Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи і моделі оцінки стану клієнтів оздоровчих і відпочивальних закладів

(тема)

Виконав:

студент II курсу, групи КСМм-20-1

Рубан А.М.
(прізвище, ініціали)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерні системи та мережі
(повна назва освітньої програми)

Керівник: проф. Завизіступ Ю.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Рубану Андрію Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи і моделі оцінки стану клієнтів оздоровчих і відпочивальних закладів

затверджена наказом по університету від “ 5 ” листопада 2021 р. № 1656 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13 грудня 2021 р.

3. Вхідні дані до роботи 1) зображення енергограм пальців рук людини
2) отримання енергограм на основі поляризації діелектриків
3) хмарний веб сервіс

4. Перелік питань, що потрібно опрацювати у роботі
1) аналіз об’єкту дослідження
2) аналіз хмарних технологій
3) розробити архітектуру хмарного сервісу для аналізу енергограм
4) розробити хмарний сервіс для аналізу енергограм

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд презентація – 18 слайдів

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання наукових матеріалів	09.11.21-12.11.21	
2	Аналіз наукових матеріалів	13.11.21-16.11.21	
3	Розробка архітектури сервісу	17.11.21-22.11.21	
4	Розробка програмного забезпечення	23.11.21-28.11.21	
5	Оформлення результатів роботи	29.11.21-02.12.21	
6	Подання кваліфікаційної роботи керівникові та	03.12.21-04.12.21	
	її попередній захист		
7	Подання кваліфікаційної роботи на рецензу-	05.12.21-06.12.21	
	вання		

Дата видачі завдання 8 листопада 2021 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Завизіступ Ю.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 82 с., 23 рис., 1 табл., 1 дод., 16 джерел.

ЗОБРАЖЕННЯ ЕНЕРГОГРАМ, ХМАРНИЙ СЕРВІС, ІНТЕРНЕТ, ПРОТОКОЛ, СЕРВЕР.

Метою кваліфікаційної роботи є огляд і розробка методів і моделей аналізу стану пацієнтів оздоровчих і відпочивальних закладів, розробка хмарного сервісу для автоматизації аналізу стану пацієнта оним з оглянутих методів.

У ході виконання кваліфікаційної роботи було зроблено огляд двох методів аналізу стану пацієнтів і розроблено сервіс для автоматизації аналізу одним з методів.

ABSTRACT

Master’s thesis: 82 pages, 23 figures, 1 tables, 1 appendices, 16 sources.

ENERGYGRAPH IMAGE, CLOUD SERVICE, INTERNET, PROTOCOL, SERVER.

The major goal of this thesis is a review of methods and models of analysis of the condition of patients of health and recreation facilities, the development of a cloud service to automate the analysis of the patient’s condition by one of the reviewed methods.

In order to review two methods of the analysis of a condition of patients and made and service for automation of the analysis by one of revieved methods.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ГРВ – газорозрядна візуалізація

ЕОМ – електро обчислювальні машини

БД – база даних

СКБД – система керування базами даних

API – програмний інтерфейс програмування (англ., Application Programming Interface)

JSON – джава скрипт об’єктна нотація (англ., Java Script Object Notation)

SQL – структурна мова запитів (англ., Structured Query Language)

ВСТУП

В наш час стало дуже актуальною тема покращення методів аналізу стану пацієнтів медичних закладів.

Перевантаженість медичної системи вимагає впровадження нових методів оцінки стану пацієнтів, які б були швидшими і розвантажили частину робітників медичних закладів.

Метою кваліфікаційної роботи є огляд існуючих методів оцінки стану пацієнтів засобами енергетичної діагностики, шляхом аналізу енергограм пацієнтів. А також розробка хмарного сервісу для автоматизації проведення оцінки стану пацієнта одним з розглянутих методів.

Предметом дослідження є знімки енергограм пацієнтів, що є основою використання методів енергетичної діагностики.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- проаналізувати існуючі методи енергетичної діагностики;
- проаналізувати побудову хмарного сервісу;
- розробити архітектуру хмарного сервісу;
- розробити хмарний сервіс для автоматизації оцінки стану пацієнта.

1 ОБ’ЄКТ ДОСЛІДЖЕННЯ

Об’єктами дослідження є методи оцінки стану пацієнтів оздоровчих і відпочивальних закладів, а саме, оцінка стану пацієнта методом розробленим і запатентованим кафедрою ЕОМ Харківського національного університету радіоелектроніки, на основі енергетичної діагностики та покращення цього методу шляхом інтеграції у процес оцінки стану пацієнта сучасних хмарних технологій.

В цій частині увага буде надана теоретичним відомостям про методи аналізу стану пацієнта медичного, або відпочивального закладу. Буде розглянута актуальність даного напрямку дослідження, та важливість продовження досліджень даної теми.

1.1 Актуальність дослідження

В останні роки в Україні та світі помічено кризу медичної системи, пов’язану зі спалахом вірусних захворювань. Згідно з даними представленими під час аналізу епідеміологічної ситуації по COVID-19 в Україні, за період з 12 березня по 10 жовтня 2020 року в Україні офіційно підтверджено випадків коронавірусної інфекції у 2179920 людей, з них 48048 пацієнта померли від ускладнень та 1027419 людей одужали [1].

Всі ці постраждалі люди створили величезне навантаження на медичні заклади. Обмеженість простору в медичних закладах, обмеженість співробітників, обмеженість апаратури і інших ресурсів стали серйозними проблемами. Але однією з найбільших проблем стали витрати часу на проведення діагностики, тож, наразі, очевидна необхідність покращення та оптимізації навантаження на медичні заклади.

В даній роботі буде розглянуто швидкий метод аналізу стану пацієнта, а також буде розглянуто спосіб оптимізації й пришвидшення використання

даного методу, що потенційно може мати позитивний вплив на навантаження медичної системи в її сучасних реаліях. Метод, що буде розглянутий далі є порівняно новим і не розповсюдженим, а також вимагає більше подальших досліджень, але його використання в комбінації з сучасними мережевими технологіями є потенційним шляхом розвантаження частини медичних працівників, шляхом автоматизації їх роботи.

Метод вимагає втручання персоналу лише на етапі збору даних пацієнта, подальший аналіз проводиться автоматично, без втручання персоналу. Тож, це не тільки дозволить обслугувати більше людей і, як результат, врятувати більше життів, а й зменшити навантаження на персонал медичного закладу.

Отже, актуальність роботи полягає у використанні потенційно дуже швидкого і ефективного методу діагностики стану пацієнтів. А в рамках даної роботи буде виконане покращення використання цього методу, шляхом пришвидшення діагностики що зробить метод ще привабливішим.

1.2 Енергетична діагностика

Методи, що будуть розглядатись далі, включно з методом запатентованим кафедрою ЕОМ, є методами енергетичної діагностики.

Дану категорію методів, на сьогоднішній день можна назвати популярною для діагностики біологічних і технічних об’єктів. Так, наприклад, енергетична діагностика в межах медицини дозволяє помічати тонкі зміни стану енергетики людини в рамках цілісного підходу до здоров’я людини.

Методи енергетичної діагностики є порівняно новими в медицині, але дуже перспективними в проведенні саме діагностик різного типу.

Серед переваг енергетичної діагностики можна виділити безпечність та відсутність протипоказань за віком, або за станом здоров’я. Діагностику можна проводити не обмежену кількість разів.

За основу методу взято дослідження світіння різних об’єктів, в тому числі біологічних, в електромагнітних полях високої напруженості. Дане явище було знайдено в 1930 – 40 роках подружжям Кірліан, тож і метод називають: кірліанографією, газорозрядною візуалізацією, або біоелектрографією.

Даний метод відомий вже понад два століття, однак складність апаратури не дозволяла проводити достатньо досліджень в цій області, тож лише зараз енергетична діагностика починає отримувати розповсюдження. З використанням сучасних потужностей сучасних комп’ютерів стало можливо продовжити і поглибити дослідження.

Наразі є певна кількість досліджень на цю тему, в тому числі і тих, які підтверджують наявність кореляції між параметрами світіння біологічних об’єктів і станом деяких систем органів (наприклад, серцево – судинної системи) [2] і активністю вегетативної нервової системи з врахуванням системи адаптаційних рівнів [3].

