

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики і радіоелектроніки, Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему: РОЗРОБКА ОПЕРАТИВНО-ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ
АНАЛІЗУ ЗАХВОРЮВАНЬ НА ОСОБЛИВО НЕБЕЗПЕЧНІ ХВОРОБИ.
INFORMATION SYSTEM DEVELOPMENT FOR DETECTING THE
DEADLIEST DISEASES

Виконав: студент(ка) 4 курсу, групи КНТ-217

Спеціальності 122 «Комп'ютерні науки»
(код і найменування спеціальності)

Освітня програма (спеціалізація)
«Комп'ютерні науки»

Гончарова А. В.
(прізвище та ініціали)

Керівник Гладкова О. М.
(прізвище та ініціали)

Рецензент Терещенко Е. В.
(прізвище та ініціали)

ЗАТВЕРДЖУЮ
Завідувач кафедри _____
проф., д.т.н., С.О. Субботін
« » 20 року

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5	Гладкова О.М., доцент		
Нормоконтроль	Дейнега Л.Ю., ст.викладач		

7. Дата видачі завдання « 02 » _____ квітня _____ 2021року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області та огляд аналогів	1 тиждень	Висновки за розділом
2	Визначення вимог та обґрунтування структури системи, що розроблюється	1-2 тиждень	Технічне завдання, структурна схема, функціональна схема
3	Основні рішення з реалізації програмного продукту	2-3 тиждень	Компоновка програми
4	Проектування інтерфейсів системи	3-4 тиждень	Екранні форми
5	Керівництво користувача	4-5 тиждень	Інструкції по користуванню
6	Розробка розділів пояснювальної записки	5-6 тиждень	Пояснювальна записка
7	Розробка слайдів презентації	6 тиждень	Слайди презентації
8	Захист дипломної роботи	7 тиждень	

Студент(ка)

_____ Гончарова А. В.

(підпис)

(прізвище та ініціали)

Керівник проєкту (роботи)

_____ Гладкова О. М.

(підпис)

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 113 с., 1 табл., 44 рис., 4 дод., 17 джерел.

ІНФОРМАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, АНОНІМІЗАЦІЯ ДАНИХ, COVID-19, ЗАХВОРЮВАНІСТЬ, ЛІКАРНЯ, СИМПТОМИ, ІНТЕРФЕЙС.

Об'єкт проектування – вебсайт для перегляду користувачами та реєстрації лікарями випадків захворювання на особливо небезпечні хвороби.

Предмет проектування – база даних щодо зберігання випадків захворювання та вебінтерфейс для доступу до неї.

Мета роботи – створення оперативно-інформаційної системи для аналізу захворювань на COVID-19, яка дозволить підвищити ефективність протиепідемічних заходів щодо боротьби проти смертельної хвороби.

Розроблена інформаційна система дозволить виконувати такі функції:

- збір даних щодо захворюваності на COVID-19 у будь-якій регіональній структурі Запорізької області: область, район міста або села, з будь-якої клініки, лабораторії чи станції первинної допомоги за наявності доступу до всесвітньої мережі Інтернет;

- можливість розширення кількості варіантів особливо небезпечних хвороб з однієї COVID-19 до 40 або навіть і більше без суттєвих переробок програмного коду;

- додаткова функція зворотнього зв'язку для пацієнтів з підозрілими симптомами;

- моніторинг зі зміненнями у реальному часі актуальних статистичних даних щодо кількості хворих на COVID-19, померлих від ускладнень та інше;

- відображення графіків: «Загальна кількість інфікованих», «Кількість інфікованих по днях», «Загальна кількість померлих», «Кількість смертей по днях», «Кількість видужуючих по днях» тощо.

Серверну частину призначено для роботи під операційною системою Linux, а клієнтська частина може працювати у будь-якому браузері.

ABSTRACT

Bachelor thesis: 113 p., 1 tabs, 44 fig., 4 appendixes, 17 references.

INFORMATION SYSTEM, DATABASE, DATA ANONYMIZING, COVID-19, MORBIDITY, HOSPITAL, SYMPTOMS, INTERFACE.

The object of study is the website for users to view and for doctors to register cases of especially deadliest diseases.

Design subject is a database of disease cases and a web interface for accessing it.

The purpose of the work is to create an instant information system development for detecting the COVID-19 diseases, which will increase the effectiveness of anti-epidemic measures to deal with the deadly disease.

Developed information system will allow performing the following functions:

- data collection on the incidence of COVID-19 in any regional part of Zaporizhzhia oblast: region, district of city or village, from any clinic, laboratory or primary care station, if access to the Internet is available;
- the ability to expand the number of variants of particularly dangerous diseases from single one like COVID-19 up to 40 or even more without significant software code rewriting;
- additional feedback function for patients with suspicious symptoms;
- real-time monitoring of changes in current statistics on the number of patients with COVID-19, including the ones dead from complications, etc;
- graphs display for "Total number of infected", "Daily number of infected", "Total number of deaths", "Daily number of deaths", "Daily number of recovering", etc.

The server part is designed to run on the Linux operating system, and the client part can run in any browser.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ	7
ВСТУП	8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД АНАЛОГІВ	9
1.1 Характеристика особливо небезпечних хвороб	9
1.2 Аналіз централізованої системи моніторингу поширення епідемії коронавірусу	11
1.3 Висновки за розділом 1	13
2 ВИЗНАЧЕННЯ ВИМОГ ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЩО РОЗРОБЛЮЄТЬСЯ	14
2.1 Створення структурної схеми інформаційної системи	15
2.2 Розробка функціональної схеми інформаційної системи	16
2.3 Концептуальне проєктування бази даних	17
2.4 Нормалізація	22
3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ	25
3.1 Обґрунтування вибору середовища розробки інформаційної системи	25
3.2 Вибір системи керування базами даних	25
3.3 Проєктування таблиць	34
3.4 Використання розширення PostgreSQL-Anonymizer за для приховування конфіденційних даних	37
4 ПРОЄКТУВАННЯ ІНТЕРФЕЙСІВ СИСТЕМИ	42
4.1 Обґрунтування вибору типу інтерфейсів оперативно-інформаційної системи	42
4.2 Структура вебсайту оперативно-інформаційної системи	43
5 КЕРІВНИЦТВО КОРИСТУВАЧА	48
5.1 Призначення програми	48
5.2 Характеристики програми	48

5.3 Звертання до програми	50
ВИСНОВКИ.....	57
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	58
ДОДАТОК А Технічне завдання	60
ДОДАТОК Б Опис програми	66
ДОДАТОК В Текст програми	70
ДОДАТОК Г Слайди презентації	105

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД	– база даних;
СКБД	– система керування базами даних;
ШВЛ	– система штучної вентиляції легень;
1NF (1 Normal Form)	– перша нормальна форма;
2NF (2 Normal Form)	– друга нормальна форма;
Anonymizer	– розширення для маскуванню даних;
ChartJS	– бібліотека на мові JavaScript для побудови діаграм;
COVID-19	– особливо небезпечне захворювання (виявлене 2019 р.);
CSS (Cascading Style Sheets)	– каскадні таблиці стилю;
ER (Entry Relationship)	– діаграма відображення концептуальної моделі бази даних;
HTML (HyperText Markup Language)	– мова розмітки гіпертексту;
JS (JavaScript)	– динамічна, об'єктно-орієнтована прототипна мова програмування;
JSON (JavaScript Object Notation)	– текстовий формат обміну даними;
PHP (Hypertext Preprocessor)	– скриптова мова програмування;
SARS-CoV-2	– код штаму грипу, який викликає захворювання на COVID-19;
SHA-256 (Secure Hash Algorithm)	– безпечний алгоритм гешування;
SQL (Structured Query Language)	– мова структурованих запитів.

ВСТУП

У цілому Світі завжди та в усі часи найважливішим було є і буде життя людини. Однією із загальних проблем людства, яка впливає на життя людей є хвороби. Кількість хвороб безмежна, але з них є особливо небезпечні, які призводять до смерті та мають властивість розповсюджуватись від людини до людини, а іноді від тварини до людини, та ще й швидкість розповсюдження може бути катастрофічно великою.

Найактуальнішою проблемою людства на сьогодні є боротьба з вірусними інфекціями різних штамів грипу, а особливо COVID-19, який викликає у людини швидкотечну пневмонію та багато інших ускладнень, і від цього вірусу вже за 2 роки з моменту виявлення було зафіксовано більше ніж 3 млн. летальних випадків у людей та більше ніж 161 млн. інфікованих [1].

Важливою задачею боротьби з поширенням збудників, які викликають особливо небезпечні хвороби людини є своєчасне інформування епідемічних служб і населення для застосування їми протиепідемічних заходів та зменшення рівнів захворюваності у цілому Світі. Для вирішення цієї задачі можна застосувати сучасні методи та засоби інформатики, програмування та комп'ютерної інженерії.

На даний момент часу задача створення оперативно-інформаційної системи для аналізу захворювань на особливо небезпечні хвороби має стати найактуальнішою, що допоможе спростити та пришвидшити процедури масового інформування населення, а також надасть можливість використовувати статистичні дані для епідеміологічної та медичної боротьби з особливо небезпечними збудниками хвороб, що може значно знизити смертність і дозволить зупинити поширення Світом інфекцій грипу COVID-19 та інших.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД АНАЛОГІВ

1.1 Характеристика особливо небезпечних хвороб

Захворюваннями на особливо небезпечні хвороби є захворювання, які викликані особливо небезпечними інфекціями та представляють надзвичайну епідемічну небезпеку для оточення.

Особливо небезпечні інфекції з'являються миттєво, розповсюджуються надшвидко та охоплюють значну частину населення у найкоротші строки. Такі інфекції протікають з яскраво-вираженою клінічною картиною, як правило, мають важку течію та високу летальність.

На сьогодні Всесвітньою організацією охорони здоров'я (ВООЗ) до списку особливо небезпечних інфекцій додано більше 100 захворювань.

Також встановлений перелік карантинних інфекцій: поліомієліт, чума (легенева форма), холера, жовта лихоманка, натуральна віспа, лихоманка Ебола та Марбург, грип (новий підтип), гострий респіраторний синдром.

У країнах із жарким кліматом поширені особливо небезпечні інфекційні захворювання, такі як холера, лихоманки Денге і Зіка, жовта лихоманка, чума, малярія та ряд інших. Щорічно в Україні реєструються завезені випадки захворювань малярією, тропічними гельмінтозами.

Верховною Радою України було затверджено Наказ Міністерства охорони здоров'я України № 133 від 19.07.1995р. із змінами, внесеними згідно з Наказом Міністерства охорони здоров'я № 521 (v0521282-20) від 25.02.2020р. «Про затвердження Переліку особливо небезпечних, небезпечних інфекційних та паразитарних хвороб людини і носійства збудників цих хвороб», де було зазначено та оновлено перелік особливо небезпечних інфекційних хвороб для України [2]:

а) особливо небезпечні інфекційні хвороби:

- 1) чума;
- 2) холера;
- 3) натуральна віспа;

- 4) жовта гарячка;
- 5) синдром набутого імунodefіциту (снід);
- 6) пастерельоз;
- 7) хвороба марбург;
- 8) гарячка ласса;
- 9) гарячка ебола;
- 10) контагіозні вірусні гарячки – денге, чикунгунья, долини ріфт, західного нілу;
- 11) енцефаломієліти – західно- і східно-американські, венесуельський;
- 12) енцефаліти – каліфорнійський, сент-луїс, долини муррея;
- 13) бруцельоз;
- 14) туляремія;
- 15) сибірка;
- 16) сап;
- 17) меліоїдоз;
- 18) орнітоз;
- 19) лістеріоз;
- 20) сказ;
- 21) еризипелоїд;
- 22) легіонельоз;
- 23) епідемічний висипний тиф;
- 24) хвороба брілла;
- 25) ку – гарячка;
- 26) мишиний тиф;
- 27) марсельська гарячка;
- 28) кліщовий поворотний висипний тиф;
- 29) туберкульоз;
- 30) псевдотуберкульоз;
- 31) геморагічна гарячка;

- 32) кримська гарячка;
- 33) омська гарячка;
- 34) гарячка з нирковим синдромом;
- 35) лептоспіроз;
- 36) ящур;
- 37) кліщовий енцефаліт;
- 38) хвороба лайма;
- 39) covid-19 (за міжнародною класифікацією sars-cov-2);

б) носійство збудників особливо небезпечних хвороб:

- 1) холери;
- 2) віл-інфекції.

Визнання особи, хворою на інфекційну або паразитарну хворобу, або носієм збудника інфекційної, або паразитарної хвороби здійснюється лікарями-інфекціоністами, терапевтами, педіатрами, фтизіатрами на підставі клінічного перебігу хвороби, результатів лабораторних досліджень та епідеміологічного анамнезу. В окремих випадках визнання особи, хворою на інфекційну або паразитарну хворобу здійснюється комісійно (інфекціоніст, педіатр, терапевт, епідеміолог, невропатолог, фтизіатр та інші). Протиепідемічні, профілактичні і карантинні заходи щодо інфекційних та паразитарних хвороб людини передбачені у відповідних правилах, положеннях, інструкціях, вказівках та інших нормативних актах, які встановлюються Міністерством охорони здоров'я України [2].

1.2 Аналіз централізованої системи моніторингу поширення епідемії коронавірусу

Майже завжди виникнення осередків захворюваностей на особливо небезпечні хвороби було локальним та не приводило до всесвітньої пандемії, але з появою у Світі в кінці 2019 року штаму грипу SARS-CoV-2 все змінилося і майже цілий Світ поринув у тотальний карантин. Пандемія

захворюваності на COVID-19 поширюється надзвичайно швидко по всьому Світі. Людство зустрілося вже із двома хвилями поширення цієї хвороби. Проводиться багато досліджень, створюються декілька вакцин для лікування, створено безліч примірників та брошур з рекомендаціями та умовами протидії поширенню інфекції. Крім цих порад, важливим є інформування населення щодо поширення грипу SARS-CoV-2 під назвою «Коронавірус», а також оперативного розповсюдження даних щодо захворюваностей та збору статистики серед лікарів, щоб завдяки протиепідемічним заходам найшвидше припинити поширення цієї інфекції, яка вже завдала у Світі багато смертельних випадків.

Для вирішення цих задач було створено централізовану систему моніторингу поширення епідемії коронавірусу [3]. Головну сторінку всеукраїнського вузлу цієї системи відображено на рисунку 1.1.



Рисунок 1.1 – Головна сторінка всеукраїнського вузлу системи моніторингу на захворюваність коронавірусом

Ця система дозволяє отримати статистичну інформацію щодо поширення захворюваності на COVID-19 і в Україні, і в усьому Світі.

У процесі дипломного проектування було проведено детальний аналіз цієї інформаційної системи, визначені її можливості, виділено переваги та зазначено недоліки.

Переваги:

- глобальне інформування, отримання даних захворюваності через Інтернет щодо будь-якого регіону Світу;
- моніторинг даних зі зміненнями у реальному часі;
- простий доступ до системи через браузер без потреби встановлення програмного забезпечення.

Недоліки:

- централізоване надходження даних, спочатку до центрального міжнародного серверу, а потім розповсюджується по клієнтським вузлам, відсутнє ієрархічне пірамідальне структурування;
- надмірне використання ресурсів браузера клієнтського комп'ютера, як наслідок першого недоліку;
- перелік показників дуже скудний, що недостатньо для розширеного статистичного аналізу медпрацівниками;
- максимальну деталізацію показників захворюваності визначено середніми показниками по областях України, що зводить на нівець моніторинг за районами або лікарнями області чи міста;
- закритий код системи унеможливорює автоматичний парсинг даних для використання у нових міських або державних розробках.

1.3 Висновки за розділом 1

Швидкість розповсюдження коронавірусної інфекції у Світі та сукупні зусилля людства на подолання цієї проблеми надали можливість створити поки що тільки одну централізовану систему моніторингу поширення епідемії коронавірусу [3]. Зрозуміло, що в неї є недоліки, які можна спробувати подолати у процесі створення нової програмної системи.

2 ВИЗНАЧЕННЯ ВИМОГ ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЩО РОЗРОБЛЮЄТЬСЯ

Згідно технічного завдання (додаток А) необхідно розробити оперативно-інформаційну систему для аналізу захворювань на особливо небезпечні хвороби. Ця система повинна функціонувати оперативно, тобто у режимі реального часу, зберігати дані щодо захворюваності на коронавірусну інфекцію COVID-19 та виконувати статистичний аналіз даних з відображенням результатів на графіках.

Розроблювана інформаційна система дозволить виконувати такі функції:

- збір даних щодо захворюваності на COVID-19 у будь-якій регіональній структурі: область, район міста або села, з будь-якої клініки, лабораторії чи станції первинної допомоги за наявності доступу до всесвітньої мережі Інтернет;

- можливість структурної організації децентралізованого ієрархічного надходження даних до головного серверу, що підвищить швидкодію інформаційної системи та значно зменшить використання ресурсів на клієнтських вузлах;

- можливість розширення кількості варіантів особливо небезпечних хвороб з однієї COVID-19 до 40 або навіть і більше без суттєвих переробок програмного коду;

- зберігання у системі розширеного переліку показників для подальшого додаткового дослідження медпрацівниками;

- можливість організації взаємодії оперативно-інформаційної системи з центральною медичною базою даних Міністерства охорони здоров'я України;

- додаткова функція зворотнього зв'язку для пацієнтів з підозрілими симптомами;

- простий доступ до системи через браузер без потреби встановлення програмного забезпечення;
- моніторинг зі зміненнями у реальному часі актуальних статистичних даних щодо кількості хворих на COVID-19, померлих від ускладнень та інше;
- відображення графіків: «Загальна кількість інфікованих», «Кількість інфікованих по днях», «Загальна кількість померлих», «Кількість смертей по днях», «Кількість видужуючих по днях» тощо.

Головною відмінністю розроблюваної оперативно-інформаційної системи від централізованої системи моніторингу поширення епідемії коронавірусу [3] є реалізація онлайн-моніторингу за районами або лікарнями міста Запоріжжя та Запорізької області.

2.1 Створення структурної схеми інформаційної системи

Розроблювана оперативно-інформаційна система складається із частин, які мають взаємодіяти один з одним. Це вказує на модульність структури розробки, що має дозволити у майбутньому легко модернізувати її або виконати заміну будь-якого модулю.

Загалом інформаційна система складається із чотирьох частин:

- а) операторська (призначено для лікаря, якій вносить дані щодо захворюваностей);
- б) клієнтська (для відвідувачів сайту інформаційної системи);
- в) адмінська (для головного адміністратора сайту інформаційної системи, який авторизує лікарів та робить налаштування серверу);
- г) центральний сервер, якій містить базу даних та модуль аутентифікації для розгалуження доступу користувачів до БД, що має звести до мінімуму вірогідність зламу системи.

Структурну схему розроблюваної оперативно-інформаційної системи наведено на рисунку 2.1.

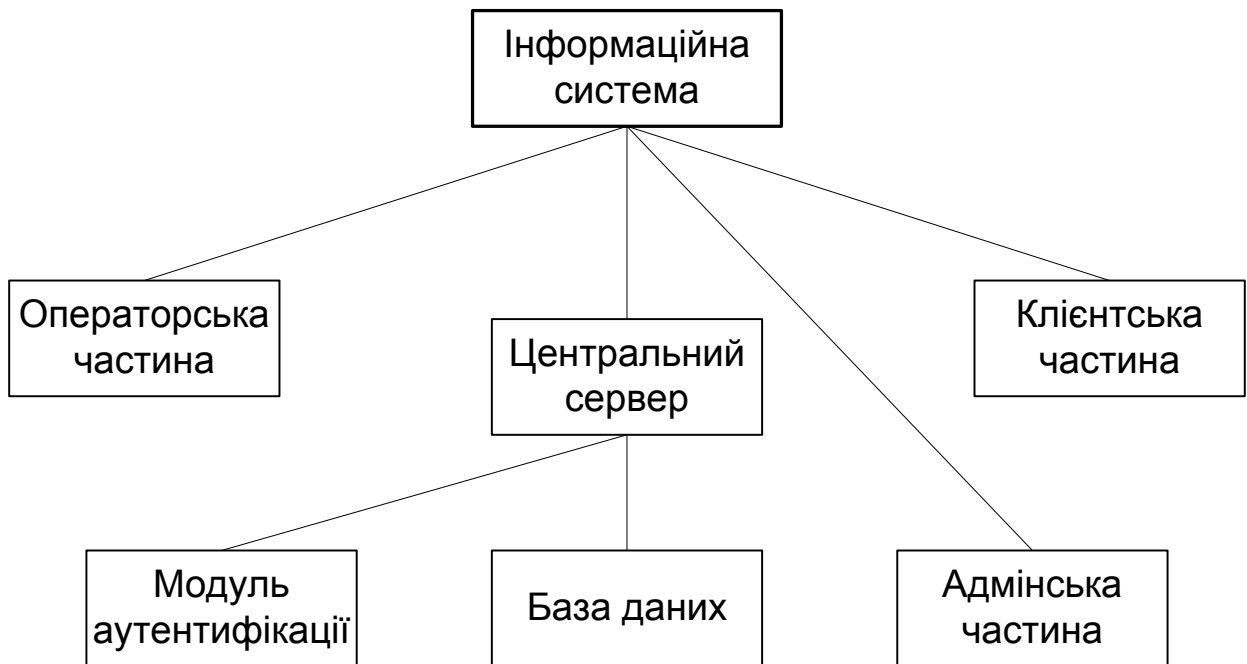


Рисунок 2.1 – Структурна схема оперативно-інформаційної системи

2.2 Розробка функціональної схеми інформаційної системи

Функціональна схема оперативно-інформаційної системи відображає склад модулів та підмодулів системи та деталізує процес передачі даних від одного модуля до іншого.

Функціональна схема складається з центральної бази даних і модуля аутентифікації, які розташовані на головному сервері та інтерфейсів оператора (лікаря) і клієнта (відвідувача), через які відповідно будуть взаємодіяти велика кількість операторів (1..М) та клієнтів (1..N). Головний адміністратор системи (root) безпосередньо керує центральною базою даних, налаштовує параметри системи та керує іншими модулями.

З інтерфейсу оператора у двох напрямках до БД для даних дозволені права додавання, читання, та змінення. У процесі входу оператора до системи логін та геш паролю передаються до модуля аутентифікації для перевірки достовірності користувача, що дозволяє відкрити БД для оператора. З інтерфейсу клієнта до БД передаються вхідні параметри, за якими на відповідь отримуються статистичні дані для аналізу та відображення на

графіках. Функціональну схему розроблюваної оперативно-інформаційної системи наведено на рисунку 2.2.

