякі прогалини в тестуванні автоматизації Selenium. Наприклад, експерти Test Automation також стверджують, що Selenium не є комплексним інструментом для автоматизації

тестування вебзастосунків, оскільки йому потрібні сторонні фреймворки та мовна підтримка, щоб працювати абсолютно функціонально та показувати необхідні результати.

Розробка, керована тестуванням (TDD) – це підхід до розробки програмного забезпечення, в якому тестові випадки розробляються для визначення та перевірки того, що буде робити код. Простіше кажучи, спочатку створюються та тестуються тестові випадки для кожної функціональності, а якщо тест не вдається, то пишеться новий код, щоб пройти тест і зробити код простим і без помилок. Розробка, орієнтована на тестування, починається з проектування та розробки тестів для кожної невеликої функціональності програми.

Фреймворк TDD інструктує розробників писати новий код лише в тому випадку, якщо автоматизований тест не пройшов. Це дозволяє уникнути дублювання коду. Повна форма TDD – це тестова розробка.

Проста концепція TDD полягає в написанні та виправленні невдалих тестів перед написанням нового коду (до розробки). Це допомагає уникнути дублювання коду, оскільки ми пишемо невелику кількість коду за раз, щоб пройти тести. (Тести – це не що інше, як умови вимоги, які нам потрібно перевірити, щоб їх виконати).

Розробка, керована тестуванням – це процес розробки та запуску автоматизованого тестування перед фактичною розробкою програми. Тому TDD іноді також називають тестовою першою розробкою.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований компонент МІС підбору персоналу для людей з обмеженими можливостями.

У ході кваліфікаційної роботи була розглянута частина невирішених питань медичної інформаційної системи для пацієнтів з обмеженими можливостями, а саме:

– онлайн підтримка пацієнта з обмеженими можливостями;
– виклик медсестри або сиділки на дім.

Для розв’язання проблем була спроектована модель компоненту медичної інформаційної системи для автоматизації процесу виклику медсестри пацієнту та можливості онлайн зв’язку з лікарем для пацієнтів з обмеженими можливостями.

Було розглянуто та проаналізовано існуючі рішення медичних інформаційних систем, виділено їх плюси та мінуси, досліджено відкриті питання щодо людей з обмеженими можливостями у медичній інформаційній системі та можливості і шляхи їх сучасного вирішення.

Для реалізації системи була створена поетапна модель розробки та проектування кожного компонента розроблюваної системи, опис необхідних бізнес та функціональних вимог. Спроєктовано та описано модель даних у системі, а також модель функціональних можливостей компоненту медичної інформаційної системи.

В якості тестування компоненту було розроблено модель автоматизованого тестування системи до її функціональних та бізнес вимог.

У ході проектування компоненту була розроблена модель основної концепції компоненту медичної інформаційної системи та аналіз вимог до частини невирішених питань підтримки пацієнтів з обмеженими можливостями.

У рамках розробки системи, було створено та адаптовано:

– компонент онлайн підтримки пацієнта;

– компонент виклику лікаря на дім.

Результати роботи було апробовано на Міжнародній науково-практичній конференції «MODERN TRENDS IN DEVELOPMENT SCIENCE AND PRACTICE».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Інформаційна система та програмне забезпечення інформаційної. URL: http://www.kievoit.ippo.kubg.edu.ua/kievoit/2013/95/95.html (дата звернення:19.10.2021).

2. Основні поняття баз даних. Відомості про інформаційні системи URL: https://sites.google.com/view/ddkbmta-info/лекції/системи (дата звернення:19.10.2021).

3. Засоби технічного забезпечення управління інформаційними ресурсами. URL: http://um.co.ua/8/8-12/8-127157.html (дата звернення:19.10.2021).

4. України, З. (2007). Про Основні засади розвитку інформаційного суспільства в Україні на 2007-2015 роки. Відомості Верховної Ради України (ВВР), 1(1), 12.