Даний ефект працює за таким принципом: якщо біоенергія людини є носієм інформації, то наслідком нормального перебігу енергії є гармонійне функціонування клітин, якщо з яких-небудь причин інформація змінюється і втрачає рівноважний ритм, то це повинно змінити та функціонування клітин.

В сучасних дослідженнях береться світіння, що виникає на пальцях людей при впливі поля високої напруженості, таке світіння називається енергограмою. Енергограми пальців кінцівок людини, на яких зосереджені початкові і кінцеві точки енергомерідіанів, відображають, інтегративний енергостан організму. Для аналізу зображень, використовується методи традиційної китайської медицини, згідно з якими, стан нервових закінчень і судин на кінчиках пальців відповідає стану органів та систем органів в тілі людини.

Отже, енергетична діагностика є сферею діагностики що активно розвивається і має високий потенціал у використанні в медичніх закладах і закладах відпочинку з метою проведення діагностування різного характеру.

1.3 Метод газорозрядної візуалізації

Метод газорозрядної візуалізації, є одним з методів енергетичної діагностики.

Метод енергетичного дослідження шляхом газорозрядної візуалізації (ГРВ), заснований на описаному вище ефекті Кірліан і потребує наявності спеціального обладнення для створення зображення. На далі, метод газорозрядної візуалізації буде називатись скорочено ГРВ.

Метод ГРВ дає можливість оцінити структурно-функціональний стан організму з отриманням стабільних відтворюваних результатів в реальному масштабі часу.

Для отримання зображення енергетичного поля пальців рук, пальці розміщуються на поверхні діелектричної пластини, на яку подаються імпульси напруги від імпульсного генератора. При високій напруженості поля в газовому середовищі простору контакту об’єкта (пальця) і пластини розвивається розряд у газовій фазі, що носить назву «ковзний газовий розряд», параметри якого визначаються властивостями об’єкта. Подальше світіння реєструється відеокамерами [4].

ГРВ-грами, що отримуються в ході дослідження, являють собою просторовий розподіл освітленості, який залежить від стану досліджуваного об’єкта.

Серед сфер використання ГРВ діагностики можна виділити такі [5]:

- массовий скринінг-діагностика захворювань (превентивна експрес-діагностика), її можна проводити серед здорового контингенту населення у рамках професійних оглядів у школах, на підприємствах тощо (це дозволяє виявити проблемні і навіть потенційно небезпечні зони організму);

- експрес-метод синдромної оцінки стану хворих на стаціонарах, лікарнях, диспансерах, санаторіях, що дозволяє різко обмежити коло діагностичного пошуку;

- підбір індивідуальних методів лікування на основі даних аналізу ГРВ-грам (передбачаються дослідження до та після впливу на організм людини лікарських препаратів, психотерапії, фізіотерапії);

- контроль стану пацієнта в процесі лікування (моніторинг стану хворих) та оцінка ефективності лікування (передбачає динамічну оцінку ГРВ-грам у процесі лікування);

- підбір ефективного підтримуючого лікування;

- первинна профілактика захворювань.

Отже, розглянутий метод ГРВ діагностики може використовуватись у цілій низці ситуацій, але має недоліки у вигляді необхідності використання коштовного обладнення. Тим не менш, метод має сфери використання і має потенціал в розширенні сфер і методів використання данного методу.

1.4 Метод отримання енергограм на основі поляризації діелектриків

Отримання енергограм на основі поляризації діелектриків є ще одним з методів енергетичної діагностики, запатентований кафедрою електро обчислювальних машин (далі ЕОМ) харківського національного університету радіоелектроніки.

ГРВ діагностика вимагає наявності коштовного і складного обладнення, що є серйозним недоліком даного методу. Альтернативою цьому методу є метод діагностики біологічних і технічних об’єктів на основі поляризації діелектриків.

Даний метод розроблений і запатентований кафедрою ЕОМ Харківського національного університету радіоелектроніки. Розробка належить до галузі діагностичної медицини та стосовується для дефектоскопії фізичних об’єктів.

Метод полягає у тому, що створюється приховане зображення розподілення зон електричної провідності на поверхні досліджуваного об’єкта за рахунок поляризації деяких видів діелектриків в імпульсному

високочастотному електричному полі. Зони об’єкта з більшою електричною провідністю створюють на поверхні діелектрика в зоні розподілення діелектрика з об’єктом більшу щільність зарядів і навпаки.

Напруга високовольтного джерела вмикається на фіксований час при натисканні на діелектричну пластину. Це дозволяє створити на поверхні діелектрика приховане зображення об’єкта дослідження в вигляді півтонового зображення, або енергограму, високої чіткості і контрасту, яке проявляється шляхом розпилення принтерного тонера на поверхні діелектрика і структура якого подібна структурі зображення, що отримується при газорозрядній візуалізації. Цей процес відбувається при значно менших напругах на практично всіх побутових прозорих упаковочних матеріалах: целофан, полістирол, органічне скло і т. і.

Крім того цей процес дає можливість одночасного отримання вимірів зі всіх пальців досліджуваного, що не може бути досягнуто легким шляхом в методі ГРВ, через складність і обмеженість необхідної для діагностики апаратури.

Отримане зображення може бути введене для аналізу в будь який комп’ютер, або перенесене на папір і закріплене нагріванням [6].

Сфери використання даного методу ідентичні методу ГРВ, але метод на основі поляризації дієлектриків має переваги:

- порівняна дешевизна (порівнюючи з методом ГРВ);
- швидкість процесу (за рахунок можливості зняття показів з всіх пальців одразу);
- використання меншої напруги мережі;

На рисунку 1.1 зображено пристрій для зняття енергограм з пальців рук пацієнта.

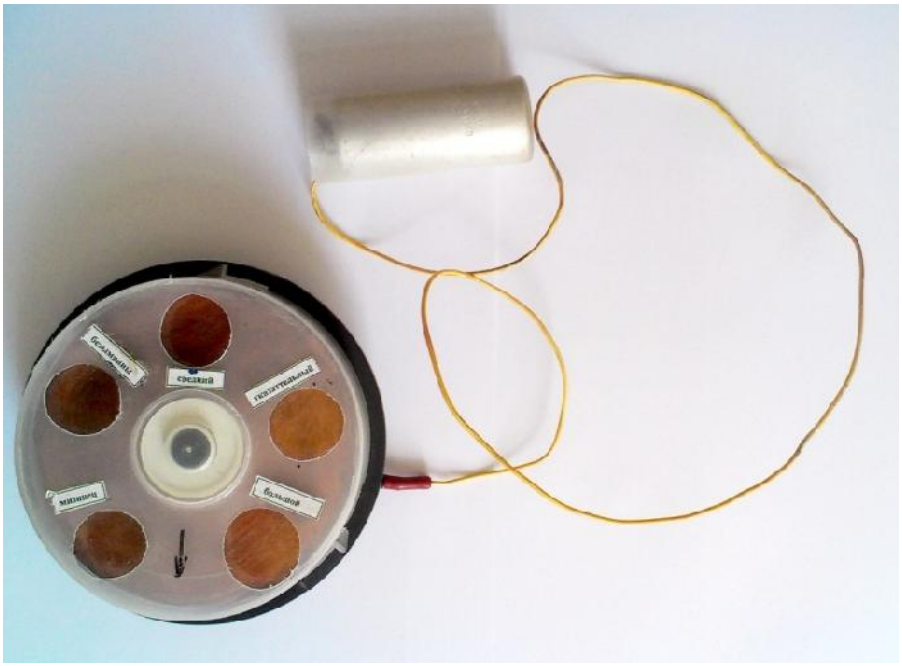


Рисунок 1.1 – Пристрій для зняття енергограм з пальців пацієнта

Після зняття показань з пальців, отримуються зображення, подібні до зображень отриманих в результаті ГРВ діагностики, приклад яких зображена на рисунку 1.2.