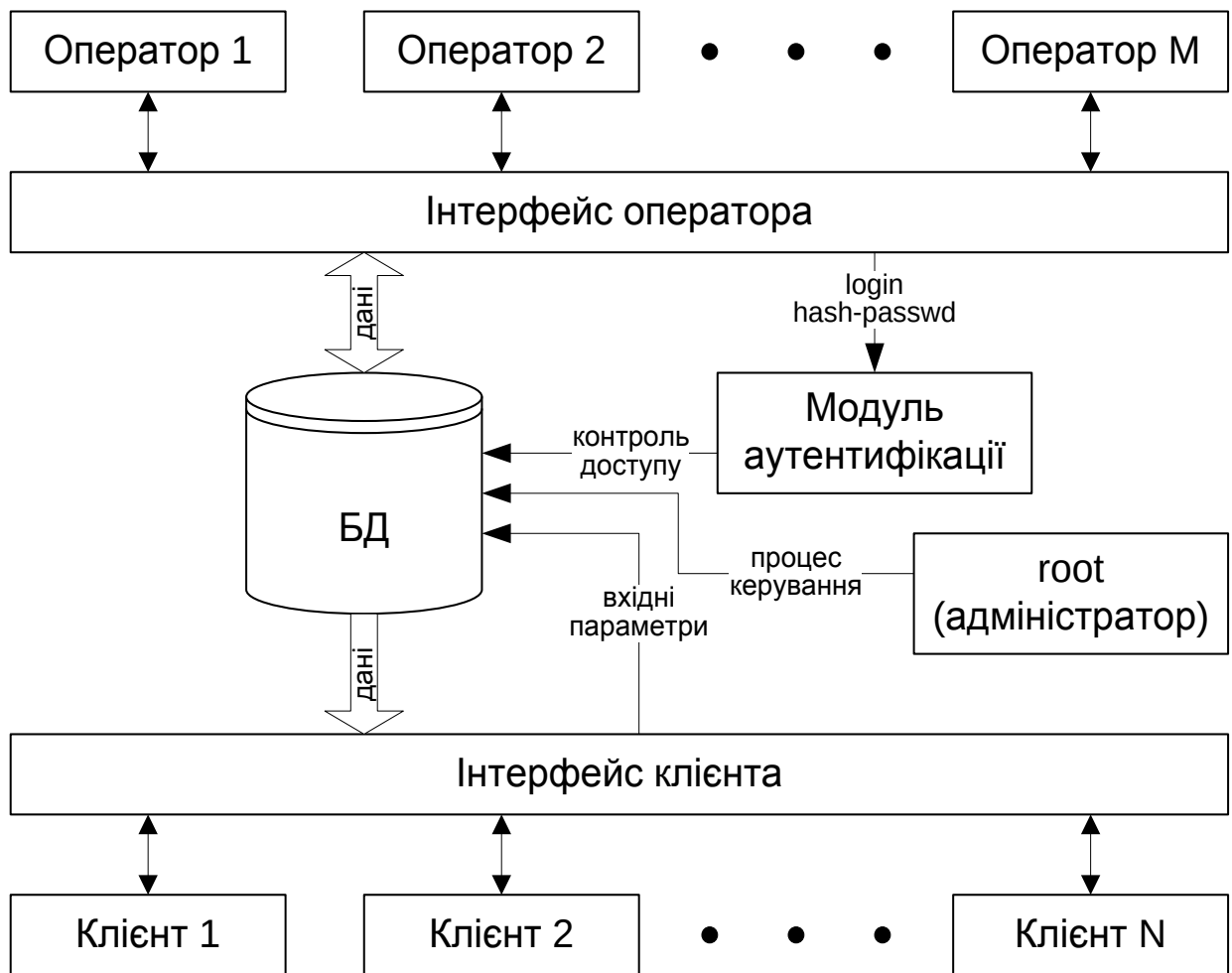


Рисунок 2.2 – Функціональна схема оперативно-інформаційної системи

2.3 Концептуальне проєктування бази даних

Після створення структурної та функціональної схем розроблюваної інформаційної системи, необхідно створити базу даних. У цьому випадку найкраще використати наступні дві моделі даних – концептуальну та реляційну [4].

Процес проєктування бази даних має наступну визначену послідовність:

а) концептуальне проєктування;

- б) логічне проєктування;
- в) фізичне проєктування.

Протягом концептуального проєктування повинно завершитися формування задумів розроблюваної бази даних, але можна не враховувати фізичні аспекти її реалізації. На поточному етапі проєктування розробника не повинна цікавити ні конкретна СКБД, яку буде потім застосовано щодо розгортання БД, ні використовувані для створення інформаційної системи мови програмування, ні технології, ні фреймворки, ні особливості апаратної платформи. Спочатку необхідно створити загальну модель, що сприяє процесу функціонування інформаційної системи та містить всі необхідні для системи дані. Вся потрібна для цього інформація має вже бути зібрана на попередньому етапі циклу проєктування бази даних [5].

Основним засобом побудови концептуальної моделі БД виступає модель «сутність-відношення» (ER-модель), але іноді можуть бути використані споріднені моделі. Суть таких моделей – проста і одночасно ефективна методика, яка дозволить уявити сенс всього того, що підлягає зберіганню у базі даних. На останньому етапі концептуального проєктування розробник бази даних повинен обов'язково перевірити адекватність створеної моделі, і для цього створюється перевірка отриманих результатів (елементів ER-діаграм). Кінцевим результатом етапу концептуального проєктування буде ER-модель майбутньої бази даних, яка буде містити опис:

- типів сутностей;
- відношень між типами сутностей;
- атрибутів (з попереднім описом варіацій та обмежень);
- первинних ключів [6].

Для предметної області оперативно-інформаційної системи було визначено наступні сутності:

- «Клініки»;
- «Пацієнти»;
- «Користувачі».

Кожна визначена сутність має відповідні атрибути:

а) «Клініки»:

- 1) код;
- 2) назва;
- 3) кількість ліжок всього;
- 4) кількість ліжок із ШВЛ;

б) «Пацієнти»:

- 1) код;
- 2) дата прибуття;
- 3) стан здоров'я на дату прибуття (легка/середня/важка форма);
- 4) П.І.Б.;
- 5) вік (років);
- 6) адреса реєстрації;
- 7) телефон;
- 8) кількість контактних осіб (всього);
- 9) кількість контактних осіб (захворілих);
- 10) потреба у апараті ШВЛ;
- 11) дата виписки;
- 12) стан здоров'я на дату виписки (включно з летальним випадком);
- 13) місце розташування;
- 14) лікар;

в) «Користувачі»:

- 1) код;
- 2) ім'я (логін);
- 3) пароль (парольний геш SHA-256);
- 4) статус користувача;
- 5) локація користувача.

Інформаційна система розроблюється як багатокористувацька і призначена, як для лікарів медичних клінік та амбулаторій щодо реєстрації випадків захворювання так і для пересічних людей для реєстрації симптомів захворювання із зворотнім зв'язком. У концептуальній моделі визначено відношення між сутностями «Клініки» та «Пацієнти». Так як в одному медичному закладі можливо обслуговувати багато пацієнтів, то відношення між сутностями «Клініки» і «Пацієнти» визначено як один-до-багатьох.

Так як один лікар може обслідувати багато пацієнтів, то відношення між сутностями «Користувачі» і «Пацієнти» визначено теж як один-до-багатьох.

Протягом процесу концептуального проектування було створено модель бази даних, яка зображена на ER-діаграмі (рис. 2.3).



Рисунок 2.3 – Концептуальна модель бази даних

На етапах логічного проєктування необхідно перетворити узагальнену концептуальну модель до завершеної логічної моделі. Потрібно зробити уточнення усіх вимог, які були виявлені на стадії концептуального проєктування і обов'язково спростити рішення (не знижуючи їх функціональних можливостей). Для цього ER-модель необхідно перевірити за допомогою правил нормалізації. У результаті отримуються ненадлишкові реляційні таблиці, які мають бути вільні від наступних аномалій – вставки, редагування та видалення [6].

Окрім нормалізації, на логічному етапі треба здійснити наступні дії:

- уточнити обмеження даних;
- визначити домени даних;
- увести бізнес-правила та корпоративні обмеження цілісності;
- визначити місця розташування майбутніх таблиць.

Надалі необхідно повторно виконати звіряння логічної моделі. Внести завершуючи уточнення та виправлення.

На етапах концептуального та логічного проєктування розробники мають використати одну із двох стратегій створення БД: висхідне проєктування (bottom – up design) або низхідне проєктування (top – down design). Висхідний підхід найчастіше застосовується для невеликих проєктів. Суть цього методу полягає в тому, що проєктувальник має створити повний список атрибутів, що підлягають зберіганню. Пізніше ці атрибути групують у типи сутностей, які потрапляють до моделі. Процес нормалізації заснований на висхідному підході. Зворотній висхідному процесу – низхідна стратегія найкраще підходить для середніх і великих проєктів [5].

Проєктування починається із процесу виявлення основних типів сутностей та тільки тоді сутності «наповнюються» атрибутами. Класичний приклад низхідного методу – це модель «сутність-зв'язок» (ER-модель). Необхідно спільно застосовувати висхідну і низхідну методології (у даному випадку – нормалізацію таблиць та ER-модель).

2.4 Нормалізація

Процес нормалізації переслідує декілька цілей. Важливий з них – боротьба з надмірністю даних. Усунення надмірностей досягається не тільки за рахунок раціонального призначення розмірності полям таблиць, але й за рахунок формування ефективних взаємопов'язаних табличних структур, що мають об'єднувати декілька відношень. Отже процес нормалізації має виступити в ролі основної частини процесу проєктування баз даних [6].

У результаті нормалізації бази даних, таблиці приймають додаткову гнучкість, що дозволить без особливих наслідків змінювати структуру таблиць у майбутньому, наприклад, у процесі модернізації функціонуючої бази даних. У боротьбі за зайві байти не лише мінімізуються витрати на зберігання даних, а також усувається ряд непрямих проблем або аномалій, які властиві некоректно побудованим таблицям. До них належать аномалії вставки, редагування і видалення даних. Для усунення цих аномалій проводиться нормалізація даних. Нормалізація – це процес послідовного перетворення таблиць бази даних до такого виду, який прийнято в реляційній моделі даних [7].

Розрізняють п'ять послідовних форм (етапів) приведення даних до реляційного виду від першої нормальної форми (First Normal Form, 1NF) до п'ятої (5NF). На кожному з етапів нормалізації, таблиця приймає нові риси, які необхідні для переходу до чергового етапу. Тому, щоб прийти до 5NF, необхідно на кожному з етапів крок за кроком методично вдосконалити структуру таблиць. Перестрибнути через будь-який з етапів нормалізації не можна [8].

Приведеною до першої нормальної форми (1NF) таблиця, якщо в ній відсутні групи, що повторюються і значення, які зберігаються, є неділимі (атомарні). У результаті приведення до 1NF в цілях боротьби з групами, що повторюються, необхідно спростити класифікацію станів здоров'я пацієнтів на момент обстеження та на момент виписки. У процесі проєктування

таблиці розробник намагається, щоб інформація, яка зберігається у неї, стала не лише ненадлишковою, але й раціонально взаємопов'язаною [6].

Таблиця буде відповідати другій нормальній формі (2NF), якщо вона приведена до 1NF і в ній відсутні часткові залежності. У цій таблиці не повинно бути атрибутів, які залежні тільки від частини первинного ключу. Відмова від складеного первинного ключу має ще одну перевагу – спрощення процесу приведення таблиці до 2NF. Якщо первинний ключ не є складеним, то немає бути часткових залежностей. Таблиця приведена до другої нормальної форми тільки тоді, якщо вона відповідає 1NF та в ній немає часткових залежностей, тобто немає неключових полів, які мають залежати від частини первинного ключу [7].

На іншому етапі нормалізації, за рахунок відмови від складеного ключа необхідно позбавитися від часткових залежностей. Але при цьому можуть залишитися транзитивні залежності. Транзитивною є така залежність між атрибутами відношення, коли один з атрибутів пов'язаний з іншим через атрибут-посередник $A \rightarrow B \rightarrow C$. У даному випадку, перетворення таблиці із 2NF до 3NF призводить до декомпозиції початкового відношення. Таблиці, які не залежать від будь-яких інших таблиць, називають «довідковими», оскільки вони є джерелами довідкових даних щодо підлеглих за відношенням до них таблиць.

Довідкові таблиці особливо корисні, і от чому:

- істотно скорочується фізичний розмір підлеглої таблиці. Замість реальних даних, що займають десятки байтів у підлеглий таблиці зберігається значення зовнішнього ключа, яке зазвичай не перевищує одного байту;
- при редагуванні значень немає необхідності переглядати безліч записів, а досить змінити одне значення в довідковій таблиці;
- при введенні даних значно знижується вірогідність появи помилки користувача, оскільки замість введення з клавіатури пропонується на вибір заздалегідь підготовлені значення [8].

Таблиця буде приведена до третьої нормальної форми (3NF), якщо вона має відповідати 2NF та у ній відсутні транзитивні залежності.

Третя нормальна форма використовується для боротьби із транзитивними залежностями, але четверта нормальна форма (4NF) відповідає у конфронтації з іншим недоліком реляційної моделі – багатозначними залежностями. Багатозначна залежність – це відношення «багато-до-багатьох».

Логічну схему бази даних оперативно-інформаційної системи наведено на рисунку 2.4.

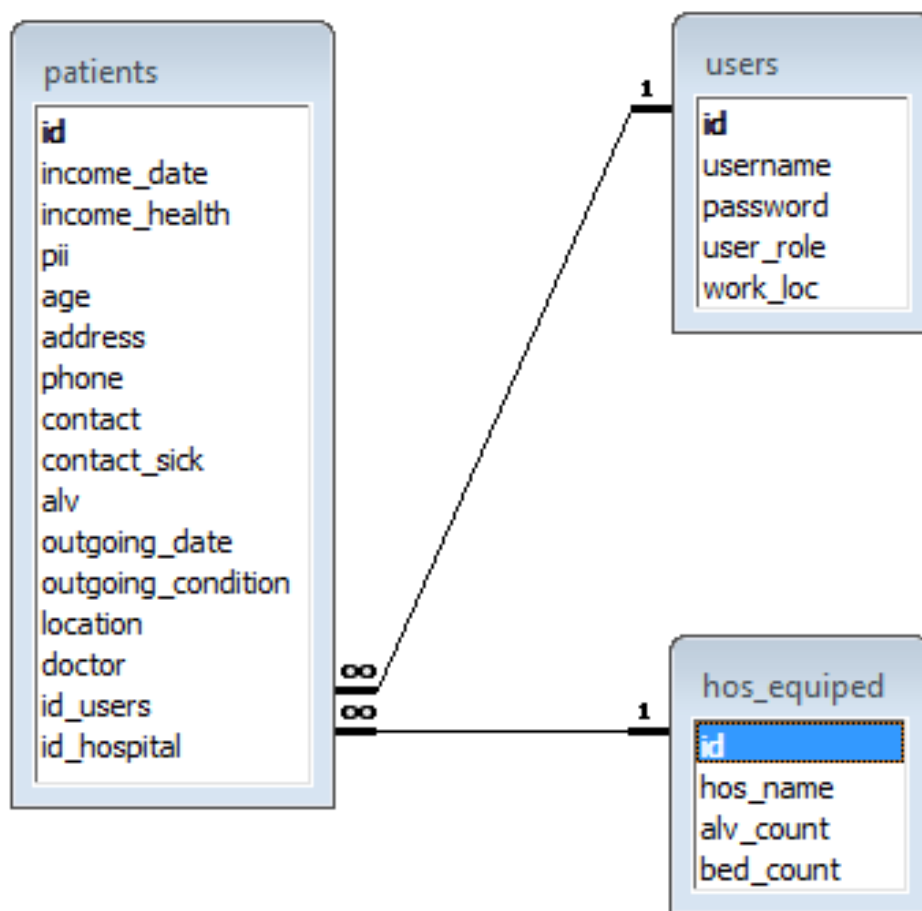


Рисунок 2.4 – Логічна схема реляційної бази даних

Згідно правил нормалізації логічну схему доведено до рівня 2NF, спрощено структури таблиць до мінімуму та визначено зв'язки між таблицями.

3 ОСНОВНІ РІШЕННЯ З РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ

3.1 Обґрунтування вибору середовища розробки інформаційної системи

До етапів фізичного проєктування бази даних переходять тільки після обрання цільової системи керування базами даних (СКБД), яка має визначати особливості майбутнього програмного продукту. Визначитися з СКБД необхідно ще до початку етапів фізичної розробки БД.

Фізичне проєктування – уточнення рішень з урахуванням наявних розробників технологій, можливостей реалізації і необхідної продуктивності. У процесі фізичного проєктування завданням проєктувальника є перенесення логічної моделі до платформи цільової СКБД. З цією метою розробник має виконати наступні дії:

- створити таблиці та зв'язки між ними;
- призначити вторинні індекси таблиць;
- реалізувати бізнес-логіку бази даних;
- визначити функціональні характеристики транзакцій;
- розробити представлення;
- впровадити механізми захисту (як мінімум – цей процес передбачає авторизацію користувачів і призначення правил доступу до даних) [5].

Надважливе – це визначити призначені для користувача типи даних, зробити описи таблиць і зв'язків між ними, задати порядок підтримки бізнес-логіки, встановити призначення порядку викликів процедур, які зберігаються і тригерів.

3.2 Вибір системи керування базами даних

Ефективність функціонування бази даних найтіснішим чином пов'язано з СКБД, завдяки якій буде реалізовано проєкт. Обрання відповідної

платформи багато в чому визначає переваги і недоліки бази даних. Існує великий набір критеріїв оцінки СКБД, виконавши які проєктувальник може зупинитися на конкретному програмному продукті [9].

Найперший крок – визначення типу моделі даних. Провівши аналіз ER-моделі (рис. 2.3) та логічної схеми (рис. 2.4) бази даних інформаційної системи, було прийнято рішення використати реляційну модель даних.

Для забезпечення доступу до інформації, яка зберігається у базі даних, а також для керування цією БД застосовується система керування базами даних (СКБД). СКБД – це комплекс мовних та програмних засобів, якій призначено для створення, ведення і спільного використання бази даних багатьма користувачами. Звично СКБД розрізняють за використовуваною моделлю даних. СКБД, які базуються на використанні реляційної моделі даних мають назву реляційних СКБД [8].

Функції і можливості будь-якої СКБД можуть істотно відрізнятися. Різні СКБД документовані більш менш ретельно. По-різному надається технічна підтримка. Щодо порівняння різних популярних систем керування базами даних враховуємо, що зручно для користувача, чи масштабована ця конкретна СКБД, а також чи буде добре інтегруватися із іншими продуктами, які вже використовуються. Під час вибору враховується вартість системи та підтримки, яка буде надаватися розробником.

Існує декілька популярних СКБД, за гроші та безкоштовних, які можна рекомендувати для застосування в оперативно-інформаційній системі. Виконавши пошук, буде розглянуто перелік з СКБД, які наведені нижче.

Oracle 12.

Перша версія Oracle була створена у кінці 70-х років. Існує декілька версій цього продукту для застосування у різних сферах.

Актуальну версію Oracle 12 (с) – призначено для хмарних технологій і її може бути розміщено на одному або декілька серверах, що дозволить керувати базами даних, які містять більш мільярдів записів. Деякі з функцій цієї версії Oracle містять технології Grid-Framework і використання як

фізичних так і логічних структур. Це означає, що фізичне керування даними не має впливати на доступ до логічних структур. Крім того, безпека у цій версії доведена до найвищого рівня завдяки ізоляції транзакцій однієї від інших.

Переваги:

- інновації та новий функціонал;
- висока надійність серед подібних систем.

Недоліки:

- непомірно висока вартість, особливо для невеликих або бюджетних організацій;
- потреба системою значних ресурсів одразу після інсталяції;
- найкращий вибір для великих організацій, які працюють із величезними базами даних і різноманітними функціями.

MySQL.

Одна з найпопулярніших систем керування базами даних для вебзастосунків. Фактично – це стандарт для вебсерверів, які працюють під керуванням операційної системи Linux. MySQL – безкоштовний пакет програм, але постійно з'являються нові версії, які розширюють функціонал і покращують безпеку. У наявності спеціальні платні версії, які призначено для комерційного використання. В безкоштовних версіях найбільший напрям робиться на швидкість і надійність, але не на повноту функціонала, який може стати як перевагою так і недоліком, залежно від галузі впровадження.

З недавнього часу розробку та підтримку MySQL здійснює корпорація Oracle, яка отримала права на торгівельну марку разом з поглиненою Sun Microsystems, котра раніше придбала шведську компанію MySQL AB. Продукт поширюється як під GNU (General Public License) так і під власною комерційною ліцензією. Крім того, розробники створюють функціональність по замовленням ліцензійних користувачів. Завдяки такому замовленню майже у самих ранніх версіях з'явився механізм реплікації [10].

Ця СКБД дозволяє обрати різні рушії щодо системи зберігання даних, які мають дозволити змінити функціонал інструменту і виконувати обробку цих таблиць, що зберігаються у різних типах. Гнучкість MySQL забезпечується підтримкою великої кількості типів таблиць. Завдяки відкритій архітектурі та GPL-ліцензуванню у MySQL постійно з'являються нові типи таблиць. Також ця СКБД має простий у використанні інтерфейс і пакетні команди, що дозволяють зручно обробити величезні об'єми даних. Система дуже надійна та не має надмірної потреби щодо ресурсів [10].

Переваги:

- безкоштовна;
- документована;
- безліч функцій, навіть у безкоштовній версії;
- пакет MySQL міститься у стандартних репозиторіях найбільш поширених дистрибутивів операційних систем Linux;
- підтримка набору призначених для користувача інтерфейсів;
- сумісність з іншими базами даних, включно DB2 та Oracle.

Недоліки:

- складність у створенні інкрементних резервних копій;
- відсутність вбудованої підтримки XML або OLAP;
- безкоштовна версія має тільки платну підтримку [10].

DB2.

Ця система керування базою даних створена компанією IBM. DB2 є СКБД, яка має можливості NoSQL, але додатково може використовувати JSON та XML-файли. Система розроблялася для серверів компанії IBM модельного ряду iSeries, працює на Windows, Linux та Unix.

Діалект мови SQL, який використовується у DB2 строго декларативний за рідкісними виключеннями. Система забезпечена багатофазовим оптимізатором, який будує за цими декларативними конструкціями план виконання запиту. У діалекті SQL DB2 відсутні підказки оптимізатора, недостатньо розвинена мова процедур, що зберігаються. Мова

SQL DB2 являється повністю обчислювальною, тобто потенційно дозволяє у декларативній формі визначати будь-які обчислювані відповідності між початковими даними і результатом. Це досягається в тому числі за рахунок використання табличних виразів, рекурсії та інших розвинених механізмів маніпулювання даними.

Оптимізатор DB2 широко використовує статистику розподілу даних в таблицях, тому один і той же запит на мові SQL може бути трансльований до абсолютно різних планів виконання залежно від статистичних характеристик даних, які він має обробити.

В концепції підвищення рівня інтеграції засобів безпеки у комп'ютерній системі, DB2 не має власних засобів аутентифікації користувачів, інтегруючись з засобами операційної системи або спеціалізованими серверами безпеки. В рамках DB2 здійснюється тільки авторизація користувачів, які аутентифіковані системою.

Сучасні версії DB2 забезпечують розширену підтримку використання даних у форматі XML. Остання версія DB2 забезпечує вдосконалені функції аварійного відновлення, сумісності та аналітики.

Переваги:

- Blu Acceleration дозволяє грамотно задіяти ресурси баз даних великих розмірів;
- бази даних мають можливість бути розміщені у хмарному середовищі, на фізичному сервері або в обох місцях одночасно;
- декілька завдань можуть бути виконані одночасно за допомогою планувальника завдань;
- відстеження завдань завдяки кодам помилок та кодам завершення.

Недоліки:

- надмірна вартість продукту;
- вимагається стороннє застосування чи додаткове програмне забезпечення, щоб змусити функціонувати кластери або декілька вторинних вузлів;

- базова підтримка безкоштовна тільки протягом трьох років;
- підходить для великих організацій, які планують вичавлювати максимум з наявних ресурсів і обробляти великі БД.

SAP HANA.

Цю СКБД розроблено компанією SAP SE. Ядро призначене для збереження та витягання даних із застосувань та інших джерел на декількох рівнях зберігання. Дані можуть бути розміщені як на фізичних серверах так і в хмарних середовищах.

Переваги:

- підтримуються SQL, OLTP і OLAP;
- ядро має зниження вимог щодо ресурсів за рахунок використання стиснення;
- дані зберігаються у пам'яті, що значно скорочує час доступу;
- звіти формуються у реальному часі;
- у наявності можливість взаємодіяти з низкою інших застосувань.

Недоліки:

- висока вартість ліцензій;
- вимагаються постійні оновлення.

Microsoft SQL-сервер.

Ця система керування базами даних, рушій якої вже працює на хмарних серверах, а також на локальних серверах, та при цьому можна комбінувати типи вживаних серверів одночасно. Після випуску Microsoft SQL-сервер 2016 компанія Microsoft адаптувала продукт щодо операційної системи Linux.

Одна з унікальних особливостей версії 2016 року є Temporal Data Support (тимчасова підтримка даних), яка має дозволити відстежувати зміни даних з часом. Остання версія Microsoft SQL-сервер підтримує Dynamic Data Masking (динамічне маскування даних), що гарантує найкращий захист авторизованих користувачів.

Переваги:

- простота у використанні;
- швидкість та стабільність;
- можливість візуалізації на мобільних пристроях;
- коректна взаємодія з іншими Microsoft-продуктами.

Недоліки:

- вартість неприйнятна для більшої частини організацій;
- проблеми з використанням служби інтеграції щодо імпорту файлів.

MariaDB.

Ця система є безкоштовною, але пропонує і платні версії. Має безліч доступних доповнень і розширень.