5. Адамик, О. В. (2016). 5.4. Бази і сховища даних–інформаційний фундамент бухгалтерського обліку та аналізу.

6. Качмар, В. О. (2010). Медичні інформаційні системи–стан розвитку в Україні. Український журнал телемедицини та медичної телематики, (8,№ 1), 12-17.

7. Охріменко, І. В. (2016). Автоматизована система оцінки фізіологічного стану організму людини. Механіка гіроскопічних систем, (32), 14-19.

8. МЕДИЧНИЙ, В. Н., & ПИРОГОВА, І. МЕДИЧНІ ЗНАННЯ ТА ПРИЙНЯТТЯ РІШЕНЬ В МЕДИЦИНІ.

9. Мельник, К. В. (2017). Моделювання процесу інтелектуальної обробки медичних даних. Системи обробки інформації, (4), 237-244.

10. Олексієнко, М. М. (2012). Проблеми та перспективи впровадження інформаційних технологій в медичну практику. Управління розвитком складних систем, (12), 133-136.

11. Адміністратор Центральної бази даних. URL: https://ehealth.gov.ua (дата звернення: 19.10.2021).

12. Medical information system. Що таке eHealth. URL: https://www.mcmed.ua/ua/ehealth (дата звернення: 19.10.2021).

13. Тітова, О. В. (2018). Основи інформаційного забезпечення управління.

14. Опис стандарту IDEF0. URL: http://easy-code.com.ua/2011/03/opis-standartu-idef (дата звернення:19.10.2021).

15. Опис стандарту IDEF3. URL: https://www.conceptdraw.com/examples/ (дата звернення:19.10.2021).

16. Ситников, Д. Э., Ситникова, П. Э., Титов, С. В., & Титова, Е. В. (2019). Определение параметров обобщенных ассоциативных правил методом декомпозиции. Системи обробки інформації, (1), 58-63.

17. Definition - What does IntelliJ IDEA mean? URL: https://www.conceptdraw.com/examples/idef3-diagram-software (дата звернення:19.10.2021).

18. Definition - What does Android Studio mean? URL: https://searchmobilecomputing.techtarget.com/definition/Android-Studio (дата звернення:19.10.2021).

19. What is REST. Guilding Principles of REST. URL: https://restfulapi.net (дата звернення:19.10.2021).

20. Spring Framework. URL: https://uk.wikipedia.org/wiki/Spring,_Framework (дата звернення:19.10.2021).

21. Understand how does Retrofiet works. URL: https://medium.com/mindorks/understand-how-does-retrofit-work-c9e264 (дата звернення:19.10.2021).

22. Android Nougat: Everything you need to know. Retrieved from https://www.androidcentral.com/nougat

23. What is MongoDB? Introduction, Architecture, Features & Example. URL: https://www.guru99.com/what-is-mongodb.html (дата звернення:19.10.2021).

24. Брацький, В. О., & М’якшило, О. М. (2016). Дослідження особливостей застосування реляційних і нереляційних баз даних на прикладі SQL Server та MongoDb. Наукові праці Національного університету харчових технологій, (22,№ 5), 15-24.

25. Ситников, Д. Э., Титов, С. В., & Титова, Е. В. (2011). Семантические свойства информативности ассоциативных зависимостей между признаками объектов в базах данных.

26. Титова, Е. В. (2003). Сравнительная характеристика простых и расширенных ассоциативных правил для признаков объектов в базах данных. Системи обробки інформації.–Х.: ХВУ, 2003.–Вип. 2, 31-37.

27. Sitnikov, D., Titova, O., Minukhin, S., Kovalenko, A., & Titov, S. (2018, October). Informativity of association rules from the viewpoint of information theory. In 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T) (pp. 595-598). IEEE.

28. Miroshnyk Y. (2021) ВАЖЛИВІСТЬ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПІДТРИМКИ ЛЮДЕЙ З ОБМЕЖЕНИМИ МОЖЛИВОСТЯМИ