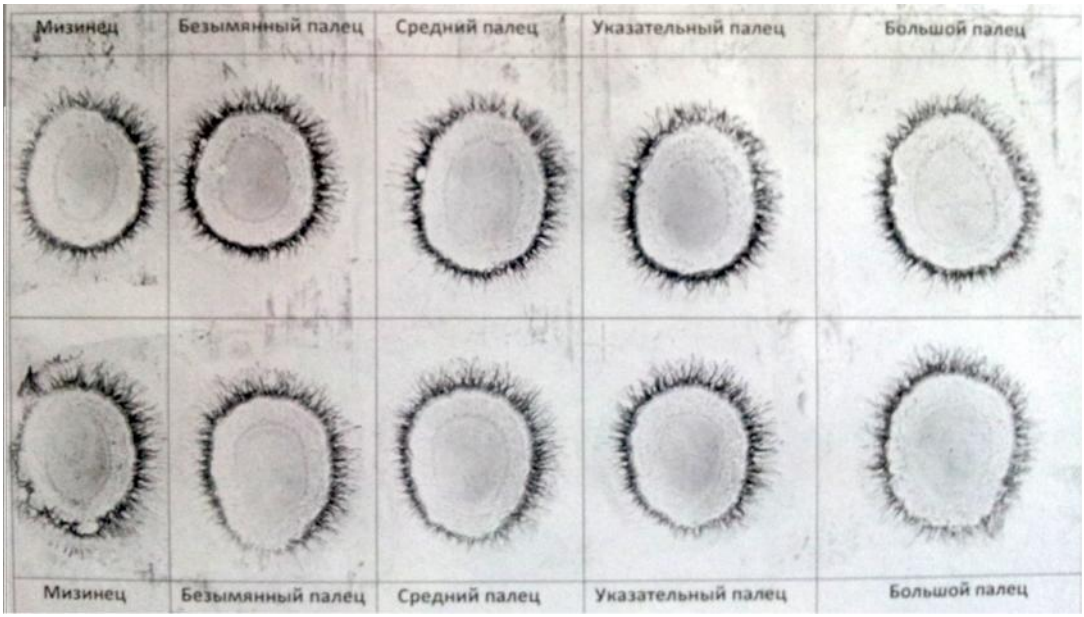


Рисунок 1.2 – Приклад зображень енергограм пальців пацієнта

Отже, даний метод оцінки стану пацієнта, є не тільки працездатним не гірше за описаний раніше ГРВ метод, а і має ряд переваг над ним. Використання даного методу є доступнішим для більшості медичних закладів і закладів відпочинку.

Після отримання зображень енергограм пацієнта у випадку обох методів необхідно проаналізувати дані і зрозуміти стан здоров’я пацієнта. Для цього необхідно скористатись методом секторальної діагностики, що дозволить надати сенс отриманим зображенням і дозволить зрозуміти стан пацієнта.

1.5 Метод секторальної діагностики

Для аналізу зображень отриманих шляхом методу ГРВ, або методу на основі поляризації дієлектриків, необхідно скористатись яким небуть методом аналізу зображень, щоб зрозуміти стан пацієнта.

Інтерпретація результатів отриманих енергограм пацієнтів може бути виконана методом описаним П. Менделєм, сутність якого полягає у комбінуванні енергограм пацієнтів з традиційною китайською медициною.

Відповідно до традиційної китайської медицини, а також безлічі клінічних досліджень, починаючи з 1996 року, стан судин і нервових закінчень на кінчиках пальців пов’язаний певним чином зі станом органів і систем у нашому організмі.

На кожному пальці є точки, пов’язані з певними органами та системами, стан яких можна оцінити завдяки аналізу енергограми [7]. Взаємозв’язки цих точок, органів та систем органів можна відобразити на рисунку 1.3.

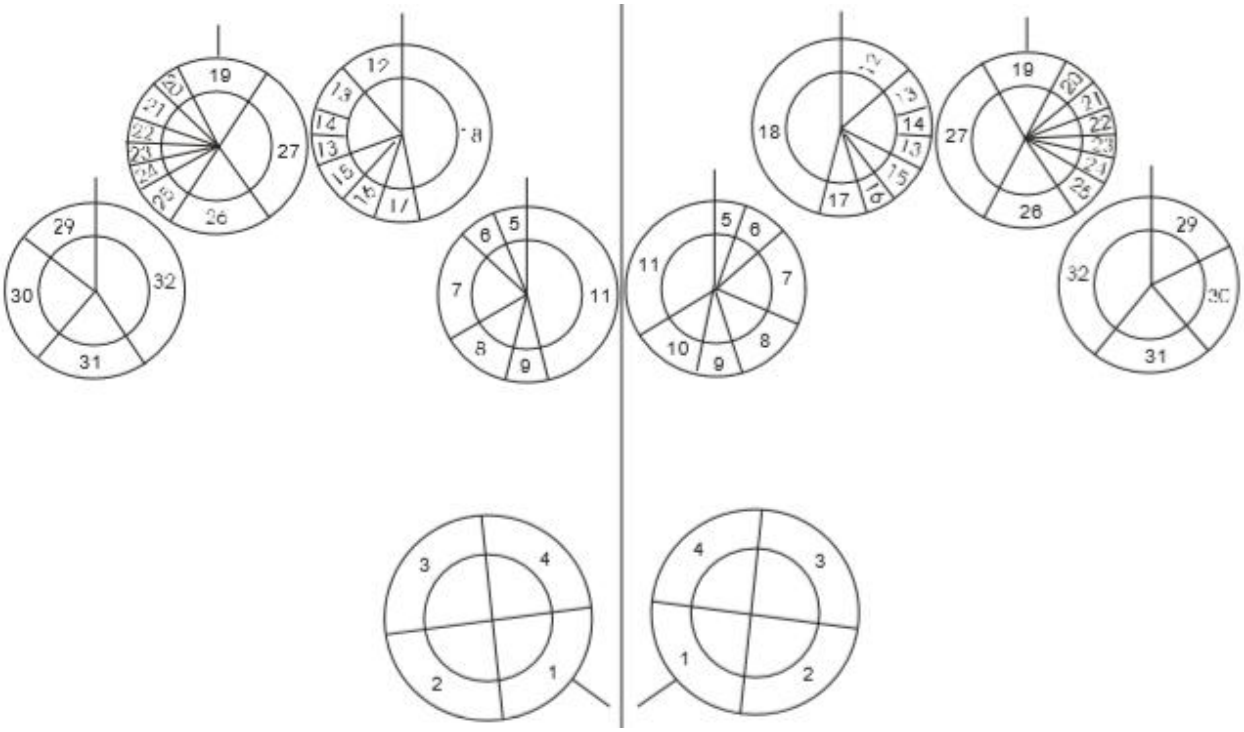


Рисунок 1.3 – Секторальна діаграма

Розшифровка даних зображених на секторальній діаграмі, має бути зроблена у відповідності до даних з таблиці 1.1, наведеної нижче.

У правій колонці вказані значення для секторів пальців правої руки, а у лівій значення для лівої. Підзаголовки позначають об’єднання секторів, що належать одному пальцю.

Таблиця 1.1 – Органи і системи в секторальної діаграмі пальців рук по системі П. Манделя.

Ліва рука	Права рука
1	2
Великий палець (1)	
1. Лобова пазуха, придаткових пазуха, верхня щелепа	1. Лобова пазуха, придаткових пазуха, верхня щелепа
2. Ніс	2. Ніс

Продовження таблиці 1.1

1	2
3. Нижня щелепа, лімфоглоткового кільце, вухо	3. Нижня щелепа, лімфоглоткового кільце, вухо
4. Мигдалик, шия	4. Мигдалик, шия
Вказівний палець (2)	
5. Нервова дегенерація-товстий кишечник	5. Нервова дегенерація-товстий кишечник
6. Головний мозок	6. Головний мозок
7. Грудний відділ спинного мозку	7. Грудний відділ спинного мозку
8. Поперечно-кресцовий відділ спинного мозку	8. Поперечно-кресцовий відділ спинного мозку
9. Пряма кишка	9. Копчик
10. Апендикс	10. Апендикс
11. Товстий кишечник	11. Товстий кишечник
Середній палець (3)	
12. Зона голови і очей	12. Зона голови і очей
13. Грудна клітка	13. Грудна клітка
14. Лімфа	14. Лімфа
15. Зона живота	15. Зона живота
16. Печінка (ноги)	16. Печінка (ноги)
17. Нирки	17. Нирки
18. Циркуляція крові каналу голованогі	18. Циркуляція крові каналу голованогі
Безіменний палець (4)	
19. Голова	19. Голова
20. Прищитовидні залози	20. Прищитовидні залози
21. Щитовидна залоза	21. Щитовидна залоза
22. Тимус	22. Тимус

Продовження таблиці 1.1

1	2
23. Підшлункова залоза	23. Підшлункова залоза
24. Наднирники	24. Наднирники
25. Яєчник (яєчко)	25. Яєчник (яєчко)
26. Матка (простата)	26. Матка (простата)
27. Нервова система (психіка)	27. Нервова система (психіка)
Мізинець (5)	
28. Сліпокишкова заслінка	28. Гастро-дуоденальна зона
29. Клубова кишка	29. Тонка кишка
30. Застійна зона лімфи, молочні заліза, легені, бронхи	30. Застійна зона лімфи, молочні заліза, легені, бронхи
31. Серце	31. Серце

Після розбиття зображення на сектори можна співставити сектори з даними таблиці для того, аби зрозуміти за що відповідає кожен сектор, а отже і кожна частина зображення енергограми пальців рук пацієнта.