MariaDB – відгалуження від СКБД MySQL, яка розробляється співтовариством під ліцензією GNU GPL. Розробка і підтримка MariaDB здійснюється компанією MariaDB Corporation AB і фондом MariaDB Foundation.

Ядро системи керування базами даних дозволяє робити вибір із декількох систем зберігання і це робить використання ресурсів більш оптимізованим, що підвищує продуктивність запитів та обробки. До складу MariaDB включені підсистеми зберігання даних XtraDB для можливості заміни InnoDB, як основної підсистеми зберігання. Також включені підсистеми Aria, PBXT і FederateX. Вона повністю сумісна із MySQL та прекрасно підходить як заміна, оскільки повністю відповідає як набір команд так і API.

Переваги:

- гарна швидкодія;
- індикація обробки запитів;
- розширювана архітектура та доповнення;
- шифрування доступне не тільки у мережі, але й на сервері та на рівні застосування.

Недоліки:

- стабільність нижча, ніж у MySQL;
- платна підтримка.

MongoDB.

Безкоштовна система керування базами даних, яка має комерційну версію. Дана СКБД призначена для застосувань, які можуть використовувати і структуровані, і неструктуровані дані. Ядро цієї системи дуже гнучке та працює щодо підключення баз даних до застосувань через драйвери MongoDB. Існує широкий вибір доступних драйверів.

Переваги:

- швидкість і простота у використанні;
- рушій підтримує JSON та інші документи NoSQL.

Недоліки:

- SQL не використовується як мова запитів;
- інструменти для SQL-запитів у MongoDB як доповнення.

PostgreSQL.

PostgreSQL є однією з декількох безкоштовних популярних варіантів СКБД, яка найчастіше використовується щодо ведення баз даних вебсайтів. Вона була одна з найперших розроблених систем керування базами даних, тому вона добре розвинена і дозволяє користувачам керувати як структурованими так і неструктурованими даними. Вона може бути використана на більшості основних платформ, включно Linux. Відмінно справляється з завданнями імпорту інформації з інших типів баз даних за допомогою власного інструментарію [11].

Рушій СКБД може бути розміщений в ряді середовищ, у тому числі віртуальних, фізичних і хмарних. Найсвіжіша версія – PostgreSQL 11.11, пропонує обробку великих об'ємів даних і збільшення кількості одночасно працюючих користувачів. Безпеку було поліпшено завдяки підтримці DBMS_SESSION [11].

Переваги:

- масштабована та здатна обробляти терабайти даних;
- підтримка формату JSON;
- має безліч зумовлених функцій;
- доступно багато інтерфейсів.

Недоліки:

- неповна документація;
- складне конфігурування;
- швидкість роботи знижується під час проведення пакетних операцій

або виконання запитів читання [11].

Всі розглянуті особливості кожної СКБД впорядковано у звідну таблицю 3.1.

Таблиця 3.1 – Порівняльний аналіз параметрів СКБД

СКБД	Розробник	Тип	Операційна система	Ліцензія	Початковий код	Підтримка	Стабільність	Популярність
Oracle Database	Oracle Corporation	Мульти-модельна	Linux, Microsoft Windows, Oracle Solaris, IBM AIX, HP-UX	Комерційна	Закритий	За гроші	+	+
MySQL	Oracle Corporation	Реляційна	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	GNU GPL та комерційна	Відкритий	За гроші	+	+
DB2	IBM	Об'єктно-реляційна	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	Пропріетарна EULA	Закритий	Безкоштовна	+	–
SAP HANA	SAP SE	Реляційна, in-memory	Linux	Комерційна	Закритий	За гроші	+	–
SQL Server	Microsoft	Реляційна	Linux, Microsoft Windows	Комерційна	Закритий	Безкоштовна	+	+
MariaDB	MariaDB Corporation AB, MariaDB Foundation	Реляційна	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	GNU GPL	Відкритий	За гроші	–	–
MongoDB	MongoDB Inc.	Документо-орієнтована	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	GNU AGPL (СКБД) та Apache License (драйвери)	Відкритий	За гроші	+	+
PostgreSQL	PostgreSQL Global Development Group	Об'єктно-реляційна	Linux, Microsoft Windows, Oracle Solaris, IBM AIX, HP-UX, macOS, QNX	Вільне та відкрите програмне забезпечення, дозвільна ліцензія	Відкритий	За гроші	+	+

За результатами таблиці можна зробити висновок, що в процесі проєктування вебсерверу оперативно-інформаційної системи слід у першу

чергу обернути увагу на PostgreSQL, яка залишається затребуваною вже більше десяти років. Альтернативами будуть MySQL та MariaDB.

3.3 Проектування таблиць

Система керування базами даних PostgreSQL версії 11.11 дозволяє швидко та легко створити структуру БД, яку зображено на рисунку 2.4.

Перш за все необхідно назначити назву бази даних. Для бази даних розроблюваної інформаційної системи було прийнято назву «covid_data». Надалі необхідно коректно налаштувати базу даних у СКБД, створити таблиці та увімкнути додатковий функціонал.

Таблиці створюються за допомогою вбудованого до СКБД інструментарію phpPgAdmin під акаунтом головного адміністратора системи (рис. 3.1).

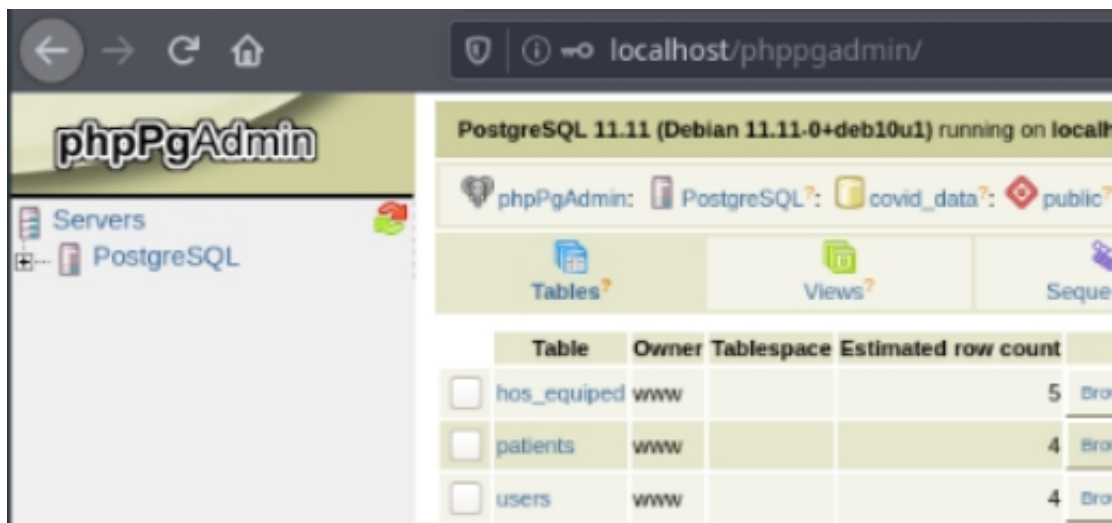


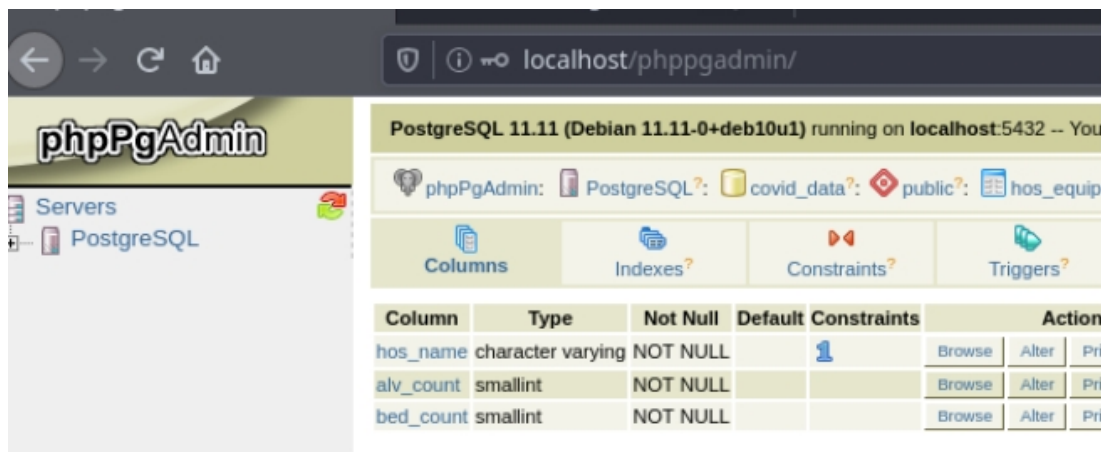
Рисунок 3.1 – Головні таблиці бази даних

Перелік головних таблиць:

- hos_equiped;
- patients;
- users.

Надалі необхідно створити структуру кожної таблиці.

На рисунку 3.2 відображено створення таблиці `hos_equiped`, яка зберігає дані щодо клінік, загальну кількість ліжок та кількість ліжок із системами ШВЛ, що може надати лікарня для пацієнтів.

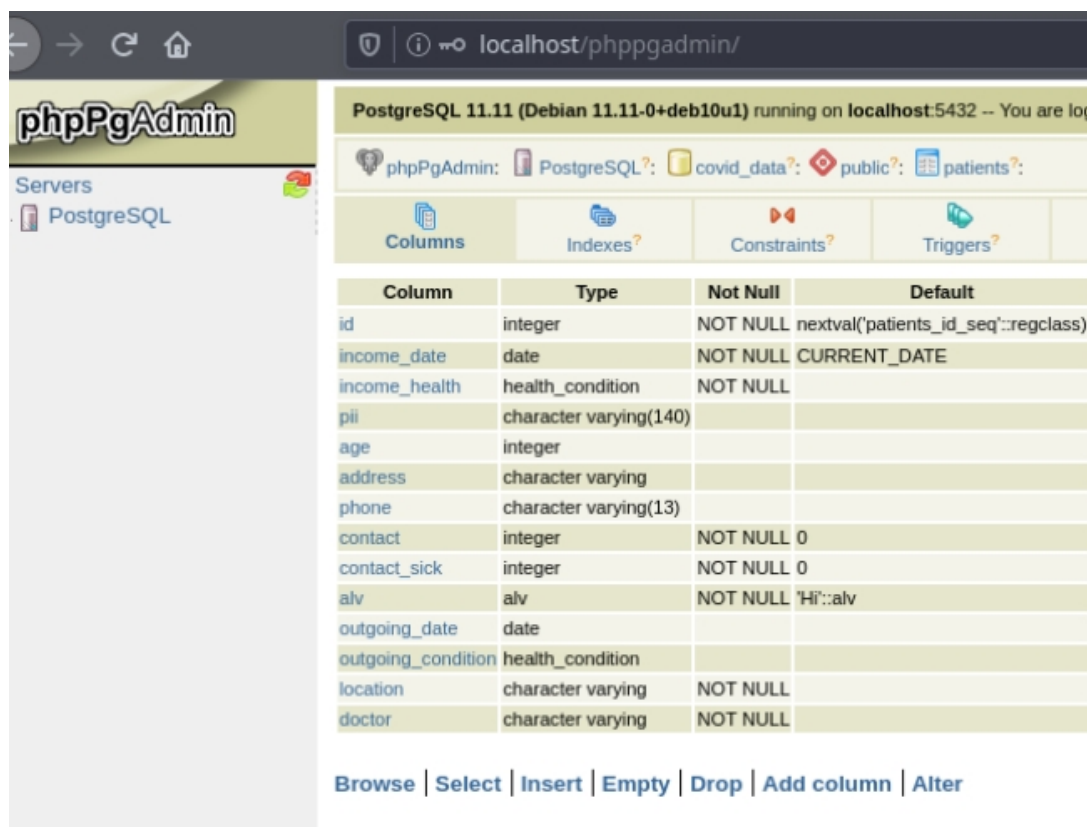


The screenshot shows the phpPgAdmin interface for PostgreSQL 11.11. The left sidebar shows 'Servers' and 'PostgreSQL'. The main panel displays the table 'hos_equiped' with its structure. The table has three columns: 'hos_name' (character varying, NOT NULL, primary key), 'alv_count' (smallint, NOT NULL), and 'bed_count' (smallint, NOT NULL). The interface includes tabs for Columns, Indexes, Constraints, and Triggers. Below the table structure, there are links for Browse, Alter, and Pri for each column.

Column	Type	Not Null	Default	Constraints	Action
hos_name	character varying	NOT NULL		1	Browse Alter Pri
alv_count	smallint	NOT NULL			Browse Alter Pri
bed_count	smallint	NOT NULL			Browse Alter Pri

Рисунок 3.2 – Вигляд структури таблиці `hos_equiped`

На рисунку 3.3 відображено створення таблиці `patients`, яка зберігає дані про пацієнтів.



The screenshot shows the phpPgAdmin interface for PostgreSQL 11.11. The left sidebar shows 'Servers' and 'PostgreSQL'. The main panel displays the table 'patients' with its structure. The table has 17 columns: 'id' (integer, NOT NULL, primary key), 'income_date' (date, NOT NULL), 'income_health' (health_condition, NOT NULL), 'pii' (character varying(140)), 'age' (integer), 'address' (character varying), 'phone' (character varying(13)), 'contact' (integer, NOT NULL, 0), 'contact_sick' (integer, NOT NULL, 0), 'alv' (alv, NOT NULL, 'Hi'::alv), 'outgoing_date' (date), 'outgoing_condition' (health_condition), 'location' (character varying, NOT NULL), and 'doctor' (character varying, NOT NULL). The interface includes tabs for Columns, Indexes, Constraints, and Triggers. Below the table structure, there are links for Browse, Select, Insert, Empty, Drop, Add column, and Alter.

Column	Type	Not Null	Default
id	integer	NOT NULL	nextval('patients_id_seq'::regclass)
income_date	date	NOT NULL	CURRENT_DATE
income_health	health_condition	NOT NULL	
pii	character varying(140)		
age	integer		
address	character varying		
phone	character varying(13)		
contact	integer	NOT NULL	0
contact_sick	integer	NOT NULL	0
alv	alv	NOT NULL	'Hi'::alv
outgoing_date	date		
outgoing_condition	health_condition		
location	character varying	NOT NULL	
doctor	character varying	NOT NULL	

Browse | Select | Insert | Empty | Drop | Add column | Alter

Рисунок 3.3 – Вигляд структури таблиці `patients`

На рисунку 3.4 відображено створення таблиці users, яка зберігає необхідні дані для авторизації та розподілу доступу користувачів.

PostgreSQL 11.11 (Debian 11.11-0+deb10u1) running on localhost:5432 -- You are

phpPgAdmin: PostgreSQL?: covid_data?: public?: users?:

Columns Indexes? Constraints? Triggers?

Column	Type	Not Null	Default	Constraints
id	integer	NOT NULL	nextval('users_id_seq'::regclass)	
username	character varying	NOT NULL		
password	character varying	NOT NULL		
user_role	user_status	NOT NULL	'nikap'::user_status	
work_loc	worktable	NOT NULL		

Browse | Select | Insert | Empty | Drop | Add column | Alter

Рисунок 3.4 – Вигляд структури таблиці users

Після створення головних таблиць необхідно розширити функціонал бази даних, для цього створюється три типи перелічень (enum), які будуть використовуватися у БД для легкості введення даних та оптимізації використання ресурсів (рис. 3.5).

```

Файл  Правка  Вид  Закладки  Настройка  Справка
covid_data=# select enum_range(enum_first(null::user_status),null::user_status);
          enum_range
-----
{лікар,завідуючий,адміністратор}
(1 строка)

covid_data=# select enum_range(enum_first(null::health_condition),null::health_condition);
          enum_range
-----
{"NULL","Легка форма","Середня форма","Важка форма",Одужав(ла),Помер(ла)}
(1 строка)

covid_data=# select enum_range(enum_first(null::alv),null::alv);
          enum_range
-----
{Так,Ні}
(1 строка)

covid_data=#

```

Рисунок 3.5 – Створення користувацьких типів перелічень

Користувачькі типи перелічень допомагають оптимізувати базу даних за рахунок кількості необхідних для них ресурсів у порівнянні з більш традиційним підходом – створення окремої довідкової таблиці з id та значенням. Дослідження проводились у наступних статтях [12,13].

3.4 Використання розширення PostgreSQL-Anonymizer за для приховування конфіденційних даних

Кожна база даних отримує, зберігає та оброблює дані. Так як розроблювана оперативно-інформаційна система буде зберігати певні конфіденційні дані пацієнтів, то необхідно забезпечити їх безпеку при використанні та обробці. Щоб не зменшувати швидкодію інформаційної системи навантаженнями шифрувальних засобів, найпростіше зробити приховування деяких даних завдяки готовому розширенню PostgreSQL-Anonymizer, яке спеціально було створено для використання у СКБД PostgreSQL. Це розширення маскує дані ще на сервері для певних груп користувачів, що не дозволяє конфіденційним даним потрапити на клієнтську машину [14].

Концепція даної розробки дозволяє відобразити реальні дані з лікарні лише лікарям, що працюють в цій самій лікарні. Так у БД представлені всього 5 тестових лікарень «Лікарня №1», «Лікарня №2», «Лікарня №3», «Лікарня №4», «Лікарня №5». Відповідно їм додані тестові користувачі. Кожен лікар з лікарні №1 може переглянути персональні дані усіх пацієнтів цієї лікарні, але не в змозі переглянути персональні дані пацієнтів лікарень №№2–5. Звичайний відвідувач сайту також не має можливості переглянути персональні дані жодної з лікарень, але бачить загальну статистику та перебіг хвиль захворювання.

Вигляд БД із активованим розширенням виглядає дещо інакше (рис. 3.6).



	Schema	Owner	Actions			Comment
☐	anon	www	Drop	Privileges	Alter	
☐	mask	www	Drop	Privileges	Alter	
☐	public	postgres	Drop	Privileges	Alter	standard public schema

Рисунок 3.6 – Зображення схем БД з phpPgAdmin із активованим розширенням PostgreSQL-Anonymizer

Встановлене до СКБД розширення PostgreSQL-Anonymizer додає до інструментарію phpPgAdmin додаткову схему з таблицями із назвами «anon» та «mask». Зазвичай всі користувацькі таблиці створюються та розташовуються до схеми «public».

У такому випадку алгоритм роботи з БД дещо змінюється. Як показує досвід із всесвітньої практики, не зважаючи на те, що кількість операцій запитів збільшується, час роботи і відповіді від цього не зростають, у порівнянні із випадком коли всі необхідні данні необхідно шифрувати або дешифрувати. Цей аспект може бути дуже вагомим при масштабуванні БД інформаційної системи до рівнів обласного або всеукраїнського значення. Адже зі збільшенням покриття збільшується кількість підключень, що необхідно обробити в один і той самий момент часу.

Після створення структури БД необхідно додати розширення PostgreSQL-Anonymizer для подальшого маскування даних. Надалі структура БД зі схемами буде мати наступний вигляд (рис. 3.7).

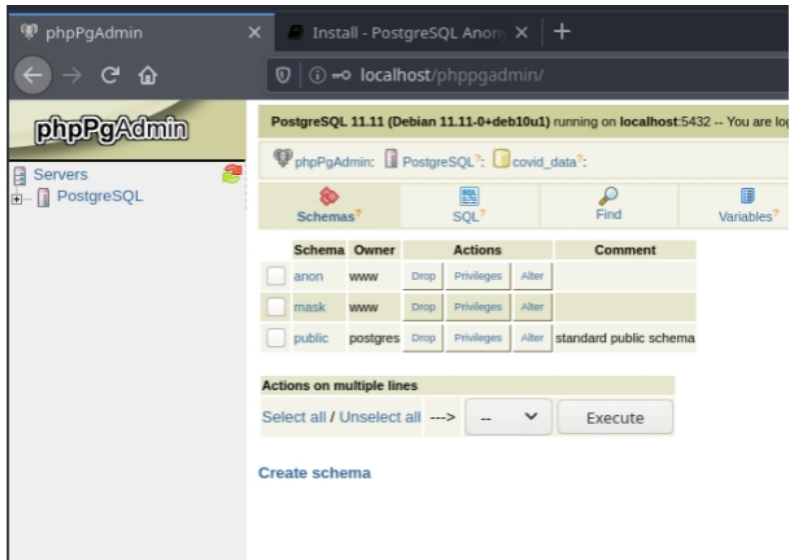


Рисунок 3.7 – Структура БД зі схемами «mask» та «anon»

Схема «mask» – це тимчасова схема, яка необхідна для компонування маскованих та немаскованих даних, які за запитом отримують користувачі. Схема «anon» вміщує в собі таблиці із даними, які будуть віддаватись натомість конфіденційних даних (рис. 3.8).

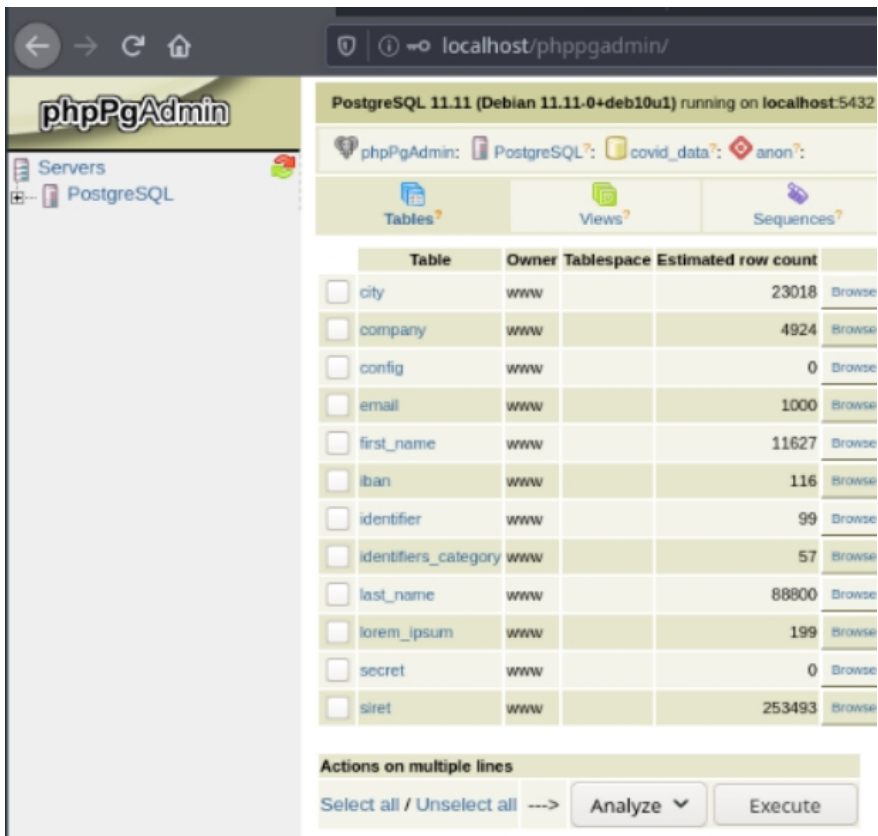


Рисунок 3.8 – Перелік таблиць схеми «anon»

Певні таблиці цієї схеми існують лише як конфігураційні файли для розширення (із значенням кількості рядків 0). Усі інші таблиці вміщують в собі першу колонку з даними: назви міст, перелік прізвищ, перелік імен, поштових скриньок, банківських номерів тощо.

У даному випадку в цьому розширенні використовується динамічний тип маскування, тобто підміняється відповідь на певний запит. Статичний тип маскування – змінює всі дані, що захищаються одноразово й безповоротно. Такий тип варто використовувати, якщо базу необхідно надати для розробки певного застосунку, який буде використовувати цю БД (для перевірки на максимально відповідних до реальності даних).

Згідно з такою парадигмою необхідно створити користувача, який буде бачити динамічно замасковані дані (рис. 3.9).