Отже, отримані в результаті діагностики зображення енергограми, можна розбити на ділянки, відповідно до того, за що який орган, чи систему органів відповідає кожна ділянка зображення.

1.6 Аналіз енергограми пацієнта

Маючи зображення енергограми пацієнта і знаючи за що відповідає кожна з ділянок цього зображення, залишається лише розрахувати показники стану органів, чи систем органів пацієнта.

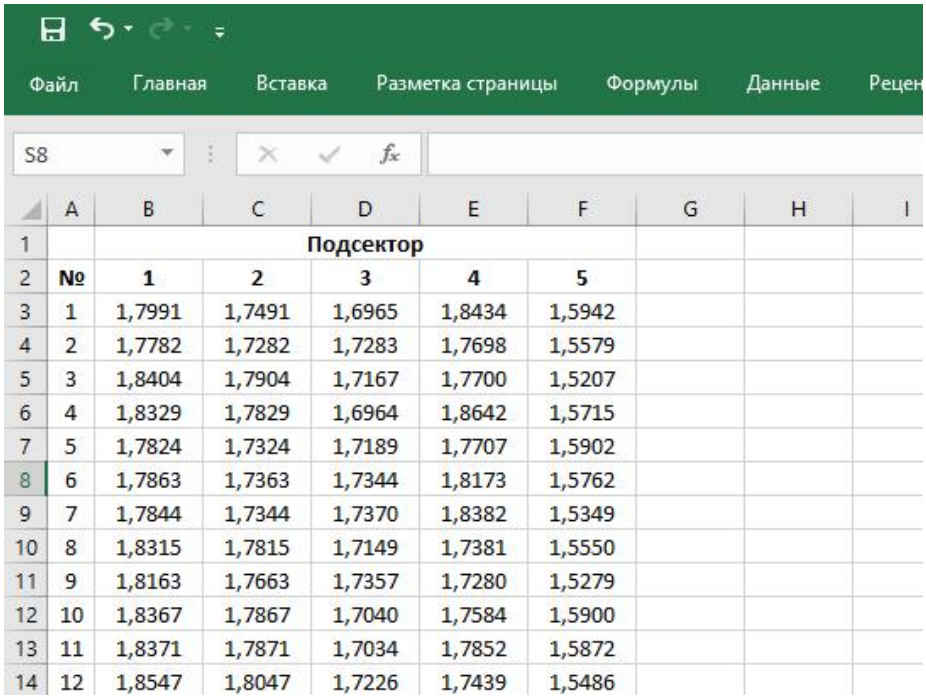
Після отримання зображень енергетичного поля пальців рук пацієнта, необхідно правильно застосувати метод секторальної діагностики. Для цього необхідно окрестити кожен знімок колом і розмітити радіальні кути, що вже були показані на рисунку 1.3 попередньої частини. Використовуючи данні з

вказанної вище таблиці необхідно правильно розмітити точки, що відповідають певним органам.

Отримані точки не дають розуміння про стан пацієнта, отже для того щоб зрозуміти що саме свідчать зібрані дані, необхідно співставити їх з еталонними значеннями.

Для подальшої роботи з результатами аналізу, а саме класифікації енергограми, що досліджується, нам знадобиться статистична вибірка еталонних значень досить великого обсягу – 50-100 значень для правильного визначення меж нечітких відносин.

Для статистичного аналізу використовується програмне забезпечення Microsoft Office Excel 2016.



	A	B	C	D	E	F	G	H	I
1		Подсектор							
2	№	1	2	3	4	5			
3	1	1,7991	1,7491	1,6965	1,8434	1,5942			
4	2	1,7782	1,7282	1,7283	1,7698	1,5579			
5	3	1,8404	1,7904	1,7167	1,7700	1,5207			
6	4	1,8329	1,7829	1,6964	1,8642	1,5715			
7	5	1,7824	1,7324	1,7189	1,7707	1,5902			
8	6	1,7863	1,7363	1,7344	1,8173	1,5762			
9	7	1,7844	1,7344	1,7370	1,8382	1,5349			
10	8	1,8315	1,7815	1,7149	1,7381	1,5550			
11	9	1,8163	1,7663	1,7357	1,7280	1,5279			
12	10	1,8367	1,7867	1,7040	1,7584	1,5900			
13	11	1,8371	1,7871	1,7034	1,7852	1,5872			
14	12	1,8547	1,8047	1,7226	1,7439	1,5486			

Рисунок 1.4 – Вибірка еталонних значень в програмі Microsoft Office Excel 2016

Маючи статистичні дані можна визначити нечіткі межі відносин в рамках кожного підсектору секторальної діаграми.

Після аналізу статистичних даних і визначення меж норми можна сміливо побудувати розмітку що демонструє стан пацієнта.

Приклад такої розмітки наведено на рисунку 1.4.

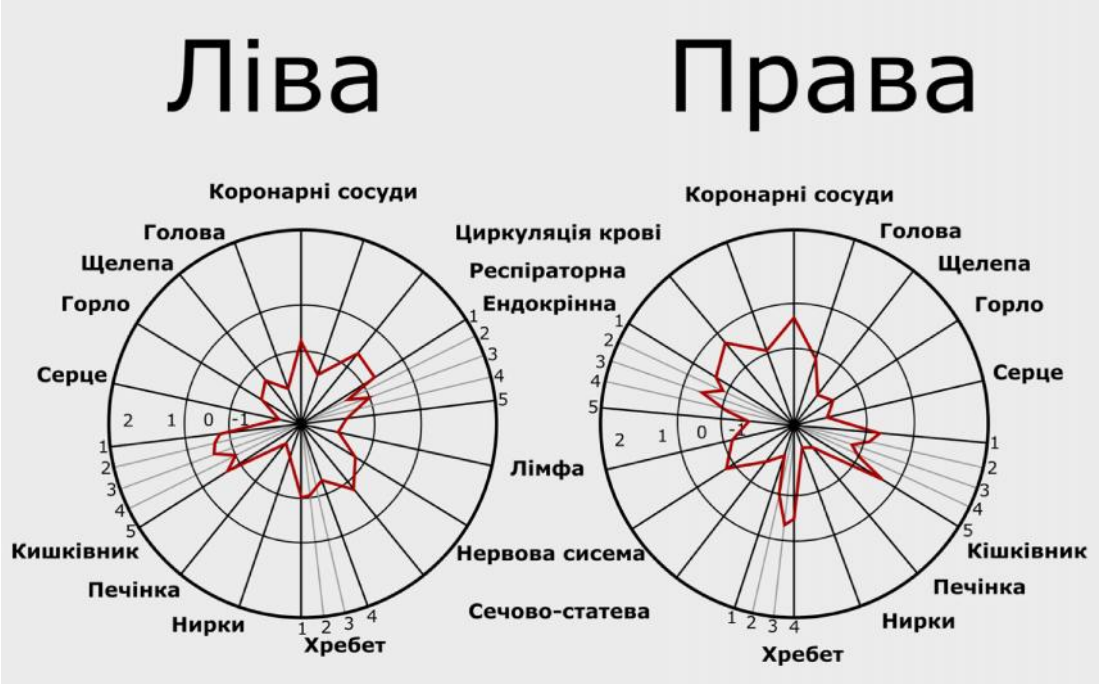


Рисунок 1.5 – Секторальна діаграма

Співставлення даних з таблиці вручну є дуже важким завданням, враховуючи необхідність вручну будувати секторальну діаграму. Це і є основним недоліком методу.

Оскільки сучасні технічні засоби здатні самотужки обробляти зображення, а процес співставлення діаграми з даними таблиці є тривіальною, хоч і рутинною задачею, немає жодних причин не створити програмний засіб, що міг би автоматизувати цей процес.

Отже, після отримання енергограм пацієнта, їх можна перевести в електроний формат шляхом сканування, або фотографування, після чого скористатись програмним засобом, що сам зможе побудувати секторальну діаграму і співставити її значення зі значеннями у таблиці. Дана операція вимагає багато часу, що і є сильним недоліком даного методу.

1.7 Автоматизований аналіз енергограм

Виконання дій, описаних вище, для проведення діагностики стану пацієнта, може бути витратним у часі, але найбільша перевага методів енергетичної діагностики і полягає у можливості автоматизації цих дій.

1.7.1 Програмний аналіз енергограм

Для автоматизації аналізу зображень енергограм пацієнта можна створити програмний засіб, що буде виконувати як метод секторального аналізу, так і співставлення з еталоннвмв значеннями, для того, щоб зрозуміти стан пацієнта.

Для автоматизації необхідне програмне забезпечення, яке змогло б виконати попередню обробку зображення, розбиття на сектори та співставлення з таблицею 1.1.

Попередня обробка складається з:

- лінійного контрастування;
- порогової обробки напівтонових зображень;
- виділення контурів.

Подальша обробка вимагає використання методів, що дозволять створити секторну діаграму.

На щастя, для методу створення енергограм за допомогою поляризації діелектриків було розроблене спеціальне програмне рішення. Це програмне рішення було спеціально створено для цього методу створення енергограм і виконує всі необхідні попередні етапи, так само як і робить сам аналіз.

Дане програмне рішення було розроблене в рамках іншої кваліфікаційної роботи [8] і в рамках неї ж воно було протестоване і, таким чином, довело свою працездатність.

Програмне рішення дозволяє отримати результати аналізу енергограм, шляхом завантаження зображень енергограм пацієнта в програму,

встановлення опціональних налаштувань і дозволяє отримати результат аналізу у текстовому форматі і у форматі json файлу.

Програма видає ступінь відхілення показників секторів енергограми з відповідних органів і систем органів від еталонних значень. Також, дозволяє порівняти результати поточного і наступних досліджень.

Недоліком цього програмного рішення є неможливість використання програмного засоба на портативних, мобільних присторях, через технології покладені в основу програмного рішення (немає підтримки інших плтформ). Також, обробка зображень може використовувати багато системних ресурсів, що може унеможливити портування цієї програми на мобільні пристрої.

Враховуючи даний програмний засіб, процес аналізу енергограм вже є доволі простим, але його все ще можна покращіти шляхом інтеграції сучасних хмарних технологій. Сучасні хмарні технології дозволяють навіть відносно мало потужним мобільним пристроям бути використаними для діагностики стану пацієнта. Шляхом перенесення всіх важких обчислень на потужні комп’ютери, та використання мобільних пристроїв лише для обміну даними, можна покращити існуючий програмний засіб.

Отже, виконання аналізу зображень енергограм пацієнтів може бути автоматизоване програмним засобом, який може виконати всі необхідні завершуючі етапи діагностики і показати готові результати.

1.7.2 Хмарні технології як засіб автоматизації

Програмний засіб для аналізу зображень енергограм пацієнтів вимагає певних потужностей від крмп’ютера на якому цей програмний засіб буде працювати. Також, проблемою може мтати необхіжність портування даного програмного засобу на, наприклад, мобільні пристрої, які є дешевшими і здатними робити фотографії енергограм пацієнта, тобто ойіфровувати енергограми для подальшого аналізу у програмному засобі, що може значно пришвидшити процес проведення діагностики стану пацієнта.

Для того, щоб подолати проблему портування програми на мобільні пристрої, можна створити хмарний сервіс, який міг би робити аналіз зображень енергограм і повертати результат на мобільний пристрій який зробив зображення.

Таким чином, складні розрахунки будуть перенесені на потужні комп’ютери сервісу, а користувачі, медичні працівники, зможуть спростити процес аналізу енергограм, до простого фотографування енергограм на мобільний пристрій і надсилання зображень на обробку до сервісу.

В сучаних реаліях немає потреби у створенні власної хмарної інфраструктури, адже існують вже готові рішення, що дозволять економити на підтримці і обслуговуванні комп’ютерів. З боку медичного закладу потрібно лише розгорнути сам сервіс на будь-якій з готових хмарних платформ, після чого процес аналізу даних пацієнта стане простою задачею, що не буде вимагати спеціальних знань з боку працівників.

Використання хмарного сервісу дозволить зменшити навантаження на медичних працівників, дозволить обійтись без довготривалого ручного аналізу енергограм і зекономить кошти медичного сервісу на покупці власної системи комп’ютерів. Дозволить використовувати дешеві та малопотужні мобільні пристрої для проведення діагностики енергограм пацієнтів.

Отже, для подолання слабкостей програмного засобу і покращення роботи сервіса, необхідно створити хмарний сервіс, для обробки зображень енергограм пацієнтів.

1.8 Постановка задачі

Наразі, було розглянуто методи енергетичної діагностики з їх перевагами, слабкостями і методами аналізу даних отриманих в результаті діагностики. Були розглянуті способи автоматизації аналізу зображень енергограм пацієнтів і можливість покращення аналізу, шляхом

впровадження хмарних технологій у процес аналізу зображень енергограм пальців пацієнтів медичних закладів і закладів відпочинку.

Розглянутий метод діагностики стану пацієнтів шляхом отримання енергограм на основі поляризації діелектриків є перспективним для сучасної медицини. А наявність програмного засобу спрощує і пришвидшує процес діагностики. Але процес діагностики може бути покращений ще більше, шляхом використання сучасних хмарних технологій.

В межах даної роботи стоїть задача покращити процес виконання аналізу енергограм пацієнтів запатентованим кафедрою ЕОМ методом на основі поляризації діелектриків, шляхом впровадження сучасних хмарних технологій. Розробки архітектури хмарного сервісу, що може бути використаний медичним чи відпочивальним закладом для аналізу енергограм пацієнтів. Реалізація демонстраційного хмарного сервісу, що продемонструє можливості розробленої архітектури.

В кінці даної кваліфікаційної роботи буде зроблено тестування розробленого хмарного сервісу, в тому числі з кількома пристроями.

1.9 Висновки

Було розглянуто два методи оцінки стану пацієнтів оздоровчих і відпочивальних закладів шляхом проведення енергетичної діагностики. Метод газорозрядної візуалізації і метод на основі поляризації діелектриків, що є розробкою кафедри ЕОМ. Метод аналізу енергограм на основі поляризації діелектриків є дешевшим для використання в медичних закладах. Тому буде використаний в основі майбутнього хмарного сервісу.

З рештою, була сформульована задача для подальшого виконання в рамках цієї кваліфікаційної роботи. А саме, розробити хмарний сервіс для автоматизації аналізу енергограм пацієнтів обраним методом.

2 АНАЛІЗ ХМАРНИХ ТЕХНОЛОГІЙ І ПІДБІР МЕТОДІВ

Для того, аби інтегрувати сучасні хмарні технології в процес оцінки стану пацієнта оздоровчого чи відпочивального закладу необхідно проаналізувати з чого складається типовий хмарний сервіс, з чого складаються сучасні хмарні сервіси і які технології можна використати для побудови підходящого сервісу.

2.1 Можливості хмарних технологій

Перед тим як аналізувати склад типового хмарного сервісу, має сенс розглянути можливості хмарних технологій. Це дозволить зрозуміти переваги і всю важливість впровадження даних технологій в процес аналізу зображень енергограм пацієнтів медичних і відпочивальних закладів.

Хмарні обчислення є моделлю забезпечення повсюдного та зручного мережевого доступу на вимогу до загального пулу конфігуруємих обчислювальних ресурсів, таких як мереж передачі даних, серверів, пристроїв зберігання даних, додатків і сервісів як разом, так і окремо, які можуть бути оперативно надані й звільнені з мінімальними експлуатаційними витратами.