Step 1 : Activate the dynamic masking engine

```

=# CREATE EXTENSION IF NOT EXISTS anon CASCADE;
=# SELECT anon.start_dynamic_masking();

```

Step 2 : Declare a masked user

```

=# CREATE ROLE skynet LOGIN;
=# SECURITY LABEL FOR anon ON ROLE skynet
-# IS 'MASKED';

```

Step 3 : Declare the masking rules

```

SECURITY LABEL FOR anon ON COLUMN people.name
IS 'MASKED WITH FUNCTION anon.random_last_name()';

SECURITY LABEL FOR anon ON COLUMN people.phone
IS 'MASKED WITH FUNCTION anon.partial(phone,2,$$*****$$,2)';

```

Step 4 : Connect with the masked user

```

=# \! psql peopledb -U skynet -c 'SELECT * FROM people;'
 id | firstname | lastname | phone
-----+-----
T1 | Sarah    | Stranahan | 06*****11
(1 row)

```

Рисунок 3.9 – Додавання користувача та правил динамічного маскування

Згідно з такими положеннями необхідно створити роль reader із застосуванням правил маскування. Серед функцій маскування було обрано наступні функції:

- anon.random_zip();
- anon.fake_city();
- anon.fake_first_name();
- anon.fake_last_name();
- anon.fake_last_name();
- anon.partial().

Функція anon.partial() використовувалась для часткового скремблювання телефонних номерів.

4 ПРОЄКТУВАННЯ ІНТЕРФЕЙСІВ СИСТЕМИ

4.1 Обґрунтування вибору типу інтерфейсів оперативно-інформаційної системи

Інформаційну систему призначено для реєстрації випадків захворювання на COVID-19 та відображення статистичних показників захворюваності.

Базу даних для інформаційної системи створено у СКБД PostgreSQL версії 11.11. Для зручності доступу до бази даних запропоновано створити вебсайт.

Загальні переваги використання вебсайту:

- програмні засоби інформаційної системи встановлюються та підтримуються лише на сервері. На будь-яких інших комп'ютерах всесвітньої мережі Інтернет додатково нічого встановлювати не потрібно. Необхідно мати в операційній системі лише працездатний браузер. Ніяких дій не потрібно здійснювати при оновленні комп'ютерів чи заміні операційної системи. Усе це суттєво спрощує упровадження та обслуговування системи;

- відсутня необхідність «прив'язки» лікаря та відвідувача до певного робочого місця. Лікар, який має відповідні права може працювати із інформаційною системою з будь-якого комп'ютера всесвітньої мережі;

- вебсайт дозволить вирішити проблему віддаленості лікарень та пунктів первинної медичної допомоги, оскільки вони можуть функціонувати навіть в умовах мережі низької пропускної здатності;

- так як розроблювана інформаційна система має два інтерфейси, один для пересічного користувача, а другий для оператора, якій має увійти до системи за наявності для нього авторизаційного запису, то створення вебсайту зменшить витрати часу на створення подібного інтерфейсу та спростить початковий код розробки.

4.2 Структура вебсайту оперативно-інформаційної системи

Ієрархічну структуру вебсайту зображено на рисунку 4.1, а файлову структуру на рисунку 4.2.

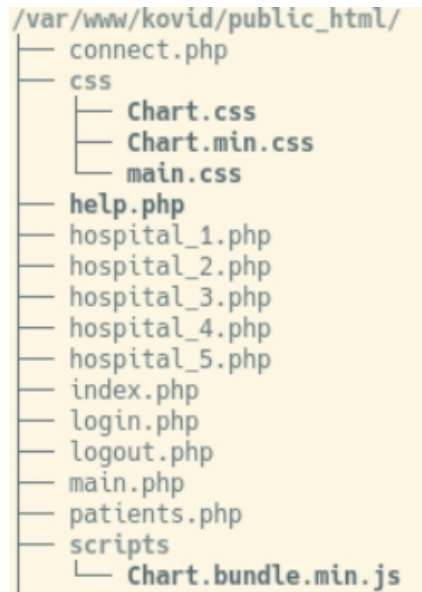


Рисунок 4.1 – Ієрархічна структура вебсайту

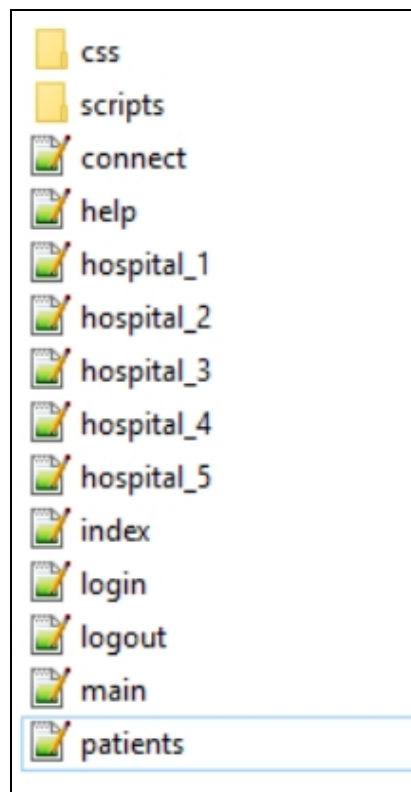


Рисунок 4.2 – Файлова структура вебсайту

Початковий код вебсайту оперативно-інформаційної системи наведено у додатку В.

У структурних розділах `css` та `scripts` (рис. 4.1) містяться файли. Файл `main.css` – це головний файл-стилю всього сайту. Зміни в ньому впливають на відображення інтерфейсів.

Інші файли структури – це файли ресурсів для використання графіків та діаграм бібліотеки `ChartJS` [15]. Вони необхідні для коректного відображення графіків.

Опис інших файлів структури вебсайту:

- а) `connect.php` – файл для з'єднання вебсайту із БД;
- б) `help.php` – файл, що генерує сторінку зворотнього зв'язку;
- в) `hospital_(1–5).php` – файли сторінок, що генерують відображення статистики для лікарень від 1 до 5 відповідно;
- г) `index.php` – головна сторінка, вміщує в собі панель навігації, має посилання до інших сторінок;
- д) `login.php` – сторінка входу користувачів;
- ж) `logout.php` – файл-функція завершення сеансу користувача;
- к) `main.php` – вміст головної сторінки, статистика загальна по місту;
- л) `patients.php` – файл, що містить усі запити та розташування графіків, форм, таблиць із клавішами керування.

Головна сторінка фактично складається із двох сторінок `index.php` (додаток В.1) та `main.php` (додаток В.11). Це було досягнуто використанням `$_GET` запиту та функції включення сторінки у вміст іншої `include()`. Функція `<?php include basename($page).' .php';?>` додає вміст сторінки у тіло `index.php`, перед цим додаючи розширення `«.php»`, адже змінна `$page` отримує необхідні назви через глобальну змінну `$_GET`. Список сторінок реалізований посиланнями у вигляді: `<a href='?page=...'>....</a>`. Синтаксис `«?page=»` вказує браузеру яке значення треба передати через змінну `$_GET` [16].

Панель керування видозмінюється, в залежності від того чи пройшов користувач аутентифікацію чи ні. Реалізовано це через перевірку глобальної змінної `$_SESSION` на наявність в цьому масиві певних даних. В цю змінну нічого не буде писатися, якщо не почати сеанс командою `session_start()` [16]. Ця команда має бути у кожному файлі вебсайту, аби можна було використовувати запис та читання зі змінної.

Щодо вкладеності одних файлів у інші варто зазначити, що файли `hospital_*.php` вміщують у себе лише деякі змінні, які передаються до `patients.php` і тільки остання сторінка відображає сторінку лікарні. Так вказуються значення `$location` та `$doctor` для авторизованих користувачів, щоб надалі при перевірці давати доступ до функцій додавання, редагування та видалення записів пацієнтів (рис. 4.3).

```
<div>
    <h1> Інформація про пацієнтів та стан лікарні №1
</div>
<?php $location = 'hospital_1';
if (isset($_SESSION['use'])) {
    $doctor = $_SESSION['use'];
};
require("patients.php");
?>
```

Рисунок 4.3 – Змінні файлу `hospital_1.php`

Щодо файлів `main.php` та `patients.php`, то ці файли дуже схожі за своїм змістом. В них загальна частина – це відображення графіків та діаграм. Відмінності полягають у наступному:

- шаблонах запитів;
- наявності форми для додавання пацієнта;
- наявності таблиці з переліком пацієнтів.

Різницю шаблонів запитів у файлах `main.php` та `patients.php` відображено на рисунку 4.4.

```

$query_patients_week = "select income_date as date, count(id) as no_of_rows
                        from patients where income_date >=current_date-7
                        group by income_date order by income_date;";
a)

$query_patients_week = "select income_date as date, count(id) as no_of_rows
                        from patients where location ='$location' and income_date >=current_date-7
                        group by income_date order by income_date;";
б)

```

a – main.php; б – patients.php

Рисунок 4.4 – Різниця запитів до БД

Відмінності у запитах також включають в себе сортування за умовою: “where location=`\$location`”.

У загальному випадку однакова частина полягає у розташуванні об’єктів (графіків) на сторінці та процедурах отримання і відображення запитів. Тому надалі буде наведено приклад у файлі patients.php (додаток В.12), де наведено повний цикл запиту та використання скрипту ChartJS для відображення на графіку показників кількості людей, які захворіли цього тижня [15].

За допомогою циклу розділюється на менші масиви значення із одного великого масиву даних БД, яке отримується за допомогою функцій pg_query() та pg_fetch_assoc(). Так як графік відображається за допомогою бібліотеки ChartJS, то використовується мова JS відповідно до тегів <script>.

З додатку В.12 можна зробити висновки:

- змінна ctx вказує на місце розташування графіку за ідентифікатором;
- тип графіку вказано як стовпчики (bar);\$
- графік призначено локальній змінній, яка наслідує клас Chart із бібліотеки ChartJS. Усі параметри графіку передаються як масив аргументів при впровадженні класу.

Параметр labels відповідає за масив даних, які вказують на дату надходження пацієнта (у межах тижня), параметр datasets передає значення data та параметри відображення для цих назв (labels), а також підписує графік за назвою label.

Інші графіки так само отримують значення, змінюється лише вид відображення графіку. У процесі проектування було використано: bar, pie, line. Для числового відображення було створено анімацію за допомогою JS. На рисунку 4.5 наведено алгоритм відображення значень від 0 до необхідного значення змінної.

```

par = document.getElementById('counter');
count = par.textContent;
i = 0;

function counter() {
    if (i < count) {
        i++;
    }
    par.textContent = i;
}
setInterval(counter, 1);

par2 = document.getElementById('counter2');
count2 = par2.textContent;
b = 0;

```

Рисунок 4.5 – Анімація на JS щодо зростання значень від 0 до верхньої шкали змінної

Таблиця та форми зворотного зв'язку побудовані завдяки звичним ресурсам HTML та скриптів PHP. На рисунку 4.6 зображено фрагмент коду, де використовується функція mail() щодо надсилання листа до оператора із заповненими даними форми [17].

```

<?php
if (isset($_POST['submit'])) {
    if (isset($_POST['submit'])) {
        $message = $_lName.' '.$fName.' '.$mName.' '.$phone.' '.$email.' '.$gender.' '.$comment;
        $message.= $_POST['symptom1'].' '.$_POST['symptom2'].' '.$_POST['symptom3'].' '.$_POST['symptom4'].'
        '.$_POST['symptom5'].' '.$_POST['symptom6'].' '.$_POST['symptom7'].' '.$_POST['symptom8'].'
        '.$_POST['symptom9'].' '.$_POST['symptom10'].' '.$_POST['symptom11'].' '.$_POST['symptom12'].'
        '.$_POST['symptom13'].' '.$_POST['symptom14'];
        $to      = 'admin@diplomtest.tk';
        $subject = 'Screening form'.date('YmdHm');
        $headers = 'X-Mailer: PHP/' . phpversion();

        mail($to, $subject, $message, $headers);
    }
}
?>

```

Рисунок 4.6 – Функція відправки повідомлення поштою на сторінці help.php

5 КЕРІВНИЦТВО КОРИСТУВАЧА

5.1 Призначення програми

Призначення оперативно-інформаційної системи – своєчасно реєструвати лікарями випадки захворювання на COVID-19, зберігати у базі даних та відображати у реальному режимі часу статистичні діаграми.

У процесі розробки оперативно-інформаційної системи було створено базу даних, яка дозволить зберігати інформацію щодо випадків захворювання на COVID-19, виконувати статистичні розрахунки захворюваності за визначеними регіонами та відображати результати розрахунків на графіках вебсторінки.

Базу даних створено у СКБД PostgreSQL версії 11.11.

У процесі створення вебінтерфейсів інформаційної системи було застосовано наступні мови: HTML 5, PHP, JS, технологію CSS, бібліотеку ChartJS та методи JSON.

5.2 Характеристики програми

Оперативно-інформаційну систему під назвою «КОВІД-19» розроблено з використанням найсучасніших технологій та призначено для роботи у браузерях будь-яких операційних систем.

У якості зберігання конфіденційної інформації застосовано розширення до СКБД PostgreSQL під назвою PostgreSQL-Anonymizer, що дозволяє маскувати дані на стороні серверу.

Для захисту від шкідливих дій та спотворення інформації реалізовано технологію розмежування доступу користувачів з використанням гешування пароллю алгоритмом SHA-256 на стороні клієнта.

Інформаційну систему адаптовано для розгортання на центральному сервері з доступом від безлічі клієнтських вузлів, що може мати обмеження тільки можливостями СКБД PostgreSQL версії 11.11.

Розроблювана інформаційна система дозволить виконувати такі функції:

- збір даних щодо захворюваності на COVID-19 у будь-якій регіональній структурі: область, район міста або села, з будь-якої клініки, лабораторії чи станції первинної допомоги за наявності доступу до всесвітньої мережі Інтернет;

- можливість структурної організації децентралізованого ієрархічного надходження даних до головного серверу, що підвищить швидкодію інформаційної системи та значно зменшить використання ресурсів на клієнтських вузлах;

- можливість розширення кількості варіантів особливо небезпечних хвороб з однієї COVID-19 до 40 або навіть і більше без суттєвих переробок програмного коду;

- зберігання у системі розширеного переліку показників для подальшого додаткового дослідження медпрацівниками;

- можливість організації взаємодії оперативно-інформаційної системи з центральною медичною базою даних Міністерства охорони здоров'я України;

- додаткова функція зворотнього зв'язку для пацієнтів з підозрілими симптомами;

- простий доступ до системи через браузер без потреби встановлення програмного забезпечення;

- моніторинг зі зміненнями у реальному часі актуальних статистичних даних щодо кількості хворих на COVID-19, померлих від ускладнень та інше;

- відображення графіків: «Загальна кількість інфікованих», «Кількість інфікованих по днях», «Загальна кількість померлих», «Кількість смертей по днях», «Кількість видужуючих по днях» тощо.

Головною відмінністю розроблюваної оперативно-інформаційної системи є реалізація онлайн-моніторингу за районами або лікарнями міста Запоріжжя та Запорізької області.

5.3 Звертання до програми

Для роботи з оперативно-інформаційною системою не має потреби у встановленні будь-яких компонентів крім стандартного браузера у складі операційної системи.

Лікарі та відвідувачі входять до сайту через браузер по доменному імені або за IP-адресою. Адміністратор БД, має можливість змінювати важливі налаштування тільки якщо він знаходиться в локальній мережі серверу. Це є одним з елементів додаткового захисту від зламу.

Основними елементами інтерфейсу, які бачить будь-який користувач є панель навігації із розділами:

- «Головна сторінка»;
- «Підтримка»;
- «Більше»;
- кнопка входу «Увійти» (або виходу – «Вихід», якщо користувач вже увійшов до ситеми).

Головну сторінку вебсайту зображено на рисунку 5.1.



Рисунок 5.1 – Головна сторінка вебсайту (без авторизації)

При натисканні «Головна сторінка» відкривається одноіменна сторінка (сторінка за замовчанням при входу до сайту). На цій сторінці розташовані графіки «Поденна кількість захворівших на день», «Поденна кількість захворівших на місяць», «Загальна кількість захворівших», діаграма «Захворіло сьогодні», діаграма «Одужало сьогодні», діаграма «Одужали всього», а також панель навігації по розділах сайту («Підтримка», «Більше»), кнопка авторизації «Увійти/Вихід» (рис. 5.1).

Розділ «Більше» складається з таких підрозділів: «Лікарня 1», «Лікарня 2», «Лікарня 3», «Лікарня 4», «Лікарня 5» (рис. 5.2).

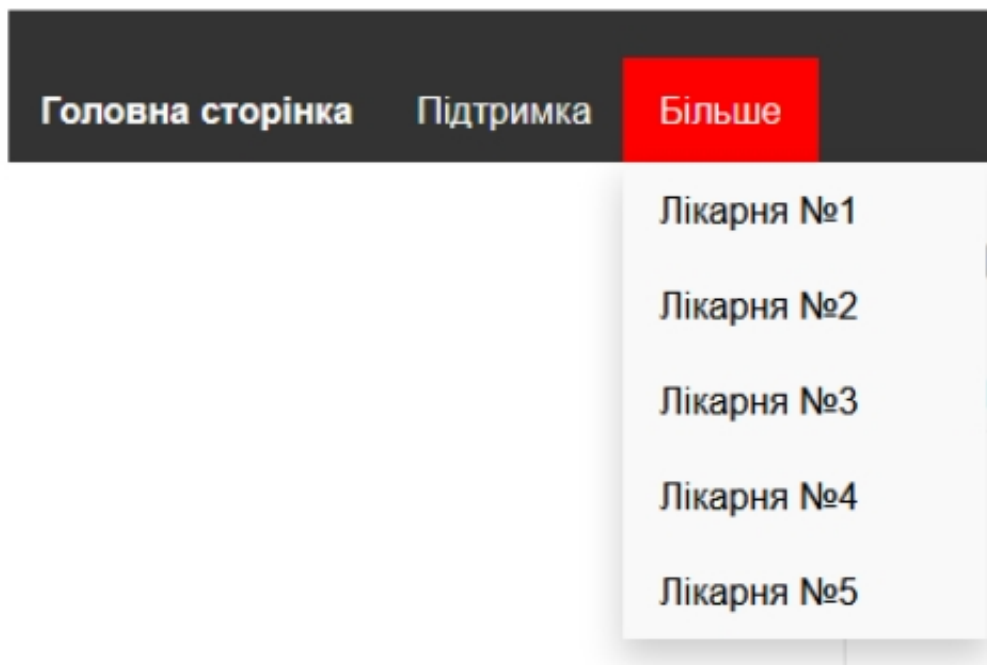


Рисунок 5.2 – Розділ «Більше» з переліком сторінок

Усі ці підрозділи в свою чергу мають наступний зміст: графіки «Поденна кількість захворівших на день», «Поденна кількість захворівших на місяць», «Загальна кількість захворівших», діаграма «Захворіло сьогодні», діаграма «Одужало сьогодні», діаграма «Одужали всього», таблиця БД з повною інформацією про пацієнтів (в залежності від прав доступу – замасковану чи ні). Усі графіки та діаграми на цих сторінках вказують на інформацію виключно з локально вказаних лікарень.

Сторінка «Підтримка» запускає для заповнення форму зворотнього зв'язку (рис. 5.3).

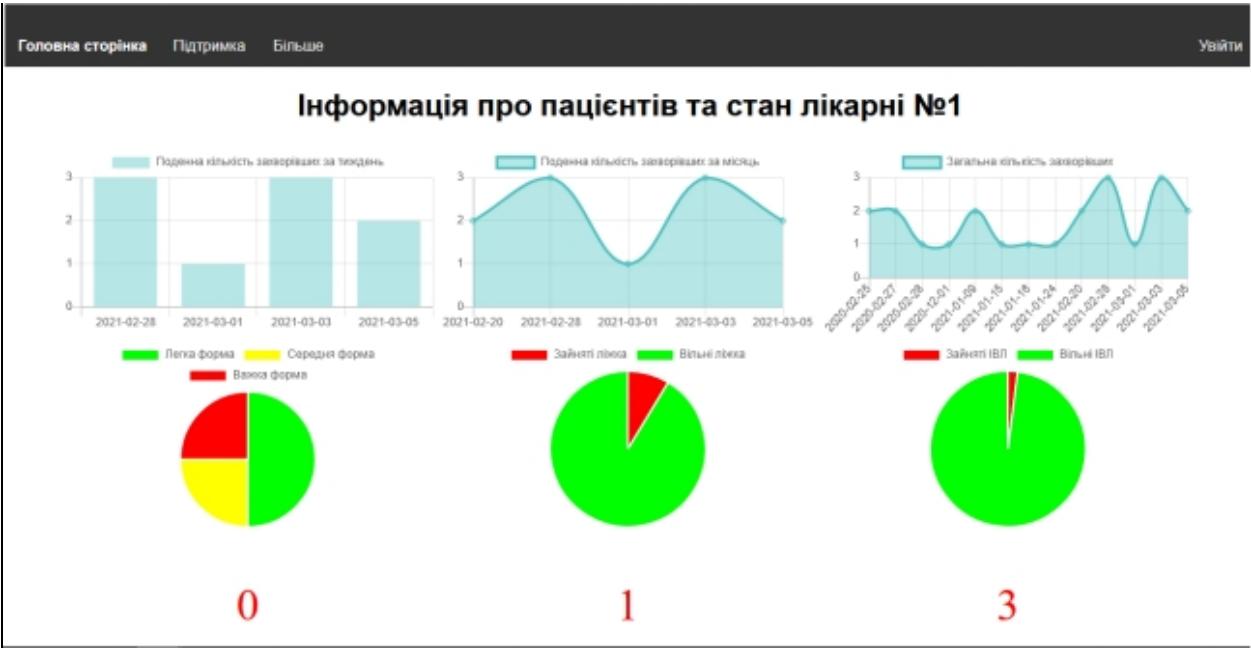
Залиште свій запит та питання тут

Будь ласка, вкажіть максимально правдиву та достовірну інформацію

Прізвище	Провище	
Ім'я*	Ім'я	
По-батькові	По-батькові	
Електронна ☐ Лихоманка скринька		
joe@mail.com		Номер телефону*
+380123456789		☐ З'явився кашель
☐ Погіршення хронічного кашлю ☐ Затруднене дихання ☐ Біль у горлі ☐ Труднощі з ковтанням ☐ Зменшення або втрата смаку або запаху ☐ Головні болі ☐ Незрозуміла втома або м'язові болі ☐ Біль у животі ☐ Нудота або блювання ☐ Почервоніння очей(кон'юнктивіт) ☐ Нежить ☐ Закладеність носа без інших відомих причин		
Вкажіть ваш запит чи інші симптоми		
Стать*: ☐ Жінка ☐ Чоловік		
Надіслати запит		
* Обов'язкові поля для заповнення		

Рисунок 5.3 – Форма зворотнього зв'язку на сторінці «Підтримка»

На рисунку 5.4 відображено вигляд головної сторінки для неавторизованого (звичайного) відвідувача сайту.



Дата надходження	Стан здоров'я	ПІБ	Вік	Адреса проживання	Телефон	Кількість контактних	Захворіло контактних	ІВЛ	Дата виписки	Стан на дату виписки
2021-03-05	Легка форма	Elsio Baltazor	0	58706 Winchester Vestibulum condimentum nisi	*****	0	0	Ні	2021-03-06	Легка форма
2021-03-05	Легка форма	Roly Verjan	0	87463 Fredericton Mauris pulvinar fermentum	*****	0	0	Ні		
2021-03-03	Важка форма	Adair Macia		71714 Rubi Nullam id enim		0	0	Ні		
2021-03-03	Легка форма	Michael Roghair	4	63091 Lexington-Fayette Integer viverra consectetur	*****	6	4	Так	2020-03-03	Легка форма
2021-03-03	Легка форма	Slian Henderosn	4	80641 Tangzhai Nulla eismod sit	+38*****23	0	0	Ні		
2021-03-01	Легка форма	Sitara Kiapp	8	49881 Bhuvanagiri Phasellus metus ex	*****	0	0	Ні	2021-03-03	NULL
2021-02-28	Важка форма	Flo Rasp		94597 Kolding Etiam lorem turpis		0	0	Ні	2021-03-04	Одужав(ла)
2021-02-28	Середня форма	Lashawn Holle	8	82652 Roosendaal Nullam malesuada arcu	*****	0	0	Ні		
2021-02-28	Легка форма	Lashawn Burtchell	0	43162 Marawi City Curabitur metus libero	*****	0	0	Ні	2021-03-01	Помер(ла)
2021-02-28	Легка форма	Berjouhi	4	85881 Panshan Nullam id	*****	0	0	Ні	2021-03-04	Легка форма

Рисунок 5.4 – Вигляд сторінки лікарні з маскуванням на прикладі «Лікарня №1»

Кнопка авторизації перенаправляє на сторінку входу (рис. 5.5).