Основними перевагами є:

- немає потреби у великих обчислювальних потужностях клієнтських комп’ютерів;
- відмовостійкість;
- висока швидкість обробки даних;
- економія коштів на підтримку обладнення (у випадку оренди серверів);
- легке масштабування сервісу (у випадку оренди серверів, або наявності власної хорошої хмарної інфраструктури).

Але є і недоліки, такі як:

- необхідність завжди бути в мережі для роботи;
- відсутність контролю над данними у випадку оренди серверів.

З точки зору споживача, ці характеристики дозволяють отримати послуги з високим рівнем доступності і низькими ризиками непрацездатності, забезпечити швидке масштабування обчислювальної системи завдяки еластичності без необхідності створення, обслуговування і модернізації власної апаратної інфраструктури [9].

Зручність і універсальність доступу забезпечується широкою доступністю послуг і підтримкою різного класу термінальних пристроїв (персональних комп'ютерів, мобільних телефонів, інтернет-планшетів).

Національним інститутом стандартів і технологій США (ANSI) зафіксовані такі обов'язкові характеристики хмарних обчислень:

- самообслуговування за вимогою;
- універсальний доступ до мережі;
- еластичність, послуги можуть бути надані, розширені, звужені в будь-який момент часу, без додаткових витрат на взаємодію з постачальником, як правило, в автоматичному режимі;
- облік споживання;

За моделлю розгортання можна виділити такі хмари:

- приватна хмара, що призначена для використання однією організацією;
- публічна хмара, що призначена для вільного використання широкою публікою;
- гібридна хмара, що є комбінацією з двох або більше різних хмарних інфраструктур (приватних, публічних або суспільних), що залишаються унікальними об'єктами, але пов'язаних між собою стандартизованими або приватними технологіями передачі даних і додатків;

- суспільна хмара, що призначена для використання конкретною спільнотою споживачів з організацій, що мають спільні завдання (наприклад, місії, вимоги безпеки, політики, та відповідності різним вимогам).

За для забеспечення приватності даних і безпеки, для будь-якого медичного, або відпочивального закладу підійде саме приватна хмара.

Отже, були розглянуті можливості і переваги використання хмарних технологій в сучасних проектах, що дозволяє оцінити і важливість використання хмарних технологій в процесі аналізу зображень енергограм пальців рук пацієнтів мадичних закладів і закладів відпочинку.

2.2 Структура хмарного сервісу

Розберемось із структурою типового хмарного сервісу, для того, аби розуміти складові частини які буде необхідно реалізувати, або використати в рамках роботи.

Типовий хмарний сервіс розгортається на певних комп’ютерах, які частіше всього об’єднані у мережу. На системі комп’ютерів розгортається віртуальна машина в середовищі якої і буде працювати сервіс. Більша частина хмарних сервісів так чи інакше збирають данні користувачів для різного типу потреб, тому в більшості ситуацій наявна база даних.

Для комунікації з сервісом клієнту, як правило, необхідно встановити клієнтський застосунок, або запустити веб застосунок, для отримання можливості комунікувати з сервісом.

На рисунку 2.1 можна побачити приклад типового хмарного сервісу.

Задача інфраструктури створити середовище в якому буде працювати сервіс. У випадку мережі комп’ютерів об’єднаних в одну мережу, інфраструктура має створити ілюзію використання одного середовища, простий інтерфейс для розгортання серверу і бази данних в ньому [9].

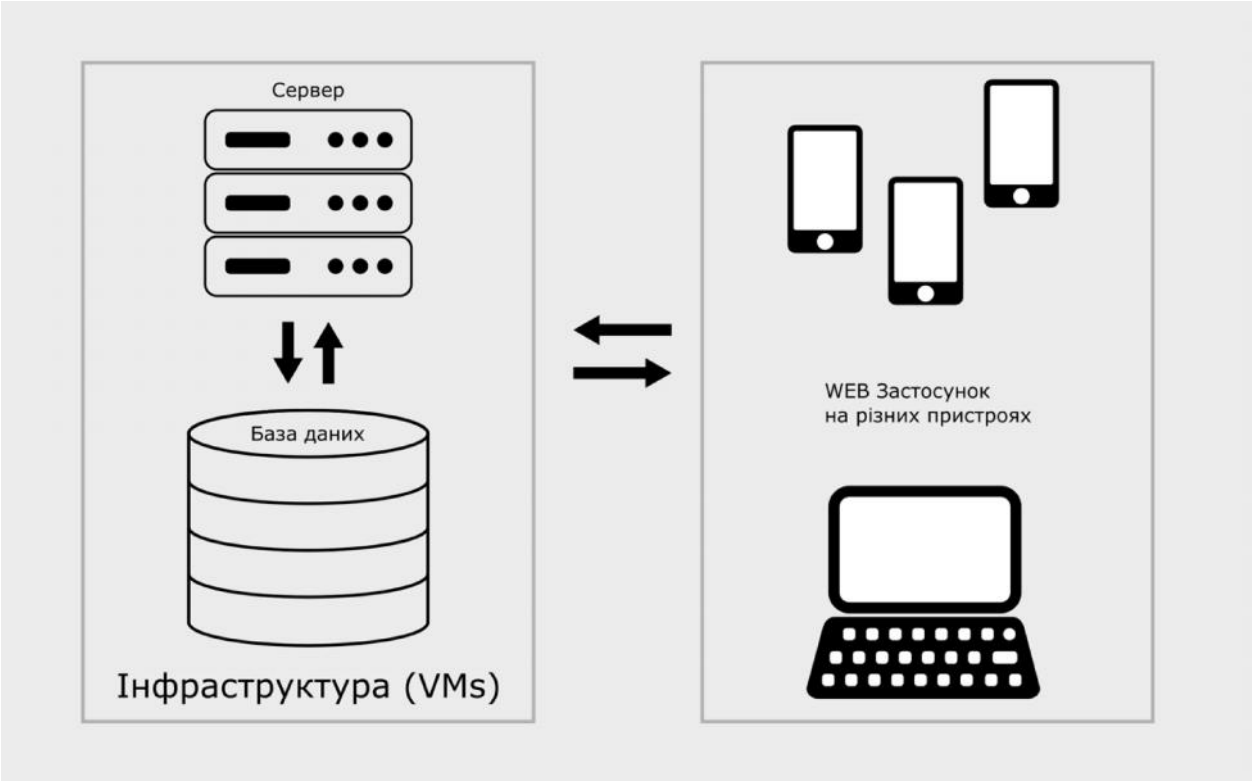


Рисунок 2.1 – Приклад типового хмарного сервісу

На рисунку 2.1 можна побачити приклад типового хмарного сервісу.

Задача інфраструктури створити середовище в якому буде працювати сервіс. У випадку мережі комп’ютерів об’єднаних в одну мережу, інфраструктура має створити ілюзію використання одного середовища, простий інтерфейс для розгортання серверу і бази данних в ньому [9].

В свою чергу основна задача бази даних, у випадку медичних закладів, забезпечити безпеку даних користувача.

Сервер, розгорнутий в хмарному сервісі має обробляти запити клієнтів і може мати в собі компоненти необхідні для аналізу і обробки даних користувача.

Клієнтський застосунок має створювати інтерфейс комунікації з хмарним сервісом, для передачі даних користувача на обробку і повернення результатів обробки користувачу.

Отже, було розглянуто складові типового хмарного сервісу і спосіб передачі даних між складовими сервісу.

2.2.1 Інфраструктура хмарного сервісу

Інфраструктура для хмарного сервісу є набором комп’ютерів що процюють з’єднаними у одну мережу і постачають простір для розгортання сервісу. В сучасних реаліях найбільш надійний спосіб отримати налаштовану і працюючу інфраструктуру, це орендувати готові сервери. Це позбавляє необхідності підтримувати комп’ютери самотужки і турбуватись про потужності. Сервіси що здають сервери в оренду мають готові інструменти для налаштування потужності і самі турбуються про працездатність серверів.

Отже, інфраструктура є важливою основою для роботи сервіса.

2.2.2 База даних

Для зберігання даних пацієнтів і результатів аналізу зображень енергограм пальців пацієнтів закладів відпочинку і лікування необхідно створити базу даних.