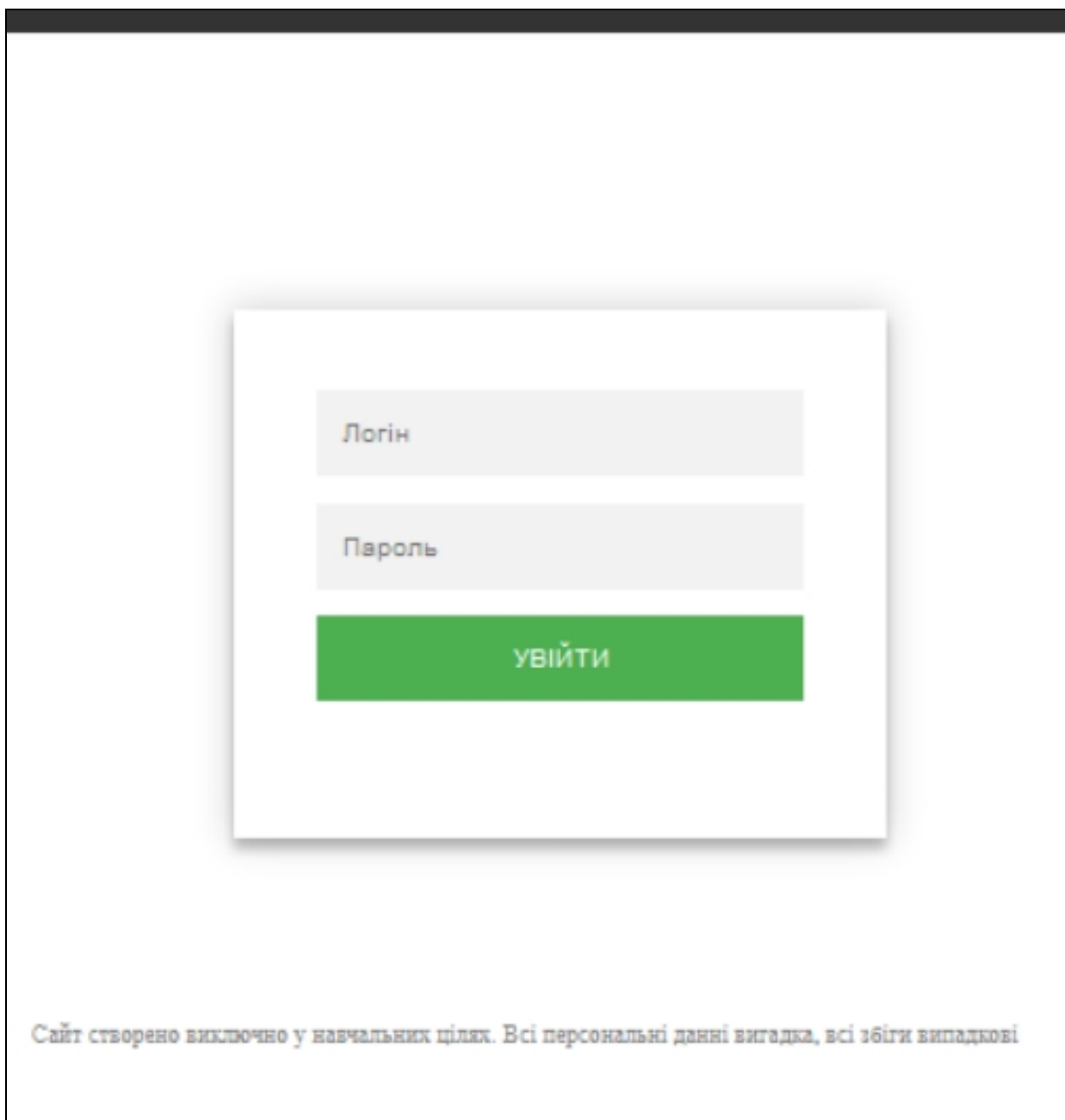


Рисунок 5.5 – Сторінка входу користувачів сайту

На можливості розширеного доступу та дії користувачів вебсайту впливає можливість входу. Як приклад варто зазначити вигляд таблиць із персональними даними (рис. 5.6–5.7). На всіх цих сторінках вказується одна й та сама таблиця з даними пацієнтів, але при цьому відвідувач бачить різні значення. Істинні значення будуть тільки лише коли користувач увійде до інформаційної системи.

Дата надходження	Стан здоров'я	ПІБ	Вік	Адреса проживання	Телефон	Кількість контактних	Захворіло контактних	ІВЛ	Дата виписки
2021-03-12	Важка форма	Irvine Prothro	4	69410 Chaochu Vestibulum ante ipsum	+38*****89	0	0	Ні	
2021-03-11	Легка форма	Berjouhi Crosdale	56	18215 Sri Mādhapur Donec commodo varius	+38*****23	0	0	Ні	
2021-03-10	Легка форма	Elmas Cooperwood	11	84694 Tutayev Phasellus sollicitudin placerat	+38*****23	2	2	Так	
2021-03-10	Легка форма	Constance Hobert	11	41136 Loves Park Vestibulum eleifend ultricies	+38*****23	2	2	Так	

Рисунок 5.6 – Вигляд таблиці пацієнтів для незареєстрованого користувача

Дата надходження	Стан здоров'я	ПІБ	Вік	Адреса проживання	Телефон	Кількість контактних	Захворіло контактних	ІВЛ	Дата виписки
2021-03-12	Важка форма	Höng Bordner	4	89662 Proddatūr Quisque posuere, lectus	+38*****89	0	0	Ні	
2021-03-11	Легка форма	Dijana Rusell	56	90384 Sri Mādhapur Mauris pulvinar fermentum	+38*****23	0	0	Ні	
2021-03-10	Легка форма	Sitara Waiters	11	49221 Arsen'yev Donec consequat metus	+38*****23	2	2	Так	
2021-03-10	Легка форма	Chanda Gattis	11	71138 San Antonio de Los Altos Curabitur molestie ac	+38*****23	2	2	Так	

Рисунок 5.7 – Вигляд таблиці пацієнтів для незареєстрованого користувача, але з нової сесії

Вхід під акаунтом адміністратора надає права від простого редагування та внесення даних до отримання правдивих персональних даних (рис. 5.8–5.10). Наприклад, внесення нової інформації про стан захворілих, заповнення форми для додавання нового пацієнта, отримання персональних даних пацієнтів тощо.

Дата надходження	Стан здоров'я	ПІБ	Вік	Адреса проживання	Телефон	Кількість контактних	Захворіло контактних	ІВЛ	Дата виписки	Стан на дату виписки
2021-03-12	Важка форма	Протагонист Вася	4		+380123456789	0	0	Ні		Редагувати
2021-03-11	Легка форма	тест	56	іаіаіа іаіаіа	+380123123123	0	0	Ні		Редагувати
2021-03-10	Легка форма	sdfgsdfgsdfg sdfgsdfg sdfg	11		+380123124123	2	2	Так		Редагувати
2021-03-10	Легка форма	sdfgsdfgsdfg sdfgsdfg sdfg	11		+380123124123	2	2	Так		Редагувати

Рисунок 5.8 – Вигляд таблиці пацієнтів для лікаря, який веде цих пацієнтів, або для головного врача

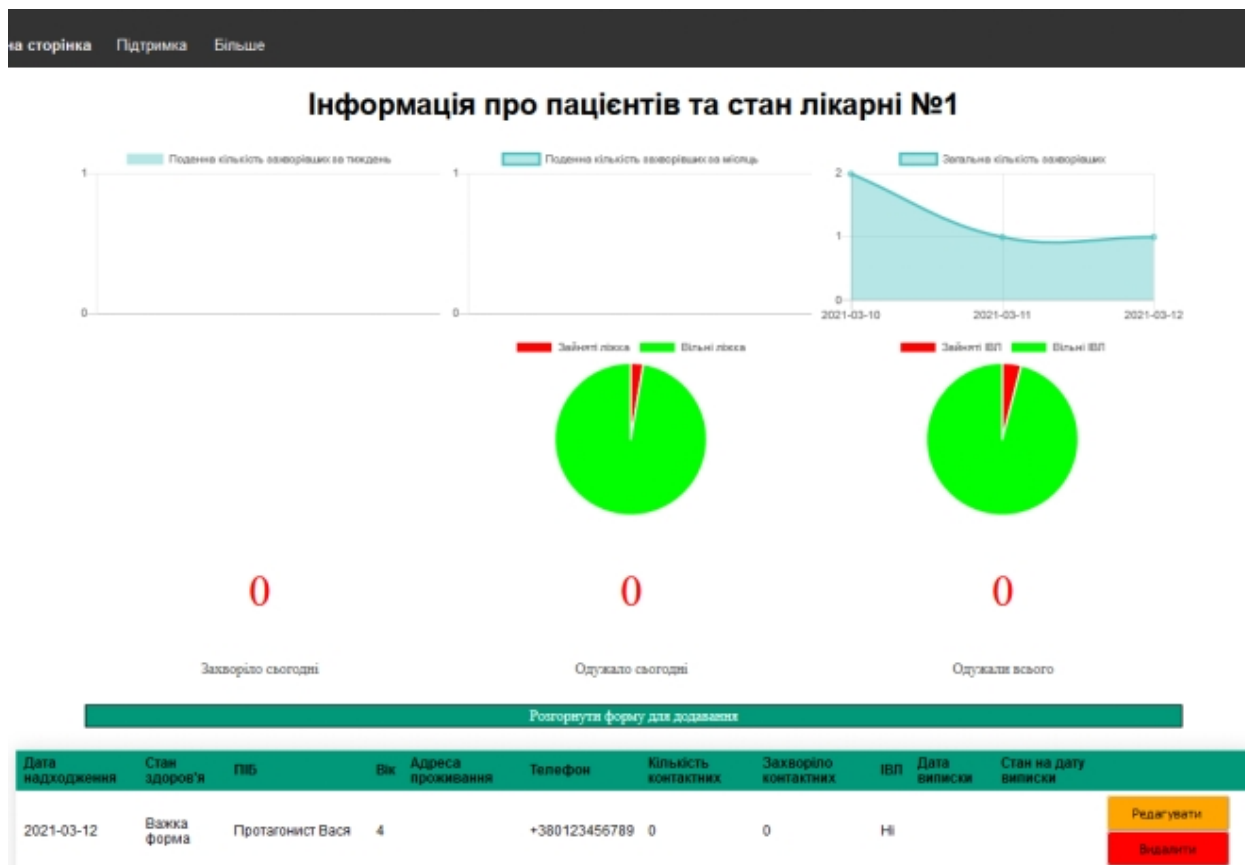


Рисунок 5.9 – Додатковий функціонал, що з'являється на сторінці після входу лікаря

Згорнути форму для додавання

Форма для додавання нового пацієнта

Будь-ласка заповніть форму для додавання запису до таблиці нижче

Дата надходження	Стан здоров'я	ПІБ	Вік пацієнта	Адреса проживання чи реєстрації	Номер телефону
2020-04-08	Легка форма		0		+380123456789
Кількість контактних осіб	Кількість захворілих контактних	Підключений до ІВЛ	Дата виписки	Стан здоров'я на момент виписки	
0	0	Ні	2020-04-08		Додати запис

Рисунок 5.10 – Вигляд форми для додавання нового пацієнта

ВИСНОВКИ

У результаті виконання дипломної кваліфікаційної роботи бакалавра було проведено аналіз існуючих аналогічних розробок, визначені вимоги до проектування, розроблено структурну та функціональну схеми оперативно-інформаційної системи, створено концептуальну та реляційну модель бази даних, виконано її нормалізацію та побудовано логічну схему БД, обрано середовище проектування – СКБД PostgreSQL версії 11.11, створено фізичну модель БД з таблицями, які побудовані за допомогою інструментального середовища phpPgAdmin, встановлено до СКБД розширення PostgreSQL-Anonymizer, яке додає можливість маскуванню даних з метою збереження конфіденційної інформації, для створення вебінтерфейсів оператора та клієнта застосовано сучасні мови HTML 5, PHP, JS, CSS, бібліотеку ChartJS та методи JSON.

Розроблена інформаційна система дозволить виконувати такі функції:

- збір даних щодо захворюваності на COVID-19 у будь-якій регіональній структурі Запорізької області: область, район міста або села, з будь-якої клініки, лабораторії чи станції первинної допомоги за наявності доступу до всесвітньої мережі Інтернет;
- можливість розширення кількості варіантів особливо небезпечних хвороб з однієї COVID-19 до 40 або навіть і більше без суттєвих переробок програмного коду;
- додаткова функція зворотнього зв'язку для пацієнтів з підозрілими симптомами;
- моніторинг зі зміненнями у реальному часі актуальних статистичних даних щодо кількості хворих на COVID-19, померлих від ускладнень та інше;
- відображення графіків: «Загальна кількість інфікованих», «Кількість інфікованих по днях», «Загальна кількість померлих», «Кількість смертей по днях», «Кількість видужуючих по днях» тощо.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Зараження коронавірусом у різних країнах [Електронний ресурс]. – Режим доступу: <https://index.minfin.com.ua/reference/coronavirus/geography/>
2. Наказ Міністерства охорони здоров'я України № 133 від 19.07.1995 із змінами згідно з Наказом МОЗУ № 521 від 25.02.2020 «Про затвердження Переліку особливо небезпечних, небезпечних інфекційних та паразитарних хвороб людини і носійства збудників цих хвороб» [Електронний ресурс]. – Режим доступу: <https://zakon.rada.gov.ua/rada/show/v0133282-95>
3. Система моніторингу поширення епідемії коронавірусу [Електронний ресурс]. – Режим доступу: <https://covid19.rnbo.gov.ua>
4. Концептуальне проектування БД [Електронний ресурс]. – Режим доступу: https://studopedia.su/13_103235_kontseptualne-proektuvannya-bd.html
5. Осипов, Д. Л. Базы данных и Delphi. Теория и практика [Текст]/ Д. Л. Осипов. – СПб.: БХВ-Петербург, 2011. – 752 с.: ил.
6. Нормалізація баз даних [Електронний ресурс]. – Режим доступу: <http://ukrbukva.net/92785-Normalizaciya-bazdannyh.html>
7. Організація баз даних та знань. Тема 3 – Проектування бази даних [Електронний ресурс]. – Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:89b3d175c06a6b137e410cb14821d0e94549ad5a/20151013153156/44233/index.html
8. Хансен, Г. Базы данных: разработка и управление: пер. с англ. [Текст]/Г. Хансен, Дж. Хансен. – М.: Бином, 1999. – 196 с.
9. Васвани, В. MySQL: использование и администрирование = MySQL Database Usage & Administration [Текст]/В. Васвани. – М.: Питер, 2011. – 368 с.
10. Суэринг, С. PHP и MySQL. Библия программиста, 2-е издание = PHP 6 and MySQL 6 Bible [Текст]/С. Суэринг, Т. Конверс, Дж. Парк. – М.: Диалектика, 2010. – 912 с.

11. Документация к PostgreSQL 11 [Электронный ресурс]. – Режим доступа: <https://postgrespro.ru/docs/postgresql/11/index>
12. Postgres поддерживает понятие перечислений (enum) [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/353556/>
13. Перечислимый тип и PostgreSQL [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/company/domclick/blog/525530/>
14. Розширення PostgreSQL-Anonymizer [Електронний ресурс]. – Режим доступу: <https://postgresql-anonymizer.readthedocs.io/en/stable/>
15. Бібліотека графічного відображення Chart.js 3.2.1 [Електронний ресурс]. – Режим доступу: <https://www.chartjs.org>
16. Зарезервовані змінні у PHP [Електронний ресурс]. – Режим доступу: <https://www.php.net/manual/en/reserved.variables.php>
17. Відправлення поштою Email у PHP [Електронний ресурс]. – Режим доступу: <https://mailtrap.io/blog/php-email-sending/>

ДОДАТОК А
Технічне завдання

Вступ

Метою дипломної кваліфікаційної роботи бакалавра є створення оперативно-інформаційної системи для аналізу захворювань на COVID-19, яка дозволить підвищити ефективність протиепідемічних заходів щодо боротьби проти смертельної хвороби.

Розроблена інформаційна система дозволить виконувати такі функції:

- збір даних щодо захворюваності на COVID-19 у будь-якій регіональній структурі Запорізької області: область, район міста або села, з будь-якої клініки, лабораторії чи станції первинної допомоги за наявності доступу до всесвітньої мережі Інтернет;
- можливість розширення кількості варіантів особливо небезпечних хвороб з однієї COVID-19 до 40 або навіть і більше без суттєвих переробок програмного коду;
- додаткова функція зворотнього зв'язку для пацієнтів з підозрілими симптомами;
- моніторинг зі зміненнями у реальному часі актуальних статистичних даних щодо кількості хворих на COVID-19, померлих від ускладнень та інше;
- відображення графіків: «Загальна кількість інфікованих», «Кількість інфікованих по днях», «Загальна кількість померлих», «Кількість смертей по днях», «Кількість видужуючих по днях» тощо.

A.1 Підстава для розробки

Підставою для розробки послужило завдання на дипломну кваліфікаційну роботу бакалавра за темою: «Розробка оперативно-інформаційної системи для аналізу захворювань на особливо небезпечні хвороби», затверджену наказом по Національному університету «Запорізька політехніка» № 103 від 30 березня 2021 року.

A.2 Призначення розробки

Оперативно-інформаційну систему призначено для своєчасної реєстрації лікарями випадків захворювання на COVID-19, зберігання у базі даних та відображення у реальному часі статистичних діаграм.

A.3 Вимоги до програми

A.3.1 Вимоги до структури програми

Інформаційна система повинна мати такі структурні елементи:

- операторська частина (призначено для лікаря, якій вносить дані щодо захворюваностей);
- клієнтська частина (для відвідувачів сайту інформаційної системи);
- адмінська частина (для головного адміністратора сайту інформаційної системи, який авторизує лікарів та робить налаштування серверу);
- центральний сервер, якій містить базу даних та модуль аутентифікації для розгалуження доступу користувачів до БД.

A.3.2 Вимоги до функціональних характеристик

Виходячи з постановки задачі, було сформовано основні функціональні вимоги до програми.

Розроблювана інформаційна система має дозволяти виконувати такі функції:

- збір даних щодо захворюваності на COVID-19 у будь-якій регіональній структурі: область, район міста або села, з будь-якої клініки, лабораторії чи станції первинної допомоги за наявності доступу до всесвітньої мережі Інтернет;

- можливість структурної організації децентралізованого ієрархічного надходження даних до головного серверу, що підвищить швидкодію інформаційної системи та значно зменшить використання ресурсів на клієнтських вузлах;
- можливість розширення кількості варіантів особливо небезпечних хвороб з однієї COVID-19 до 40 або навіть і більше без суттєвих переробок програмного коду;
- зберігання у системі розширеного переліку показників для подальшого додаткового дослідження медпрацівниками;
- можливість організації взаємодії оперативно-інформаційної системи з центральною медичною базою даних Міністерства охорони здоров'я України;
- додаткова функція зворотнього зв'язку для пацієнтів з підозрілими симптомами;
- простий доступ до системи через браузер без потреби встановлення програмного забезпечення;
- моніторинг зі зміненнями у реальному часі актуальних статистичних даних щодо кількості хворих на COVID-19, померлих від ускладнень та інше;
- відображення графіків: «Загальна кількість інфікованих», «Кількість інфікованих по днях», «Загальна кількість померлих», «Кількість смертей по днях», «Кількість видужуючих по днях» тощо.

Головною відмінністю розроблюваної оперативно-інформаційної системи повинна бути реалізація онлайн-моніторингу за районами або лікарнями міста Запоріжжя та Запорізької області.

А.3.3 Вимоги до зберігання даних

Створена інформаційна система дозволить зберігати та використовувати у якості статистичних даних для подальших медичних досліджень:

- кількість загальних ліжок у лікарнях;
- кількість ліжок із системами ШВЛ;
- дата прибуття пацієнта на обстеження;
- стан здоров'я пацієнта на дату прибуття;
- основні персональні дані пацієнта (П.І.Б. та інше);
- вік пацієнта;
- адреса реєстрації пацієнта;
- контактний телефон пацієнта;
- кількість контактних осіб (всього);
- кількість контактних осіб, які захворіли на COVID-19;
- гостра потреба пацієнта у апараті ШВЛ;
- дата виписки пацієнта;
- стан здоров'я пацієнта на дату виписки;
- місце розташування або проживання пацієнта;
- П.І.Б. лікаря, який реєстрував пацієнта.

А.3.4 Вимоги до інтерфейсу програми

Оперативно-інформаційна система повинна мати графічний інтерфейс для зручного використання. Графічний інтерфейс має бути дружнім щодо користувачів та ергономічним, із стартовою сторінкою. Взаємодія оперативно-інформаційної системи із користувачем повинна відбуватись через поля для введення даних.

А.4 Обґрунтований вибір апаратно-технічних засобів, операційної системи і мови програмування

Для функціонування оперативно-інформаційної системи необхідно мати будь-який комп'ютер, який під'єднан до всесвітньої мережі Інтернет із вбудованим або встановленим до будь-якої операційної системи браузером.

Рекомендовані вимоги до програмного забезпечення серверу щодо найкращої швидкодії інформаційної системи:

– операційна система: Debian GNU Linux версії 9.x або вище чи Red Hat Enterprise Linux версії 7.x або вище чи Ubuntu 16.04 LTS або версіями вище;

– встановлена СКБД PostgreSQL версії 11.x;

– встановлена PHP версії 7.x;

– встановлено Apache 2.4 із увімкненим параметром `mod_php`.

Особливих вимог до експлуатації системи не пред'являється.

A.5 Основні обмеження на установку і використання програми

Особливих вимог до експлуатації інформаційної системи не пред'являється. За виключенням актуального встановлення останніх оновлень операційної системи.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Створена оперативно-інформаційна система дозволить зберігати та використовувати у якості статистичних даних для подальших медичних досліджень:

- загальну кількість ліжок та кількість ліжок із системами ШВЛ зазначеної клініки;
- дата прибуття пацієнта на обстеження;
- стан здоров'я пацієнта на дату прибуття;
- основні персональні дані пацієнта (П.І.Б. та інше);
- вік пацієнта;
- адреса реєстрації пацієнта;
- контактний телефон пацієнта;
- кількість контактних осіб (всього);
- кількість контактних осіб, які захворіли на COVID-19;
- гостра потреба пацієнта у апараті ШВЛ;
- дата виписки пацієнта;
- стан здоров'я пацієнта на дату виписки;
- місце розташування або проживання пацієнта;
- П.І.Б. лікаря, який реєстрував пацієнта.

Для створення бази даних інформаційної системи використовувалася СКБД PostgreSQL версії 11.11. Система керування базами даних PostgreSQL встановлена на виділеному сервері та працює під операційною системою Linux.

Б.2 Функціональне призначення

Основним функціональним призначенням інформаційної системи є реєстрація лікарями випадків захворювання на COVID-19, зберігання у базі даних та відображення у реальному режимі часу статистичних діаграм.

Інформаційна система дозволить виконувати такі функції:

- збір даних щодо захворюваності на COVID-19 у будь-якій регіональній структурі Запорізької області: область, район міста або села, з будь-якої клініки, лабораторії чи станції первинної допомоги за наявності доступу до всесвітньої мережі Інтернет;
- можливість розширення кількості варіантів особливо небезпечних хвороб з однієї COVID-19 до 40 або навіть і більше без суттєвих переробок програмного коду;
- додаткова функція зворотнього зв'язку для пацієнтів з підозрілими симптомами;
- моніторинг зі зміненнями у реальному часі актуальних статистичних даних щодо кількості хворих на COVID-19, померлих від ускладнень та інше;
- відображення графіків: «Загальна кількість інфікованих», «Кількість інфікованих по днях», «Загальна кількість померлих», «Кількість смертей по днях», «Кількість видужуючих по днях» тощо.

Б.3 Опис логічної структури

Структура створеної інформаційної системи складається із чотирьох частин – операторська (лікар), клієнтська (відвідувачі), адмінська (головний адміністратор системи) та серверна частина. Серверна частина містить три компоненти:

- база даних;
- модуль аутентифікації;
- встановлене розширення під СКБД для маскування (анонімізації) даних.

Б.4 Використані технічні засоби та умови роботи програми

Б.4.1 Використані технічні засоби

Розробка оперативно-інформаційної системи та створення бази даних відбувалися на комп'ютері, де було встановлено операційну систему Linux Debian версії 10. При проектуванні бази даних у якості СКБД було застосовано PostgreSQL версії 11.11. Для створення вебінтерфейсів інформаційної системи було застосовано наступні мови: HTML 5, PHP, JS, технологію CSS, бібліотеку ChartJS та методи JSON.

Б.4.2 Умови роботи програми

Для забезпечення роботи оперативно-інформаційної системи необхідно мати будь-який комп'ютер, який під'єднан до всесвітньої мережі Інтернет із вбудованим або встановленим до будь-якої операційної системи браузером.

Для функціонування серверної частини найкращим вибором буде:

- операційна система: Debian GNU Linux версії 9.x або вище чи Red Hat Enterprise Linux версії 7.x або вище чи Ubuntu 16.04 LTS або версіями вище;
- встановлена СКБД PostgreSQL версії 11.x;
- встановлена PHP версії 7.x;
- встановлено Apache 2.4 із увімкненим параметром `mod_php`.

Б.5 Виклик, завантаження і звернення до програми

До виклику оперативно-інформаційної системи під назвою «КОВІД-19» лікарі та відвідувачі можуть перейти через браузер по доменному імені або за IP-адресою. Адміністратор БД, має можливість змінювати важливі налаштування тільки якщо він знаходиться в локальній мережі серверу.

ДОДАТОК В
Текст програми

В.1 Файл index.php

```

<?php
//error_reporting(-1);
//ini_set('display_errors',1);
header('Content-Type: text/html; charset=utf-8');
ob_start();
session_start();
$page = (isset($_GET['page'])) ? $_GET['page'] : 'main');
?>
<html>
    <meta charset="utf-8"/>
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <link href="css/main.css" rel="stylesheet"/>
    <link rel="stylesheet" type="text/css" href="css/Chart.css">
    <script src="scripts/Chart.bundle.min.js"></script>

<head>
<title>Статистика КОВІД-19</title>
</head>
<body>
<header>
<div class="navbar">
    <h1><a href="/?page=main">Головна сторінка</a></h1>
    <a href="/?page=help">Підтримка</a>
    <div class="dropdown">
        <button class="dropbtn">Більше
            <i class="fa fa-caret-down"></i>
        </button>
        <div class="dropdown-content">
            <a href="/?page=hospital_1">Лікарня №1</a>
            <a href="/?page=hospital_2">Лікарня №2</a>
            <a href="/?page=hospital_3">Лікарня №3</a>
            <a href="/?page=hospital_4">Лікарня №4</a>
            <a href="/?page=hospital_5">Лікарня №5</a>
        </div>
    </div>
    <div class="enterbtn">
<?php
if (!isset($_SESSION['use'])) {
    echo '<a href="/?page=login" >Увійти</a>';
} else {
    echo '<a href="/?page=logout">Вихід</a>';
}
?>
    </div>
</div>
</header>
<?php include basename($page).'.php';?>
</body>
</html>

```

В.2 Файл connect.php

```
<!-- THIS IS FOR CONNECTION TO POSTGRESQL DBs ONLY-->
<?php
if (!isset($_SESSION['use'])) {
    $DB_USER="reader";
    $DB_PASS="reader2021";
} else {
    $DB_USER="www";
    $DB_PASS="post171";
}
$DB_HOST="149.12.34.193";
$DB_DBNAME="covid_data";
$con = pg_connect("host=$DB_HOST port=5432 dbname=$DB_DBNAME
user=$DB_USER password=$DB_PASS")
    or die (" Не можу ініціалізувати з'єднання з базою\n");
?>
```

В.3 Файл hospital_1.php

```
<div>
    <h1> Інформація про пацієнтів та стан лікарні №1
</div>
<?php $location = 'hospital_1';
if (isset($_SESSION['use'])) {
    $doctor = $_SESSION['use'];
};
require("patients.php");
?>
```

В.4 Файл hospital_2.php

```
<div>
    <h1> Інформація про пацієнтів та стан лікарні №2
</div>
<?php $location = 'hospital_2';
if (isset($_SESSION['use'])) {
    $doctor = $_SESSION['use'];
};
include("patients.php");
?>
```

В.5 Файл hospital_3.php

```
<div>
    <h1> Інформація про пацієнтів та стан лікарні №3
</div>
<?php $location = 'hospital_3';
if (isset($_SESSION['use'])) {
    $doctor = $_SESSION['use'];
};
include("patients.php");
?>
```

B.6 Файл hospital_4.php

```
<div>
<h1>      Інформація про пацієнтів та стан лікарні №4
</div>
<?php $location = 'hospital_4';
if (isset($_SESSION['use'])) {
    $doctor = $_SESSION['use'];
};
include("patients.php");
?>
```

B.7 Файл hospital_5.php

```
<div>
<h1>      Інформація про пацієнтів та стан лікарні №5
</div>
<?php $location = 'hospital_5';
if (isset($_SESSION['use'])) {
    $doctor = $_SESSION['use'];
};
include("patients.php");
?>
```

B.8 Файл login.php

```
<?php
//Підключення до бази даних
require_once "connect.php";
$user = $pass = $result= $row="";
if(isset($_POST['login'])) // it checks whether the user
clicked login button or not
{
    $user = $_POST['user'];
    $pass = $_POST['pass'];
    $query_user="select id,password,user_role,work_loc from
users where username='$user'";
    $result=pg_query($con,$query_user);
    $row=pg_fetch_row($result);
    $db_user_id=$row['0'];
    $db_user_pass=$row['1'];
    $db_user_role=$row['2'];
    $db_user_loc=$row['3'];
    /*print_r($row);
    echo $db_user_id;
    echo $db_user_pass;
    echo $db_user_role;
    echo $db_user_loc;*/
    if($user!=NULL && $pass!=""&&$pass!=NULL&& $pass ==
$db_user_pass)
    {
        $_SESSION['use']=$user;
        $_SESSION['role']=$db_user_role;
```

```

        $_SESSION['loc']=$db_user_loc;
        //echo 'Доброго дня, вас переадресує на головну через 2
секунди';
        header('Location: ' . $_SERVER['PHP_SELF']);
        //header('Location: /');
    } else {
        echo "Не вірний пароль чи логін";
    }
}
?>
<div class="login-page">
    <div class="form">
        <form action="" method="post" class="login-form">
            <input type="text" name="user" placeholder="Логін"/>
            <input type="password" name="pass" placeholder="Пароль"/>
            <input type="submit" name="login" value="Увійти"/>
        </form>
    </div>
</div>

<style>
.login-page {
    width: 360px;
    padding: 8% 0 0;
    margin: auto;
}
.form {
    position: relative;
    z-index: 1;
    background: #FFFFFF;
    max-width: 360px;
    margin: 0 auto 100px;
    padding: 45px;
    text-align: center;
    box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0,
0, 0, 0.24);
}
.form input {
    font-family: sans-serif;
    outline: 0;
    background: #f2f2f2;
    width: 100%;
    border: 0;
    margin: 0 0 15px;
    padding: 15px;
    box-sizing: border-box;
    font-size: 14px;
}
.form input[type=submit] {
    font-family: sans-serif;
    text-transform: uppercase;
    outline: 0;
    background: #4CAF50;

```

```

width: 100%;
border: 0;
padding: 15px;
color: #FFFFFF;
font-size: 14px;
-webkit-transition: all 0.3 ease;
transition: all 0.3 ease;
cursor: pointer;
}
.form input[type=submit]:hover, .form
input[type=submit]:active, .form input[type=submit]:focus {
    background: #43A047;
}
.form .message {
    margin: 15px 0 0;
    color: #b3b3b3;
    font-size: 12px;
}
.form .message a {
    color: #4CAF50;
    text-decoration: none;
}
body {
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
</style>
<script>
    $(' .message a').click(function(){
        $('form').animate({height: "toggle", opacity: "toggle"},
        "slow");
    });
</script>

```

В.9 Файл `logout.php`

```

<?php
    //session_start();
    session_destroy();
    echo 'Ваш сеанс завершено, переадресую на попередню
сторінку';
    header('Location: ' . $_SERVER['HTTP_REFERER']);
    //header('Refresh: 0.5; URL = /');
?>

```

В.10 Файл `help.php`

```

<div>
    <h1> Залиште свій запит та питання тут
</div>
<link href="css/main.css" rel="stylesheet" />
<div>
    <?php

```

```

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["lName"])) {
    } else {
        $lName = test_input($_POST["lName"]);
    }
    if (empty($_POST["fName"])) {
        $nameErr = "Необхідне ім'я";
    } else {
        $fName = test_input($_POST["fName"]);
    }
    if (empty($_POST["mName"])) {
    } else {
        $mName = test_input($_POST["mName"]);
    }
    if (empty($_POST["email"])) {
    } else {
        $email = test_input($_POST["email"]);
    }

    if (empty($_POST["phone"])) {
        $phoneErr = "Необхідний номер телефону";
    } else {
        $phone = test_input($_POST["phone"]);
    }
    if (empty($_POST["comment"])) {
        $comment = "";
    } else {
        $comment = test_input($_POST["comment"]);
    }

    if (empty($_POST["gender"])) {
    } else {
        $gender = test_input($_POST["gender"]);
    }
}

function test_input($data)
{
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>
<div class="container">
    <p> Будь ласка, вкажіть максимально правдиву та достовірну
інформацію</p>
    <form method="post" action="" class="help-container">
        <div class="row">
            <div class="col-25">
                <label for="lName"> Прізвище</label>
            </div>

```

```

        <div class="col-75"><input type="text" name="lName"
placeholder="Прізвище"></div>
    </div>
    <div class="row">
        <div class="col-25">
            <label for="fName"> Ім'я*</label>
        </div>
        <div class="col-75"><input required type="text"
name="fName" placeholder="Ім'я"
oninvalid="this.setCustomValidity('Введіть ім\''я') "
oninput="this.setCustomValidity('')"></div>
    </div>
    <div class="row">
        <div class="col-25">
            <label for="mName"> По-батькові</label>
        </div>
        <div class="col-75"><input type="text" name="mName"
placeholder="По-батькові"></div>
    </div>
    <div class="row">
        <div class="col-25">
            <label for="email">Електронна скринька</label>
        </div>
        <div class="col-75">
            <input type="text" name="email"
placeholder="joe@mail.com" size="30">
        </div>
    </div>

    <div class="row">
        <div class="col-25">
            <label for="phone"> Номер телефону* </label>
        </div>
        <div class="col-75">
            <input required type="text" name="phone"
oninvalid="this.setCustomValidity('Введіть телефон') "
oninput="this.setCustomValidity('') " pattern="\+380([0-9]){9}"
placeholder="+380123456789>
        </div>
    </div>
    <br>
    <div class="column">
        <input type="checkbox" id="symptom1" name="symptom1"
value="Лихоманка">
        <label for="symptom1">Лихоманка</label>
        <br>
        <input type="checkbox" id="symptom2" name="symptom2"
value="З'явився кашель">
        <label for="symptom2">З'явився кашель </label>
        <br>
        <input type="checkbox" id="symptom3" name="symptom3"
value="Погіршення хронічного кашлю">

```

```

        <label for="symptom3">Погіршення хронічного
кашлю</label>
        <br>
        <input type="checkbox" id="symptom4" name="symptom4"
value="Затруднене дихання">
        <label for="symptom4">Затруднене дихання </label>
        <br>
        <input type="checkbox" id="symptom5" name="symptom5"
value="Біль у горлі">
        <label for="symptom5">Біль у горлі</label>
        <br>
        <input type="checkbox" id="symptom6" name="symptom6"
value="Труднощі з ковтанням">
        <label for="symptom6">Труднощі з ковтанням</label>
        <br>
        <input type="checkbox" id="symptom7" name="symptom7"
value="Зменшення або втрата смаку або запаху">
        <label for="symptom7">Зменшення або втрата смаку або
запаху</label>
        <br>
        <input type="checkbox" id="symptom8" name="symptom8"
value="Головні болі">
        <label for="symptom8">Головні болі</label>
        <br>
        <input type="checkbox" id="symptom9" name="symptom9"
value="Незрозуміла втома або м'язові болі">
        <label for="symptom9">Незрозуміла втома або м'язові
болі</label>
        <br>
        <input type="checkbox" id="symptom10" name="symptom10"
value="Біль у животі">
        <label for="symptom10">Біль у животі</label>
        <br>
        <input type="checkbox" id="symptom11" name="symptom11"
value="">
        <label for="symptom11">Нудота або блювання</label>
        <br>
        <input type="checkbox" id="symptom12" name="symptom12"
value="">
        <label for="symptom12">Почервоніння
очей(кон'юктивіт)</label>
        <br>
        <input type="checkbox" id="symptom13" name="symptom13"
value="">
        <label for="symptom13">Нежить</label>
        <br>
        <input type="checkbox" id="symptom14" name="symptom14"
value="">
        <label for="symptom14">Закладеність носа без інших
відомих причин</label>
        <br>
    </div>
<br><br>

```

```

        <label for="comment">Вкажіть ваш запит чи інші
СИМПТОМИ</label>
        <br>
        <textarea name="comment" rows="5" cols="50"
size="20"></textarea>

        <div class="row">
            <div class="col-25">
                <label>Стать*:</label>
            </div>
            <div class="col-75">
                <input required="required" type="radio" name="gender"
value="female">Жінка
                <input type="radio" name="gender" value="male">Чоловік
            </div>

        </DIV>
        <br><br>
        <input type="submit" name="submit" value=" Надіслати
запит">
        <div class="row">* Обов'язкові поля для заповнення</div>
    </div>
</form>

```

```

<?php
if (isset($_POST['submit'])) {
    if (isset($_POST['submit'])) {
        $message = $lname.' '.$fname.' '.$mname.' '.$phone.'
'.$email.' '.$gender.' '.$comment;
        $message.= $_POST['symptom1'].' '.$_POST['symptom2'].'
'.$_POST['symptom3'].' '.$_POST['symptom4'].'
                '.$_POST['symptom5'].'
'.$_POST['symptom6'].' '.$_POST['symptom7'].'
'.$_POST['symptom8'].'
                '.$_POST['symptom9'].'
'.$_POST['symptom10'].' '.$_POST['symptom11'].'
'.$_POST['symptom12'].'
                '.$_POST['symptom13'].'
'.$_POST['symptom14'];
        $to = 'admin@diplomtest.tk';
        $subject = 'Screening form'.date('YmdHm');
        $headers = 'X-Mailer: PHP/' . phpversion();

        mail($to, $subject, $message, $headers);
    };
}
<div>
    <footer>
        <div>
            САМОЛІКУВАННЯ ШКІДЛИВЕ ДЛЯ ВАШОГО ЗДОРОВ'Я
        </div>
    </footer>

```

В.11 Файл main.php

```

<div>
    <h1>Вас вітає сайт на якому зібрана статистика по Запоріжжю
на захворювання COVID-19 </h1>

</div>
<?php
include_once("connect.php"); //Підключення до бази даних
//Шаблони запитів до БД для таблиці та графіків

$query_patients_week = "select income_date as date, count(id) as
no_of_rows
                        from patients where income_date >=current_date-7
                        group by income_date order by income_date;";
$query_patients_month = "select income_date as date, count(id)
as no_of_rows
                        from patients where income_date >=current_date-
30
                        group by income_date order by income_date;";
$query_patients_total = "select income_date as date, count(id)
as no_of_rows
                        from patients group by income_date order by
income_date;";
$query_patients_daily = "select count(id) as no_of_rows from
patients where income_date >= current_date;";
$query_patients_healthy_daily = "select count(id) as no_of_rows
                        from patients where outgoing_date=current_date
                        and outgoing_condition = 'Одужав(ла)';";
$query_patients_healthy_total = "select count(id) as no_of_rows
                        from patients where outgoing_condition =
'Одужав(ла)';";
$query_patients_condition = "select count(id) as no_of_rows,
income_health
                        from patients where outgoing_condition is null
and income_health != 'Одужав(ла)'
                        and income_health != 'Помер(ла)' and
income_date>=current_date-7 group by income_health;";
$query_patients_alv = "select bed_used from (select count(id) as
bed_used,
                        0 alv from patients where outgoing_date is null
union all select sum(bed_count),
                        0 from hos_equiped union all select count(alv)
as alv_used,
                        0 from patients where outgoing_date is null and
alv = 'Так' union all select sum(alv_count),0
                        from hos_equiped ) as no;";
//Передача запитів до БД
$chart_patients_week = pg_query($con, $query_patients_week) or
die("Не можу виконати запит: $$query_patients_week\n");
$chart_patients_month = pg_query($con, $query_patients_month) or
die("Не можу виконати запит: $$query_patients_month\n");

```

```

$chart_patients_total = pg_query($con, $query_patients_total) or
die("Не можу виконати запит: $$query_patients_total\n");
$chart_patients_daily = pg_query($con, $query_patients_daily) or
die("Не можу виконати запит: $$query_patients_daily\n");
$chart_patients_daily = pg_query($con,
$query_patients_healthy_daily) or die("Не можу виконати запит:
$$query_patients_healthy_daily\n");
$chart_patients_total = pg_query($con,
$query_patients_healthy_total) or die("Не можу виконати запит:
$$query_patients_healthy_total\n");
$chart_patients_condition = pg_query($con,
$query_patients_condition) or die("Не можу виконати запит:
$$query_patients_condition\n");
$chart_patients_alv = pg_query($con, $query_patients_alv) or
die("Не можу виконати запит: $$query_patients_alv\n");

```

```

$patients_week_date = $patients_month_date =
$patients_total_date = $patients_cond_name = [];
$patients_week_val = $patients_month_val = $patients_total_val =
$result_patients_daily = $result_patients_daily =
$result_patients_total = $patients_cond_val = $patients_alv_data
= [];

```

```

while ($row = pg_fetch_assoc($chart_patients_week, null)) {
    $patients_week_date[] = $row['date'];
    $patients_week_val[] = $row['no_of_rows'];
};
while ($row = pg_fetch_assoc($chart_patients_month, null)) {
    $patients_month_date[] = $row['date'];
    $patients_month_val[] = $row['no_of_rows'];
};
while ($row = pg_fetch_assoc($chart_patients_total, null)) {
    $patients_total_date[] = $row['date'];
    $patients_total_val[] = $row['no_of_rows'];
};
while ($row = pg_fetch_assoc($chart_patients_daily, null)) {
    $result_patients_daily[] = $row['no_of_rows'];
};
while ($row = pg_fetch_assoc($chart_patients_daily, null)) {
    $result_patients_daily[] = $row['no_of_rows'];
};
while ($row = pg_fetch_assoc($chart_patients_total, null)) {
    $result_patients_total[] = $row['no_of_rows'];
};
while ($row = pg_fetch_assoc($chart_patients_condition, null)) {
    $patients_cond_name[] = $row['income_health'];
    $patients_cond_val[] = $row['no_of_rows'];
};
while ($row = pg_fetch_assoc($chart_patients_alv, null)) {
    $patients_alv_data[] = $row['bed_used'];
}
?>

```

```

<!-- Розташовуємо графіки -->
<div class="graphics">
    <!--Графік за тиждень -->
    <div class="chart-container">
        <canvas id="patients_week"></canvas>
    </div>
    <!--Графік за місяць -->
    <div class="chart-container">
        <canvas id="patients_month"></canvas>
    </div>
    <!--Графік за весь час -->
    <div class="chart-container">
        <canvas id="patients_total"></canvas>
    </div>
</div>
<div class="graphics">
    <!-- Стан пацієнтів за тиждень тяжкість-->
    <div class="chart-container">
        <canvas id="patients_condition"></canvas>
    </div>
    <!-- Кількість зайнятих ліжок та ІВЛ-->
    <div class="chart-container">
        <canvas id="patients_alv"></canvas>
    </div>
    <div class="chart-container">
        <canvas id="patients_alv_bed"></canvas>
    </div>
    <!-- Кількість потраплень за день-->
</div>
<div class="graphics">
    <div class="counter-div">
        <p id="counter" class="count-p"><?php echo
$result_patients_daily; ?></p>
        <p>Захворіло сьогодні</p>
    </div>
    <!-- Кількість виписаних здорових за день -->
    <div class="counter-div">
        <!-- функція print text    query_patients_healthy_daily--
>
        <p id="counter2" class="count-p"><?php echo
$result_patients_daily[0]; ?></p>
        <p style="text-align:center">Одужало сьогодні</p>
    </div>
    <!-- Кількість виписаних здорових всього -->
    <div class="counter-div">
        <!-- функція print text    query_patients_healthy_total--
>
        <p id="counter3" class="count-p"><?php echo
$result_patients_total[0] ?></p>
        <p>Одужали всього</p>
    </div>
</div>
<!-- Малюємо всі графіки та діаграми-->

```

```

<script>
    //patients_week
    var labels = <?php echo json_encode($patients_week_date);
?>; //переопределение переменной с пхп в переменную ДжС
    var data = <?php echo json_encode($patients_week_val); ?>;
    var ctx =
document.getElementById("patients_week").getContext("2d");
    var patients_week = new Chart(ctx, {
        type: 'bar',
        data: {
            labels: labels,
            datasets: [{
                label: "Поденна кількість захворівших за
тиждень",
                fill: true,
                backgroundColor: "rgba(75,192,192,0.4)",
                borderColor: "rgba(75,192,192,1)",
                data: data,
            }]
        },
        options: {
            tooltips: {
                intersect: false
            },
            scales: {
                yAxes: [{
                    ticks: {
                        beginAtZero: true,
                        precision: 0,
                    }
                }]
            }
        }
    });
    //patients_month
    var labels = <?php echo json_encode($patients_month_date);
?>; //переопределение переменной с пхп в переменную ДжС
    var data = <?php echo json_encode($patients_month_val); ?>;
    var ctx =
document.getElementById("patients_month").getContext("2d");
    var patients_month = new Chart(ctx, {
        type: 'line',
        data: {
            labels: labels,
            datasets: [{
                label: "Поденна кількість захворівших за
місяць",
                fill: true,
                backgroundColor: "rgba(75,192,192,0.4)",
                borderColor: "rgba(75,192,192,1)",
                data: data,
            }]
        },
    },

```

```

        options: {
            tooltips: {
                intersect: false
            },
            scales: {
                yAxes: [{
                    ticks: {
                        beginAtZero: true,
                        precision: 0,
                    }
                }]
            }
        }
    });
    //patients_total
    var labels = <?php echo json_encode($patients_total_date);
?>; //переопределение переменной с пхп в переменную ДжС
    var data = <?php echo json_encode($patients_total_val); ?>;
    var ctx =
document.getElementById("patients_total").getContext("2d");
    var patients_total = new Chart(ctx, {
        type: 'line',
        data: {
            labels: labels,
            datasets: [{
                label: "Загальна кількість захворівших",
                fill: true,
                backgroundColor: "rgba(75,192,192,0.4)",
                borderColor: "rgba(75,192,192,1)",
                data: data,
            }]
        },
        options: {
            tooltips: {
                intersect: false
            },
            scales: {
                yAxes: [{
                    ticks: {
                        beginAtZero: true,
                        precision: 0,
                    }
                }]
            }
        }
    });
    //patients_condition
    var labels = <?php echo json_encode($patients_cond_name);
?>; //переопределение переменной с пхп в переменную ДжС
    var data = <?php echo json_encode($patients_cond_val); ?>;
    var ctx =
document.getElementById("patients_condition").getContext("2d");
    var patients_cond = new Chart(ctx, {