Існують різні моделі баз даних:

- ієрархічні;
- мережні;
- реляційні.

Розглянемо реляційні як найбільш придатні для сервісів з простою структурою даних, де не буде необхідності часто поєднувати таблиці з яких складається база даних.

Прикладом системи керування реляційними базами даних може слугувати MySQL, розглянемо її як приклад.

MySQL – відкрита система керування реляційними базами даних. Ця система керування базами даних скорочено (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. Зараз MySQL – одна з найпоширеніших СКБД. Вона використовується, в першу чергу, для створення динамічних веб сервісів, оскільки має чудову підтримку з боку

різних мов програмування. MySQL надає багато функцій та можливостей, які підтримують безпечну середу для зберігання, обслуговування та отримання даних [10].

Перевагами системи керування базами даних MySQL є:

- масштабованість. MySQL може підтримувати роботу БД значних розмірів, що підтверджують її реалізації в таких копаніях як Yahoo!, Google, HP, Associated Press. Згідно з документацією, що додається до MySQL, деякі БД, що використовуються компанією MySQL AB (компанією розробником MySQL), зберігають до 50 млн. одиниць записів;

- переносність. MySQL здатна працювати на різних платформах, серед яких Unix, Linux, Windows, OS/2, Solaris, Mac OC. Що робить СКБД хорошим рішенням для переносимих рішень.

- зв’язаність. MySQL має мережеву структуру. До MySQL можна одержувати доступ із будь-якої точки Internet кільком користувачам одночасно. MySQL має цілий ряд програмних інтерфейсів додатків (Application Programming Interface –API), які дозволяють встановлювати з’єднання з MySQL із додатків, написаних на таких мовах як C, C++, Perl, PHP, Java, Python;

- безпека. MySQL має систему контролю доступу до даних, забезпечує шифрування даних при передаванні;

- швидкість функціонування;

- зручність експлуатації. MySQL досить зручно встановлюється та реалізується, легко адмініструється;

- відкритий код.

Таким чином, можна зробити висновок, що розглянуті програмні засоби є достатніми для використання, в хмарних системах для обробки даних клієнтів оздоровчих чи відпочивальних закладів.

Отже, база даних є одним з ключових компонентів хмарного сервісу, адже будь який сервіс потребує зберігати дані користувачів і результати роботи сервісу.

2.2.3 Сервер

Для обробки запитів і роботи з базою даних необхідна зв’язуюча ланка, якою і є сервер. Саме сервер буде надавати досткп до браузерного Web застосунку, приймати і зберігати дані пацієнтів, проводити аналіз і повідомляти про результати. Тож, сервер є однією з ключових частин сервісу і має покладатись на хороші базові технології.

Розглянемо одну з сучасних базових технологій для створення серверів Node.js.

Node.js – це серверна платформа, що призначена для створення масштабованих розподілених серверних додатків, яка використовує подієвоорієнтовану архітектуру та неблокуючу асинхронну взаємодію. Головна ідея Node.js: використання неблокуючого подієво-орієнтованого вводу та виводу, щоб залишатися легким та ефективним під час роботи з додатками, що обробляють великі обсяги даних в реальному часі і функціонують на розподілених пристроях. Все це означає, що Node.js не є платформою на всі випадки життя, яка буде домінувати у світі веб розробки. Навпаки, вона використовується лише для вирішення завдань певного типу. Якщо ваш додаток використовує тяжкі обчислення, то вам варто відмовитися від використання Node.js, тому що дані операції анулюють всі переваги даної платформи. Та якщо ви створюєте систему, яка підтримує взаємодію з багатьма одночасними запитами, то рано чи пізно вам знадобиться організація асинхронного доступу [10].

При синхронному запиті виникає «блокування», яке виникає під час встановлення з’єднання з ресурсом і (особливо) під час зчитування з нього даних. Під час такого блокування потік виконання простоює, і це, безумовно, не є раціональним використанням ресурсів. В той час як асинхронна модель вирішує проблеми доступу принципово по іншому. Вона побудована на циклічній черзі подій (eventloop). При виникненні деякої події (прийшов запит, відповідь від бази даних, зчитування порції даних з файлу) вона

розміщується в кінець черги. Потік виконання, що обробляє цей цикл, виконує код, пов’язаний з наступною подією, і, своєю чергою, розміщує його в кінець.

Це відбувається до тих пір, поки черга не спорожніє. Ще однією перевагою є легкий спосіб додавання підтримки HTTPS запитів, що є критичним для системи документообігу. Саме тому використання Node.js буде найбільш раціональним рішенням [10].

Задля полегшення роботи над сервером необхідно використати фреймворк для Node.js із заздалегідь створеними необхідними налаштуваннями, що полегшить написання коду. Найбільш розповсюдженим є фреймворк “Express”.

Експрес – популярний непідприємницький фреймворк, написаний на JavaScript та розміщена в середовищі виконання node.js.

Він забезпечує механізми:

- створення обробників для запитів з різними HTTP-запитами та різними URL-адресами (маршрутами);

- інтеграції з двигунами візуалізації, щоб генерувати відповіді, вставляючи дані в шаблони;

- встановлення загальних параметрів веб додатків, таких як порт, який буде використовуватися для з’єднання, та розташування шаблонів, які використовуються для надання відповіді.

Хоча сам Express досить мінімалістичний, розробники створили сумісні пакети програмного забезпечення для вирішення майже будь-якої проблеми веб розробки.

Є бібліотеки для роботи з файлами cookie, сеансами, входами користувачів, параметрами, URL-адресами, даними POST, заголовками безпеки та багато іншого.

Для медичного чи відпочивального закладу переваги Node.js будуть доречними для обробки кількох запитів на обробку даних пацієнта.

Отже, сервер є однією з найважливіших частин хмарного сервісу і має

виконувати багато задач, тому до цієї частини сервісу потрібно приділяти велику кількість уваги при розробці.

2.2.4 Веб застосунок

Для комунікації з користувачем хмарного сервису необхідно використати клієнтський застосунок, що і буде надсилати, та приймати дані до серверу.

В якості клієнського застосунку має бути створений застосунок як для мобільних платформ, так і для комп’ютерів. Оскільки підтримка кількох застосунків для різних платформ є дорогою опцією, доречно буде створити браузерний застосунок, або веб додаток, який може бути використаний на будь-якій платформі, його легко оновлювати і розробка якого не буде коштувати дорого.

Веб додаток створюється для того, щоб користувач мав змогу отримати доступ до послуг сервісу. При створенні веб додатків можливості програміста стримуються тим, які технології підтримує браузер користувача, саме тому у програміста немає альтернативи використанню HTML для створення сторінки, CSS для стилізування і JS для створення корисної логіки для додатку.

HTML – це не мова програмування, це мова розмітки, яка розмічає браузеру, як зображувати вміст вебсторінки. HTML складається з серії (тегів), які використовуються для вміщення, або «загортання» в них різних частин контенту, щоб вони зображались чи діяли в певний спосіб. Ці елементи, за допомогою початкових та кінцевих (тегів), можуть зробити слово чи зображення посиланням на будь-яку іншу сторінку, можуть відображати текст курсивом, збільшити чи зменшити шрифт тощо.

Як і HTML, CSS не є справжньою мовою програмування. Це лише мова таблиць стилів, яка дозволяє задавати стилі обраним елементам у HTML документах.

JavaScript – крос-сплатформна об’єктно-зорієнтована мова сценаріїв (скриптів). Рушій JavaScript підключається до об’єктів свого середовища виконання (зазвичай, вебпереглядача) та надає можливість керування ними. Мова JavaScript має стандартну бібліотеку об’єктів (таких як Array, Date та Math) і основний набір елементів мови програмування, таких як оператори, керівні структури та вирази. Ядро JavaScript може бути розширене для різних потреб шляхом доповнення його додатковими об’єктами.

Для автоматизації процесів при створенні браузерного веб додатку має сенс використати фреймворк Angular.

Angular – JavaScript фреймворк для розробників веб додатків від Google.

Важливим фактом на користь використання цього фреймворку є те, що він має Angular CLI для генерування нових складових для проекту, що пришвидшує написання веб додатку.