```

```

        type: 'pie',
        data: {
            labels: labels,
            datasets: [{
                label: "Стан здоров'я за тиждень",
                fill: true,
                data: data,
                backgroundColor: ["rgba(0,255,0,1)",
                                "rgba(255,255,0,1)",
                                "rgba(255,0,0,1)",
                                ],
            }],
        },
        options: {
            plugins: {
                title: {
                    display: true,
                    text: 'Стан захворівших за тиждень'
                }
            }
        }
    });
    //patients_alv
    //var data= echo json_encode($patients_alv_data);
    var alv = <?php echo json_encode($patients_alv_data); ?>;
    var ctx =
document.getElementById("patients_alv").getContext("2d");
    var patients_alv = new Chart(ctx, {
        type: 'pie',
        data: {
            labels: ["Зайняті ліжка", "Вільні ліжка"],
            datasets: [{
                label: "Місця під ковід",
                fill: true,
                data: [alv[0], (alv[1] - alv[0])],
                backgroundColor: ["rgba(255,0,0,1)",
                                "rgba(0,255,0,1)",
                                ],
            }],
        },
        options: {
            plugins: {
                title: {
                    display: true,
                    text: 'ALV'
                }
            }
        }
    });

    //patients_alv_bed

```

```

var ctx =
document.getElementById("patients_alv_bed").getContext("2d");
var patients_alv_bed = new Chart(ctx, {
  type: 'pie',
  data: {
    labels: ["Зайняті ІВЛ", "Вільні ІВЛ"],
    datasets: [{
      label: "Апарати ІВЛ",
      fill: true,
      data: [alv[2], (alv[3] - alv[2])],
      backgroundColor: ["rgba(255,0,0,1)",
        "rgba(0,255,0,1)",
      ],
    }]
  },
  options: {
    plugins: {
      title: {
        display: true,
        text: 'Ліжка'
      }
    }
  }
});

par = document.getElementById('counter');
count = par.textContent;
i = 0;

function counter() {
  if (i < count) {
    i++;
  }
  par.textContent = i;
}
setInterval(counter, 1);

par2 = document.getElementById('counter2');
count2 = par2.textContent;
b = 0;

function counter2() {
  if (b < count2) {
    b++;
  }
  par2.textContent = b;
}
setInterval(counter2, 1);

par3 = document.getElementById('counter3');
count3 = par3.textContent;
c = 0;

```

```

function counter3() {
    if (c < count3) {
        c++;
    }
    par3.textContent = c;
}
setInterval(counter3, 1);
</script>

```

B.12 Файл patients.php

```

<?php
include_once("connect.php"); //Підключення до бази даних
//Шаюлони запитів до БД для таблиці та графіків
$query_patients = "select * from patients where
location='$location' order by income_date desc;"; //Запит на
отримання повної таблиці для лайв режиму
$query_patients_week = "select income_date as date, count(id)
as no_of_rows
        from patients where location ='$location' and
income_date >=current_date-7
        group by income_date order by income_date;";
$query_patients_month = "select income_date as date, count(id)
as no_of_rows
        from patients where location ='$location' and
income_date >=current_date-30
        group by income_date order by income_date;";
$query_patients_total = "select income_date as date, count(id)
as no_of_rows
        from patients where location='$location' group
by income_date order by income_date;";
$query_patients_daily = "select count(id) as no_of_rows from
patients where location ='$location' and income_date >=
current_date;";
$query_patients_healthy_daily = "select count(id) as no_of_rows
        from patients where location ='$location' and
outgoing_date=current_date
        and outgoing_condition = 'Одужав(ла)';";
$query_patients_healthy_total = "select count(id) as no_of_rows
        from patients where location ='$location' and
outgoing_condition = 'Одужав(ла)';";
$query_patients_condition = "select count(id) as no_of_rows,
income_health
        from patients where location ='$location' and
outgoing_condition is null and income_health != 'Одужав(ла)'
        and income_health != 'Помер(ла)' and
income_date>=current_date-7 group by income_health;";
$query_patients_alv = "select bed_used from (select count(id)
as bed_used,
        0 alv from patients where location ='$location'
and outgoing_date is null union all select bed_count,
        0 from hos_equiped where hos_name = '$location'
union all select count(alv) as alv_used,

```

```

        0 from patients where location = '$location' and
        outgoing_date is null and alv = 'Так' union all select
        alv_count, 0
        from hos_equiped where hos_name = '$location')
as no;";

```

```

//Передача запитів до БД
$stable_hospital = pg_query($con, $query_patients) or die("Не
можу виконати запит: $$query_patients\n");
$chart_patients_week = pg_query($con, $query_patients_week) or
die("Не можу виконати запит: $$query_patients_week\n");
$chart_patients_month = pg_query($con, $query_patients_month) or
die("Не можу виконати запит: $$query_patients_month\n");
$chart_patients_total = pg_query($con, $query_patients_total) or
die("Не можу виконати запит: $$query_patients_total\n");
$chart_patients_daily = pg_query($con, $query_patients_daily) or
die("Не можу виконати запит: $$query_patients_daily\n");
$chart_patients_h_daily = pg_query($con,
$query_patients_healthy_daily) or die("Не можу виконати запит:
$$query_patients_healthy_daily\n");
$chart_patients_h_total = pg_query($con,
$query_patients_healthy_total) or die("Не можу виконати запит:
$$query_patients_healthy_total\n");
$chart_patients_h_condition = pg_query($con,
$query_patients_condition) or die("Не можу виконати запит:
$$query_patients_condition\n");
$chart_patients_alv = pg_query($con, $query_patients_alv) or
die("Не можу виконати запит: $$query_patients_alv\n");

```

```

$patients_week_date = $patients_month_date =
$patients_total_date = $patients_cond_name = [];
$patients_week_val = $patients_month_val = $patients_total_val =
$result_patients_daily = $result_patients_h_daily =
$result_patients_h_total = $patients_cond_val =
$patients_alv_data = [];

```

```

while ($row = pg_fetch_assoc($chart_patients_week, null)) {
    $patients_week_date[] = $row['date'];
    $patients_week_val[] = $row['no_of_rows'];
};
while ($row = pg_fetch_assoc($chart_patients_month, null)) {
    $patients_month_date[] = $row['date'];
    $patients_month_val[] = $row['no_of_rows'];
};
while ($row = pg_fetch_assoc($chart_patients_total, null)) {
    $patients_total_date[] = $row['date'];
    $patients_total_val[] = $row['no_of_rows'];
};
while ($row = pg_fetch_assoc($chart_patients_daily, null)) {
    $result_patients_daily[] = $row['no_of_rows'];
};
while ($row = pg_fetch_assoc($chart_patients_h_daily, null)) {

```

```

        $result_patients_h_daily[] = $row['no_of_rows'];
    };
    while ($row = pg_fetch_assoc($chart_patients_h_total, null)) {
        $result_patients_h_total[] = $row['no_of_rows'];
    };
    while ($row = pg_fetch_assoc($chart_patients_h_condition, null)) {
        $patients_cond_name[] = $row['income_health'];
        $patients_cond_val[] = $row['no_of_rows'];
    };
    while ($row = pg_fetch_assoc($chart_patients_alv, null)) {
        $patients_alv_data[] = $row['bed_used'];
    }
    /*echo $query_patients_daily;
    print_r($chart_patients_daily);
    echo $location;
    print_r($result_patients_daily); */?>
<!-- Розташовуємо графіки -->
<div class="graphics">
    <!--Графік за тиждень -->
    <div class="chart-container">
        <canvas id="patients_week"></canvas>
    </div>
    <!--Графік за місяць -->
    <div class="chart-container">
        <canvas id="patients_month"></canvas>
    </div>
    <!--Графік за весь час -->
    <div class="chart-container">
        <canvas id="patients_total"></canvas>
    </div>
</div>
<div class="graphics">
    <!-- Стан пацієнтів за тиждень тяжкість-->
    <div class="chart-container">
        <canvas id="patients_condition"></canvas>
    </div>
    <!-- Кількість зайнятих ліжок та ІВЛ-->
    <div class="chart-container">
        <canvas id="patients_alv"></canvas>
    </div>
    <div class="chart-container">
        <canvas id="patients_alv_bed"></canvas>
    </div>
    <!-- Кількість потраплень за день-->
</div>
<div class="graphics">
    <div class="counter-div">

        <p id="counter" class="count-p"><?php echo
$result_patients_daily[0]; ?></p>
        <p>Захворіло сьогодні</p>
    </div>

```

```

        <!-- Кількість виписаних здорових за день -->
        <div class="counter-div">
            <!-- функція print text    query_patients_healthy_daily--
>
            <p id="counter2" class="count-p"><?php echo
$result_patients_h_daily[0]; ?></p>
            <p style="text-align:center">Одужало сьогодні</p>
        </div>
        <!-- Кількість виписаних здорових всього -->
        <div class="counter-div">
            <!-- функція print text    query_patients_healthy_total--
>
            <p id="counter3" class="count-p"><?php echo
$result_patients_h_total[0] ?></p>
            <p>Одужали всього</p>
        </div>
    </div>

    <?php
    if (isset($_SESSION['use']) and $location == $_SESSION['loc']) {
    ?>

        <a href="#" class="button graphics centered"
onclick="show_form(); return false"><span id="sp_sel">Відкрити
форму для додавання</span></a>
        <div id="div_form" style="display:none">

            <div class="page-header">
                <h2 style="text-align:center;"> Форма для додавання
нового пацієнта</h2>
            </div>
            <p style="text-align:center">Будь-ласка заповніть форму
для додавання запису до таблиці нижче</p>

            <div style="display:flex;justify-content:center;">
                <table class="table table-bordered table-striped">
                    <form action="" method="post">
                        <thead>
                            <tr>
                                <th>Дата надходження</th>
                                <th>Стан здоров'я</th>
                                <th>ПІБ</th>
                                <th>Вік пацієнта</th>
                                <th>Адреса проживання чи
реєстрації</th>
                                <th>Номер телефону</th>
                            </tr>
                        </thead>
                        <tbody>
                            <tr>
                                <td><input required type="text"
name="Fdate-income" oninvalid="this.setCustomValidity('Введіть

```

```

дату')" oninput="this.setCustomValidity('')"' pattern="([0-
9]){4}\-[0-1][0-9]\-[0-3][0-9]" placeholder=2020-04-08></td>
<td><select name="Fhealth_income">
    <option default value="Легка
форма">Легка форма</option>
    <option value="Середня
форма">Середня форма</option>
    <option value="Важка
форма">Важка форма</option>
</select></td>
<td><input required type="text"
name="Fpii"></td>
<td><input type="number" name="Fage"
min="0" max="120" step="1" value="0"></td>
<td><input type="text"
name="Faddress"></td>
<td><input type="tel" name="Fphone"
oninvalid="this.setCustomValidity('Введіть телефон')"'
oninput="this.setCustomValidity('')"' pattern="\+380([0-9]){9}"
placeholder="+380123456789"></td>
</tr>
</tbody>
<thead>
<th>Кількість контактних осіб</th>
<th>Кількість захворілих
контактних</th>
<th>Підключений до ІВЛ</th>
<th>Дата виписки</th>
<th>Стан здоров'я на момент виписки</th>
<th></th>
</thead>
<tbody>
<tr>
<td><input type="number"
name="Fcontacts" min="0" max="100" step="1" value="0"></td>
<td><input type="number"
name="Fcontacts_sick" min="0" max="100" step="1" value="0"></td>
<td><select name="Falv">
    <option
value='Так'>Так</option>
    <option value='Hi'
selected>Hi</option>
</select></td>
<td><input type="text"
name="Fdate_outgoing" oninvalid="this.setCustomValidity('Введіть
дату')"' oninput="this.setCustomValidity('')"' pattern="([0-
9]){4}\-[0-1][0-9]\-[0-3][0-9]" placeholder=2020-04-08></td>
<td><select name="Fhealth_outgoing">
    <option value='NULL'>
</option>
    <option value='Легка
форма'>Легка форма</option>

```

```

        форма'>Середня форма</option>
        форма'>Важка форма</option>
        value='Одужав (ла) '>Одужав (ла)</option>
        value='Помер (ла) '>Помер (ла)</option>
    </select></td>
    <td class="btn btn-primary btnmy"><input
type="submit" class="btn btn-primary btnmy" name="submit"
value="Додати запис"></td>
</tr>
</tbody>
</form>
</table>
</div>
</div>

```

```

<?php
}
if (isset($_POST['submit'])) {
    $Fincome_date = $Fincome_health = $Fpii = $Fage =
$Faddress = $Fphone = "NULL";
    //echo $Fphone;
    $Fcontacts = $Fcontacts_sick = $Fage = 0;
    $Falv = $Foutgoing_date = $Foutgoing_condition = "NULL";
    if (!$_POST['Fdate-income']) {
        $Fincome_date = "NULL";
    } else {
        $Fincome_date = $_POST['Fdate-income'];
    };
    if (!$_POST['Fhealth_income']) {
        $Fincome_health = "'Легка форма'";
    } else {
        $Fincome_health = $_POST['Fhealth_income'];
    };
    if (!$_POST['Fpii']) {
        $Fpii = NULL;
    } else {
        $Fpii = $_POST['Fpii'];
    }
    $Fage = $_POST['Fage'];
    $Faddress = $_POST['Faddress'];
    $Fphone = $_POST['Fphone'];
    $Fcontacts = $_POST['Fcontacts'];
    $Fcontacts_sick = $_POST['Fcontacts_sick'];
    if (!$_POST['Falv']) {
        $Falv = "Hi";
    } else {
        $Falv = $_POST['Falv'];
    };
    if (!$_POST['Fdate_outgoing']) {
        $Foutgoing_date = 'NULL';
    }
}

```

```

    } else {
        $Foutgoing_date = $_POST['Fdate_outgoing'];
    };
    if (!$_POST['Fhealth_outgoing']) {
        $Foutgoing_condition = 'NULL';
    } else {
        $Foutgoing_condition = $_POST['Fhealth_outgoing'];
    };
    //echo $Foutgoing_condition;
    if ($Foutgoing_date == 'NULL' and $Foutgoing_condition ==
'NULL') { //not working bitch
        $sql = "INSERT INTO patients VALUES
(nextval('patients_id_seq'::regclass), '$Fincome_date', '$Fincome_
health', '$Fpii',

'$Fage', '$Faddress', '$Fphone', '$Fcontacts', '$Fcontacts_sick', '$F
alv', NULL, NULL, '$location', '$doctor');"
    } else {
        $sql = "INSERT INTO patients VALUES
(nextval('patients_id_seq'::regclass), '$Fincome_date', '$Fincome_
health', '$Fpii',

'$Fage', '$Faddress', '$Fphone', '$Fcontacts', '$Fcontacts_sick', '$F
alv', '$Foutgoing_date', '$Foutgoing_condition', '$location', '$doct
or');"
    };

    //echo($sql);
    $sql_query_add = pg_query($con, $sql);
    if ($sql_query_add) {
        echo "Новий запис додано!";
    } else {
        echo "Помилка у запиті: " . $sql . " :-";
    }
    pg_close($con);
}
?>
<!-- Малюємо всі графіки та діаграми-->
<script>
    //patients_week
    var labels = <?php echo json_encode($patients_week_date);
?>; //переопределение переменной с пхп в переменную ДжС
    var data = <?php echo json_encode($patients_week_val); ?>;
    var ctx =
document.getElementById("patients_week").getContext("2d");
    var patients_week = new Chart(ctx, {
        type: 'bar',
        data: {
            labels: labels,
            datasets: [{
                label: "Поденна кількість захворівших за
тиждень",
                fill: true,

```

```

        backgroundColor: "rgba(75,192,192,0.4)",
        borderColor: "rgba(75,192,192,1)",
        data: data,
    }]
},
options: {
    tooltips: {
        intersect: false
    },
    scales: {
        yAxes: [{
            ticks: {
                beginAtZero: true,
                precision: 0,
            }
        }]
    }
}
});
//patients_month
var labels = <?php echo json_encode($patients_month_date);
?>; //переопределение переменной с пхп в переменную ДжС
var data = <?php echo json_encode($patients_month_val); ?>;
var ctx =
document.getElementById("patients_month").getContext("2d");
var patients_month = new Chart(ctx, {
    type: 'line',
    data: {
        labels: labels,
        datasets: [{
            label: "Поденна кількість захворівших за
місяць",
            fill: true,
            backgroundColor: "rgba(75,192,192,0.4)",
            borderColor: "rgba(75,192,192,1)",
            data: data,
        }]
    },
    options: {
        tooltips: {
            intersect: false
        },
        scales: {
            yAxes: [{
                ticks: {
                    beginAtZero: true,
                    precision: 0,
                }
            }]
        }
    }
});
//patients_total

```

```

    var labels = <?php echo json_encode($patients_total_date);
?>; //переопределение переменной с пхп в переменную ДжС
    var data = <?php echo json_encode($patients_total_val); ?>;
    var ctx =
document.getElementById("patients_total").getContext("2d");
    var patients_total = new Chart(ctx, {
        type: 'line',
        data: {
            labels: labels,
            datasets: [{
                label: "Загальна кількість захворівших",
                fill: true,
                backgroundColor: "rgba(75,192,192,0.4)",
                borderColor: "rgba(75,192,192,1)",
                data: data,
            }]
        },
        options: {
            tooltips: {
                intersect: false
            },
            scales: {
                yAxes: [{
                    ticks: {
                        beginAtZero: true,
                        precision: 0,
                    }
                }]
            }
        }
    });
//patients_condition
    var labels = <?php echo json_encode($patients_cond_name);
?>; //переопределение переменной с пхп в переменную ДжС
    var data = <?php echo json_encode($patients_cond_val); ?>;
    var ctx =
document.getElementById("patients_condition").getContext("2d");
    var patients_cond = new Chart(ctx, {
        type: 'pie',
        data: {
            labels: labels,
            datasets: [{
                label: "Стан здоров'я за тиждень",
                fill: true,
                data: data,
                backgroundColor: ["rgba(0,255,0,1)",
                    "rgba(255,255,0,1)",
                    "rgba(255,0,0,1)",
                ],
            }]
        },
        options: {
            plugins: {

```

```

        title: {
            display: true,
            text: 'Стан захворівших за тиждень'
        }
    }
});
//patients_alv
//var data= echo json_encode($patients_alv_data);
var alv = <?php echo json_encode($patients_alv_data); ?>;
var ctx =
document.getElementById("patients_alv").getContext("2d");
var patients_alv = new Chart(ctx, {
    type: 'pie',
    data: {
        labels: ["Зайняті ліжка", "Вільні ліжка"],
        datasets: [{
            label: "Місця під ковід",
            fill: true,
            data: [alv[0], (alv[1] - alv[0])],
            backgroundColor: ["rgba(255,0,0,1)",
                "rgba(0,255,0,1)",
            ],
        }]
    },
    options: {
        plugins: {
            title: {
                display: true,
                text: 'ALV'
            }
        }
    }
});

//patients_alv_bed
var ctx =
document.getElementById("patients_alv_bed").getContext("2d");
var patients_alv_bed = new Chart(ctx, {
    type: 'pie',
    data: {
        labels: ["Зайняті ІВЛ", "Вільні ІВЛ"],
        datasets: [{
            label: "Апарати ІВЛ",
            fill: true,
            data: [alv[2], (alv[3] - alv[2])],
            backgroundColor: ["rgba(255,0,0,1)",
                "rgba(0,255,0,1)",
            ],
        }]
    },
    options: {

```

```

        plugins: {
            title: {
                display: true,
                text: 'Ліжка'
            }
        }
    });

    par = document.getElementById('counter');
    count = par.textContent;
    i = 0;

    function counter() {
        if (i < count) {
            i++;
        }
        par.textContent = i;
    }
    setInterval(counter, 1);

    par2 = document.getElementById('counter2');
    count2 = par2.textContent;
    b = 0;

    function counter2() {
        if (b < count2) {
            b++;
        }
        par2.textContent = b;
    }
    setInterval(counter2, 1);

    par3 = document.getElementById('counter3');
    count3 = par3.textContent;
    c = 0;

    function counter3() {
        if (c < count3) {
            c++;
        }
        par3.textContent = c;
    }
    setInterval(counter3, 1);

    function show_form() {
        var d = document;
        var a = d.getElementById('div_form').style.display;
        d.getElementById('div_form').style.display = (a ==
'none') ? 'block' : 'none';
        d.getElementById('sp_sel').innerHTML = (a == 'none') ?
'Згорнути форму для додавання' : 'Розгорнути форму для
додавання';
    }

```

```

    }
</script>

```

```

<!--Відображаємо таблицю з даними -->
<div class="graphics">
    <table id="sickList">
        <thead>
            <tr>
                <th>Дата надходження</th>
                <th>Стан здоров'я</th>
                <th>ПІБ</th>
                <th>Вік</th>
                <th>Адреса проживання</th>
                <th>Телефон</th>
                <th>Кількість контактних</th>
                <th>Захворіло контактних</th>
                <th>ІВЛ</th>
                <th>Дата виписки</th>
                <th>Стан на дату виписки</th>
                <?php if (isset($_SESSION['use'])) {
                    echo '<th></th>';
                    echo '<th></th>';
                } ?>
            </tr>
        </thead>
        <tbody>
            <?php while ($row = pg_fetch_row($table_hospital)) {
?>
                <tr id="<?php echo $row[0]; ?>">
                    <td><?php echo $row[1]; ?></td>
                    <td><?php echo $row[2]; ?></td>
                    <td><?php echo $row[3]; ?></td>
                    <td><?php echo $row[4]; ?></td>
                    <td><?php echo $row[5]; ?></td>
                    <td><?php echo $row[6]; ?></td>
                    <td><?php echo $row[7]; ?></td>
                    <td><?php echo $row[8]; ?></td>
                    <td><?php echo $row[9]; ?></td>
                    <td><?php echo $row[10]; ?></td>
                    <td><?php echo $row[11]; ?></td>

                    <?php if (isset($_SESSION['use'])) {
                        //echo ' <td> ' . $row[12] . ' </td> <td>
' . $row[13] . ' </td>';
                        echo' <td> <input type="submit"
class="btnEd" value="Редагувати">
                        <input type="submit" class="delBtn"
value="Видалити"</td>';
                    } ?>
                </tr>
            <?php } ?>

```

```

        </tbody>
    </table>
</div>

```

B.13 Файл main.css

```

p {
    color: black;
    font-weight: normal;
    letter-spacing: normal;
    word-spacing: normal;
    font-family: 'Times New Roman', Times, serif;
}

nav {
    color: white;
    background-color: darkblue;
    font-size: larger;
    font-weight: bold;
    font-family: Arial, Helvetica, sans-serif;
}

h1 {
    font-family: Arial, Helvetica, sans-serif;
    text-align: center;
}

footer {
    position: relative;
    color: black;
    text-align: center;
    opacity: 0.5;
    font-size: smaller;
    bottom: 0;
    right: auto;
    left: auto;
    width: 98%;
}

/* Navbar container */

.navbar {
    overflow: hidden;
    background-color: #333;
    font-family: Arial;
}

/* Links inside the navbar */

.navbar a {
    float: left;
    font-size: 16px;
    color: white;

```

```

    text-align: center;
    padding: 14px 14px;
    text-decoration: none;
}

/* The dropdown container */

.dropdown {
    float: left;
    overflow: hidden;
}

/* Dropdown button */

.dropdown .dropbtn {
    font-size: 16px;
    border: none;
    outline: none;
    color: white;
    padding: 14px 16px;
    background-color: inherit;
    font-family: inherit;
    /* Important for vertical align on mobile phones */
    margin: 0;
    /* Important for vertical align on mobile phones */
}

/* Add a red background color to navbar links on hover */

.navbar a:hover, .dropdown:hover .dropbtn {
    background-color: red;
}

/* Dropdown content (hidden by default) */

.dropdown-content {
    display: none;
    position: absolute;
    background-color: #f9f9f9;
    min-width: 160px;
    box-shadow: 0px 8px 16px 0px rgba(0, 0, 0, 0.2);
    z-index: 1;
}

/* Links inside the dropdown */

.dropdown-content a {
    float: none;
    color: black;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
    text-align: left;

```

```

}

/* Add a grey background color to dropdown links on hover */

.dropdown-content a:hover {
    background-color: #ddd;
}

/* Show the dropdown menu on hover */

.dropdown:hover .dropdown-content {
    display: block;
}

/* Button used to open the contact form - fixed at the bottom of
the page */

.open-button {
    font-size: 16px;
    overflow: hidden;
    background-color: #333;
    font-family: Arial;
    color: white;
    padding: 16px 20px;
    border: none;
    cursor: pointer;
    position: relative;
    float: right;
    top: auto;
    max-width: 180px;
}

.enterbtn {
    float: right;
}

.btnEd ,.delBtn{
    padding:10px;
    width:25px;
    text-decoration:none;
    border-radius:4px;
    background-color:orange;
}

.delBtn{
    background-color: red;;
}

/* CHART*/

.graphics {
    display: flex;
    justify-content: center;
}

```

```
.centered {
  margin-left: 10%;
  margin-right: 10%;
}

.btnmy {
  color: #ffffff;
  background-color: orange;
  border: none;
  text-align: center;
  font-size: 16px;
}

.chart-container {
  height: 200;
  width: 400;
  display: inline-block;
  margin-right: 5px;
  margin-top: 5px;
}

.counter-div {
  height: 200;
  width: 400;
  display: inline-block;
  margin-right: 5px;
  margin-top: 5px;
}

.counter-div p {
  text-align: center;
  color: #535353;
}

.counter-div .count-p {
  color: red;
  font-size: 48px;
  font-weight: 500;
}

.container {
  display: block;
  justify-content: center;
}

.container p {
  text-align: center;
}

.help-container {
  position: relative;
  z-index: 1;
}
```

```

    background: #FFFFFF;
    max-width: 400px;
    margin: 0 auto 100px;
    padding: 45px;
    box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0,
0, 0, 0.24);
}

.col-25 {
    width: 25%;
    float: left;
}

.col-25 label {
    padding: 10px;
    width: 100%;
}

.col-75 {
    width: 75%;
    float: left;
}

input[type=text] {
    width: 100%;
    padding: 10px;
}

input[type=submit] {
    width: 100%;
    padding: 10px;}

.row:after {
    content: "";
    display: table;
    clear: both;
}
table {
    width: 90%;
    overflow-x: auto;
    border-collapse: collapse;
    margin: 25px 0;
    font-size: 0.9em;
    font-family: sans-serif;
    /*min-width: 400px;*/
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.15);
}
thead tr {
    background-color: #009879;
    /*color: #ffffff;*/
    text-align: left;
}
th, td, tr {

```

```

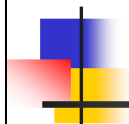
    padding: 5px 8px;
}
th {
    background-color: #009879;
}
tr {
    border-bottom: 1px solid #dddddd;
}
tr:nth-child(even) {
    background-color: #dee1ac7c;
}
tr:last-of-type {
    border-bottom: 2px solid #009879;
}
.button {
    font: bold larger;
    text-decoration: none;
    background-color: #009879;
    color: white;
    padding: 2px 6px 2px 6px;
    border: 1px solid #000;
}
.error {
    color: red;
}
.div_form {
    display: none;
}
.modal-body {
    background: white;
}

```

ДОДАТОК Г
Слайди презентації

Національний університет "Запорізька політехніка"
Кафедра програмних засобів

Дипломна робота бакалавра на тему:
**«Розробка оперативно-інформаційної
системи для аналізу захворювань
на особливо небезпечні хвороби»**



Розробила:
студентка гр. КНТ-217

А.В. Гончарова

Керівник:
к.т.н., доцент

О.М. Гладкова

Рисунок Г.1 – Слайд № 1



- Об'єкт проектування – вебсайт для перегляду користувачами та реєстрації лікарями випадків захворювання на особливо небезпечні хвороби.
- Предмет проектування – база даних щодо зберігання випадків захворювання та вебінтерфейс для доступу до неї.
- Мета роботи – створення оперативно-інформаційної системи для аналізу захворювань на COVID-19, яка дозволить підвищити ефективність протиепідемічних заходів щодо боротьби проти смертельної хвороби.

2

Рисунок Г.2 – Слайд № 2



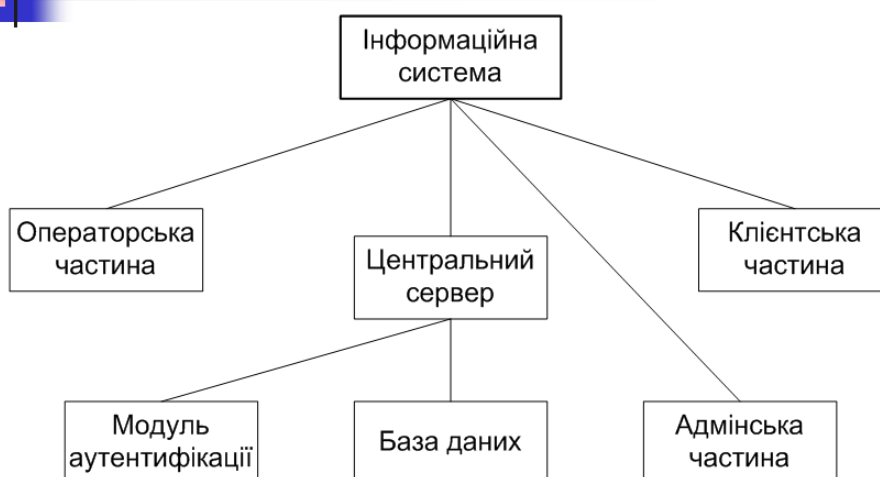
Задачі дипломної роботи

- Провести аналіз сучасних аналогічних розробок
- Розробити структурну та функціональну схеми оперативно-інформаційної системи
- Створити концептуальну та реляційну модель бази даних системи
- Виконати нормалізацію моделі БД та побудувати логічну схему
- Обрати середовище проектування СКБД
- Створити фізичну модель БД з таблицями
- Забезпечити захист конфіденційних даних
- Розробити інтерфейси інформаційної системи
- Створити екранні форми інтерфейсів

3

Рисунок Г.3 – Слайд № 3

Структурна схема



4

Рисунок Г.4 – Слайд № 4

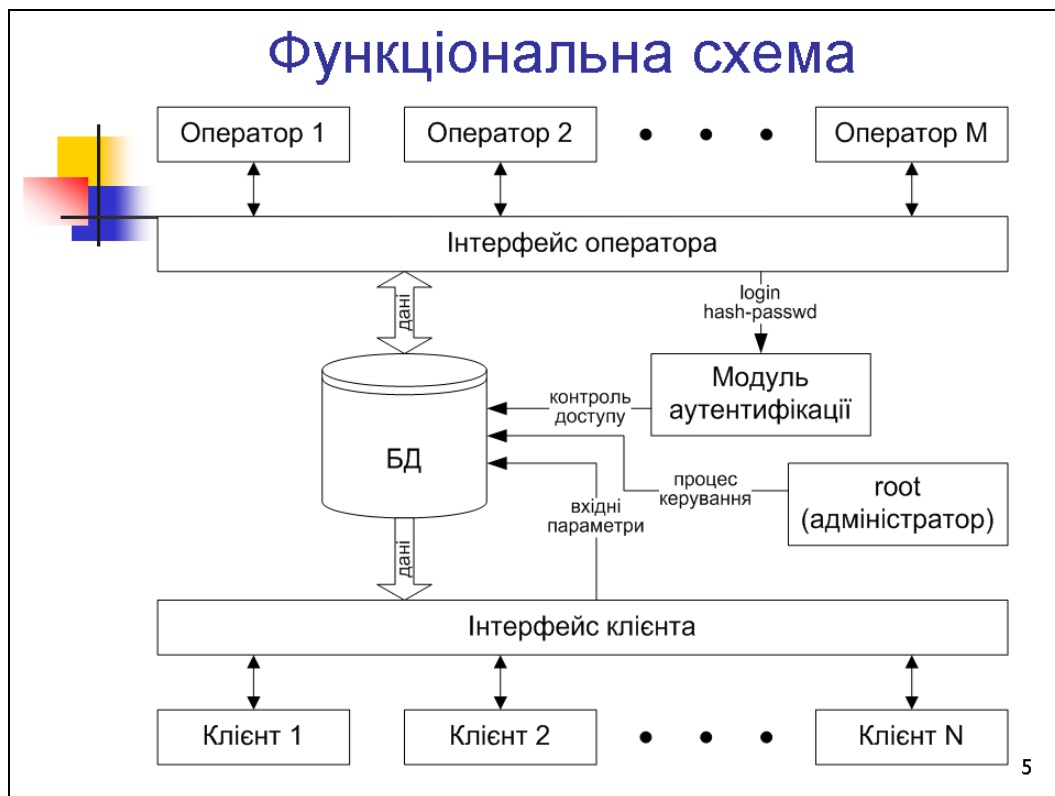


Рисунок Г.5 – Слайд № 5

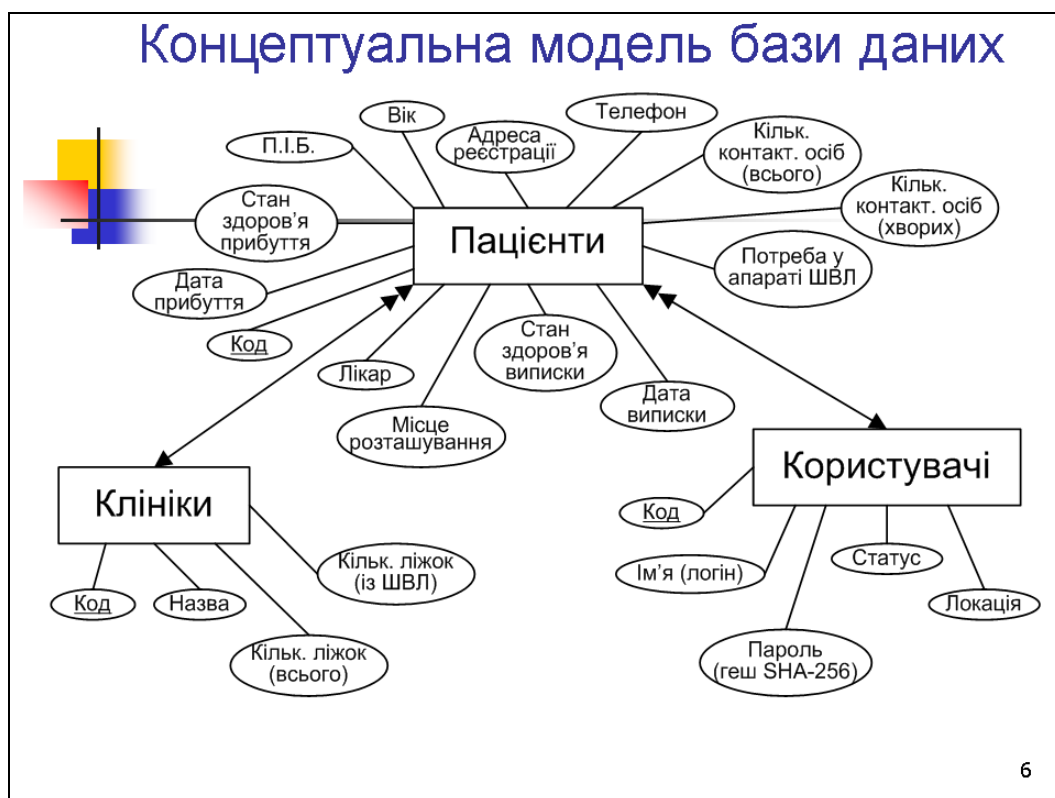
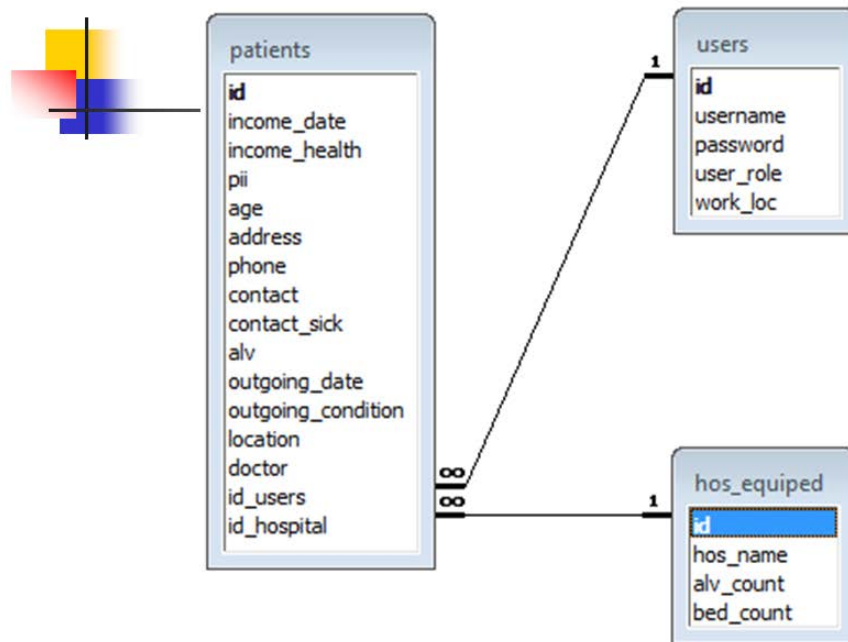


Рисунок Г.6 – Слайд № 6

Логічна схема реляційної бази даних



7

Рисунок Г.7 – Слайд № 7

Порівняльний аналіз параметрів СКБД

СКБД	Розробник	Тип	Операційна система	Ліцензія	Початковий код	Підтримка	Стабільність	Популярність
Oracle Database	Oracle Corporation	Мульти-модельна	Linux, Microsoft Windows, Oracle Solaris, IBM AIX, HP-UX	Комерційна	Закритий	За гроші	+	+
MySQL	Oracle Corporation	Реляційна	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	GNU GPL та комерційна	Відкритий	За гроші	+	+
DB2	IBM	Об'єктно-реляційна	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	Пропріетарна EULA	Закритий	Безкоштовна	+	-
SAP HANA	SAP SE	Реляційна, in-memory	Linux	Комерційна	Закритий	За гроші	+	-
SQL Server	Microsoft	Реляційна	Linux, Microsoft Windows	Комерційна	Закритий	Безкоштовна	+	+
MariaDB	MariaDB Corporation AB, MariaDB Foundation	Реляційна	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	GNU GPL	Відкритий	За гроші	-	-
MongoDB	MongoDB Inc.	Документно-орієнтована	Linux, Microsoft Windows, Oracle Solaris, macOS, FreeBSD	GNU AGPL (СКБД) та Apache License (драйвери)	Відкритий	За гроші	+	+
PostgreSQL	PostgreSQL Global Development Group	Об'єктно-реляційна	Linux, Microsoft Windows, Oracle Solaris, IBM AIX, HP-UX, macOS, QNX	Вільне та відкрите програмне забезпечення, дозвільна ліцензія	Відкритий	За гроші	+	+

8

Рисунок Г.8 – Слайд № 8

Структура схем у СКБД із активованим розширенням для маскування даних



Schema	Owner	Actions			Comment
☐ anon	www	Drop	Privileges	Alter	
☐ mask	www	Drop	Privileges	Alter	
☐ public	postgres	Drop	Privileges	Alter	standard public schema

9

Рисунок Г.9 – Слайд № 9

Перелік таблиць шаблонної схеми «anon» для анонімізації даних

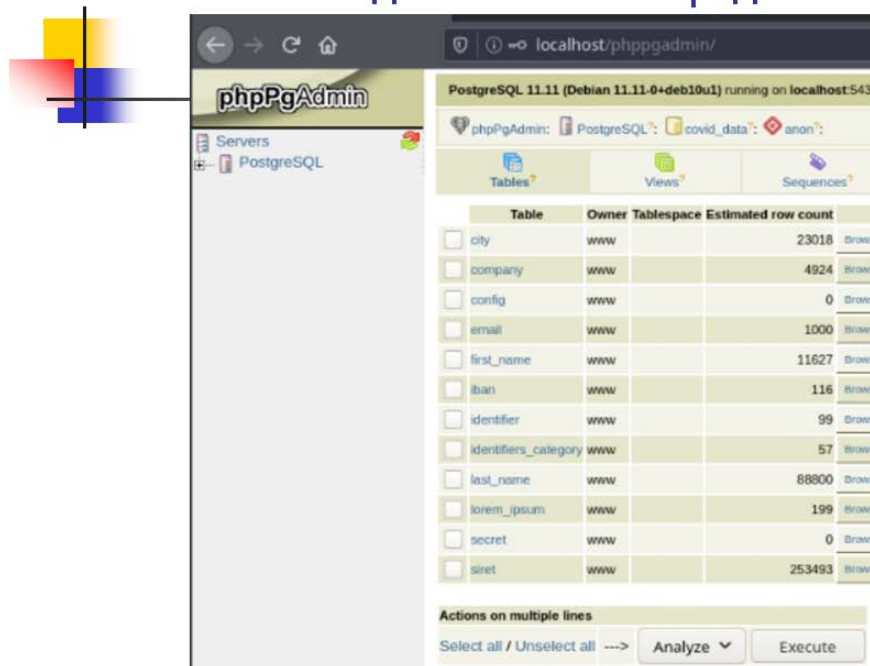


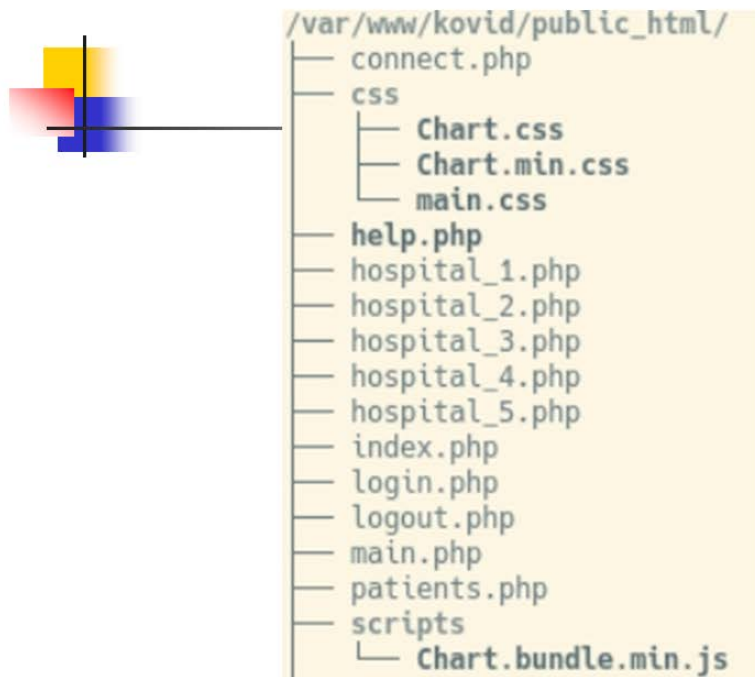
Table	Owner	Tablespace	Estimated row count
☐ city	www		23018
☐ company	www		4924
☐ config	www		0
☐ email	www		1000
☐ first_name	www		11627
☐ iban	www		116
☐ identifier	www		99
☐ identifiers_category	www		57
☐ last_name	www		88800
☐ lorem_ipsum	www		199
☐ secret	www		0
☐ sret	www		253493

Actions on multiple lines
 Select all / Unselect all ---> Analyze Execute

10

Рисунок Г.10 – Слайд № 10

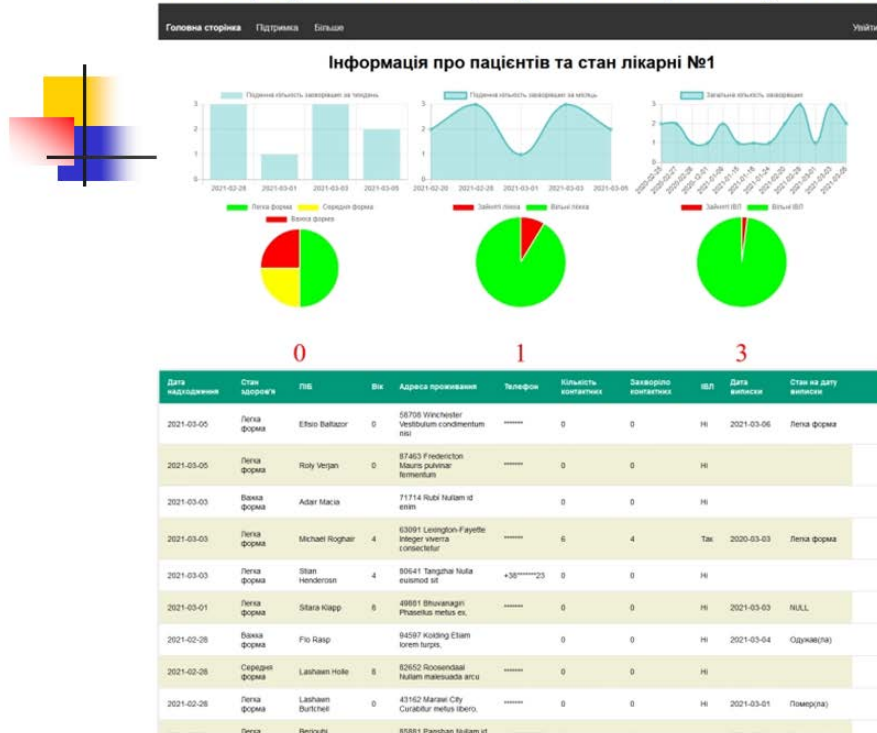
Ієрархічна структура вебсайту



11

Рисунок Г.11 – Слайд № 11

Інтерфейсна форма користувача



12

Рисунок Г.12 – Слайд № 12

Інтерфейсна форма оператора (лікаря)

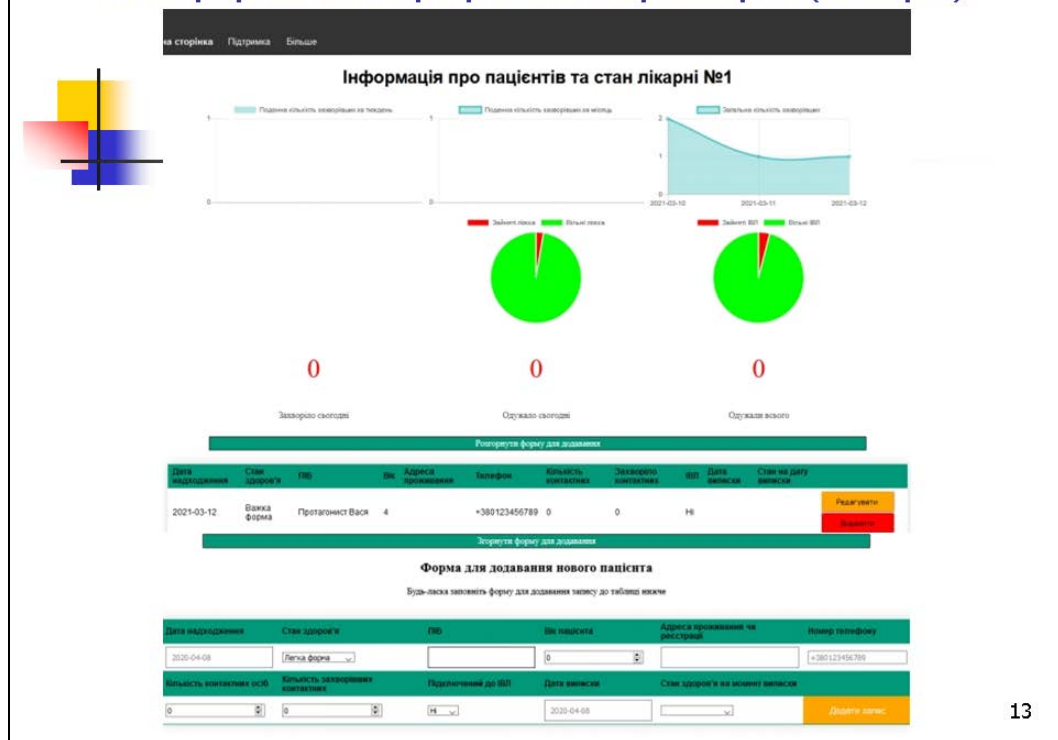


Рисунок Г.13 – Слайд № 13

Висновки

- Проведено аналіз сучасних аналогічних розробок
- Розроблено структурну та функціональну схеми оперативно-інформаційної системи
- Створено концептуальну та реляційну модель бази даних інформаційної системи
- Виконано нормалізацію моделі БД та побудовано логічну схему
- Обрано середовище проектування СКБД PostgreSQL версії 11.11
- Створено фізичну модель БД з таблицями у інструментальному середовищі phpPgAdmin
- Встановлено до СКБД та активовано розширення PostgreSQL-Anonymizer для маскуванню конфіденційних даних
- Розроблено вебінтерфейси інформаційної системи із застосуванням сучасних мов HTML 5, PHP, JS, технології CSS, бібліотеки ChartJS та методів JSON

Рисунок Г.14 – Слайд № 14