Перша версія Angular, що називається AngularJs, була багато у чому не оптимальною. Однак величезна кількість проєктів і досі використовують цю версію. Згодом команда розробників фактично з нуля переписала проект. Нова версія отримала назву Angular 2 чи просто Angular без індексу. Вона кардинально відрізнялась від першої, справляючи враження зовсім іншого фреймворку [10].

Проекти не могли з легкістю мігрувати зі старої версії на нову, тому це зумовило певні проблеми переходу на нову версію. Angular став концептуально простішим, більш технологічним та зрозумілішим.

Варто помітити, що ніяких кардинальних змін з виходом версії 2 більше не відбувається. Усі версії, з другої до сьомої, повністю сумісні.

В основному, Angular призначений для створення комплексних enterprise - додатків, а саме односторінкових веб застосунків (Single Page Application), але це зовсім не означає, що цей фреймворк неможна використовувати в інших рішеннях, навпаки, цей фреймворк дозволяє спростити велику кількість тривіальних дій, створити у додатку систему

розподілення сторінок, розбити сторінку на компоненті для повторного використання [10].

Всередині фреймворку реалізовано:

- модульність;
- анімації;
- маршрутизація;
- робота з сервером;
- зберігання, обробка, зображення даних;
- робота з формами та шаблонами тощо.

Single Page Application (SPA чи односторінковий застосунок) реалізує зручні для користувача сервіси, що наповнені інтерактивом. Найпростішими прикладами є Outlook, Gmail. SPA – веб додаток, розташований на одній фізичній HTML сторінці. Така сторінка одноразово завантажує усі необхідні ресурси (JavaScript, CSS, images тощо.) і більше не перезавантажується. Переходи за посиланнями не призводять до реального перезавантаження сторінки, а її вміст змінюється «на льоту», тобто динамічно. За необхідністю виконується запит на сервер для отримання даних, і після їх отримання формується контент «нової» сторінки. Кожна окрема віртуальна сторінка прописується у маршрутизаторі.

Треба зазначити, що Angular застосунки пишуться на TypeScript, що підходить для проекту.

Версія синтаксису для JavaScript не отримала широкого розповсюдження, тому на цей час у документації усі синтаксичні конструкції описані з використанням синтаксису TypeScript.

Архітектура Angular складається з:

- module;
- component;
- template;
- service;
- router;

- pipe;
- directives.

Модулі (Module) – структурні одиниці застосунку, які інкапсулюють певну логіку. В Angular це структури, які зберігають певні компоненти, директиви та сервіси, об’єднані певною логікою [10].

Компоненти (Component) – typescript клас, який зберігає дані та логіку зображення цих даних у шаблоні (представленні).

Шаблон тісно пов’язаний з компонентом. Дані з компонента можна з легкістю відображати у шаблоні, використовуючи спеціальний синтаксис. Компонент також може «знімати» дані з шаблона та отримувати їх безпосередньо у скрипті.

Шаблон (Template) – фрагмент html-коду з додаванням спеціального синтаксису. Він дозволяє впроваджувати в шаблон дані з компонента без використання innerHTML та подібних методів. Шаблон прописується у компоненті та є частиною його конфігурації.

Сервіс (Service) в Angular являє собою typescript класи, які виконують задачі, пов’язані з отриманням, зберіганням та обробкою даних. Наприклад, логування, перетворення даних для подальшої передачі у компонент, звернення до backend.

На відміну від компонентів та директив сервіси не працюють з представленнями (шаблонами) напряму.

Задачі сервісів:

- надання даних застосунку. Сервіс сам може зберігати дані у пам’яті або, з метою отримання даних, звертатися до якогось джерела даних, наприклад, до сервера;
- сервіс може організувати канал взаємодії між окремими компонентами застосунку;
- сервіс може інкапсулювати бізнес-логіку, різноманітні обчислювальні задачі, задачі з логування, які краще виносити поза компоненти. Таким чином, код компонентів буде зосереджений,

безпосередньо, на роботі з представленням. До того ж, можемо розв’язати проблему повторення коду, якщо нам знадобиться виконати одну й ту саму задачу у різних компонентах і класах.

Роутер (Router) – маршрутизатор, який призначений для переходу між екранами з метою відображення різного контенту [10].

Код написаний на JavaScript браузер інтерпретує напряму, тож немає потреби у компіляції, але для того аби сторінка завантажувалась і оброблювалась браузером швидше необхідно зменшити розмір скриптів і помістити їх в один файл, це допоможе зробити WebPack.

Webpack – це статичний будівельник модулів для сучасних JavaScript програм. Коли webpack обробляє програму, він внутрішньо будує граф залежності, який зображає кожен модуль, який потрібен проекту, і генерує один або кілька пакетів.

Завдяки тому, що Angular використовує WebPack за замовчуванням жодних дій для установки та налаштування проекту не знадобиться.

Через те, що при створенні стилів для додатка може виникнути необхідність у повторному використанні стилів, слід використати препроцесор для мови CSS, найрозповсюдженішим є SASS.

SASS – препроцесор мови CSS, що дозволяє використовувати такі інструменти як змінні, наслідування, розбиття на модулі з можливістю імпортування модулів, математичні вичеслення та функції.

Для спрощення стилізації конкретних елементів інтерфейсу, є сенс використати готову бібліотеку зі стилями, найпопулярнішим віріантом є Bootstrap, а завдяки тому, що ця бібліотека стилів використовує SASS як мову розробки, це збільшить рівень інтеграції з проєктом. Bootstrap спроєктовано для найкращої роботи в нових браузерах, тобто старі браузери можуть зображати стилі по-іншому, або ж можуть мати неповну функціональність при показі певного компонента.

Отже, клієнтський застосунок є важливою частиною хмарного сервісу, адже саме через нього клієнти сервісу і будуть отримувати всі послуги.

2.3 Організація передачі даних

Важливо розглянути особливості передачі даних між клієнтським застосунком і сервером, адже використання сервісу у медичних цілях вимагає уваги до аспектів безпеки передачі даних пацієнта.

Сервер передає клієнту веб додаток використовуючи HTTP запит, але він не захищає дані користувача, що є неприпустимо для будьякої хмарної системи що працює з медичними даними. Тому для хмарного сервісу має бути більш захищений спосіб передачі даних, як, наприклад, HTTPS протокол передачі даних. Протокол даних HTTPS – схема URI, що синтаксично ідентична HTTP схемі, яка зазвичай використовується для доступу до ресурсів в Інтернет. Використання “https: url ресурсу” вказує, що протокол HTTP має використовуватися, але з іншим портом за замовчуванням (443) і додатковим шаром шифрування між HTTP і TCP протоколами. Ця схема була винайдена у компанії “Netscape Communications Corporation” для забезпечення автентифікації та шифрування комунікацій і широко використовується в Інтернеті у програмному забезпеченні, в якому важлива безпека комунікацій, наприклад, у платіжних системах та системах документообігу. Саме тому використання цього протоколу є необхідною умовою для функціонування сервісу документообігу [11].

SSL (Secure Sockets Layer) та його наступник, TLS (Transport Layer Security), є протоколами для встановлення автентифікованих та зашифрованих зв’язків між мережевими комп’ютерами. Хоча протокол SSL був вичерпаний з випуском TLS 1.0 в 1999 році, дотепер прийнято називати ці пов’язані технології як «SSL» або «SSL / TLS». Найновіша версія - TLS 1.3, визначена у RFC 8446 (серпень 2018 року). SSL / TLS працює, прив’язуючи особистість таких організацій, як вебсайти та компанії, до криптографічних пар ключів за допомогою цифрових документів, відомих як сертифікати X.509. Кожна пара ключів складається з приватного ключа та відкритого ключа. Приватний ключ зберігається в безпеці, а відкритий ключ може

широко розповсюджуватися за допомогою сертифіката [11].

На самому базовому рівні, коли браузер потребує файл, розміщеного на вебсервері, браузер запитує файл через HTTP. Коли запит доходить до правильного вебсервера (апаратне забезпечення), сервер HTTP (програмне забезпечення) приймає запит, знаходить запитуваний документ (якщо він не відповідає, відповідь 404 повертається) і відправляє його назад у браузер, також через HTTP. Схема роботи сервера представлена на рисунку 2.2.

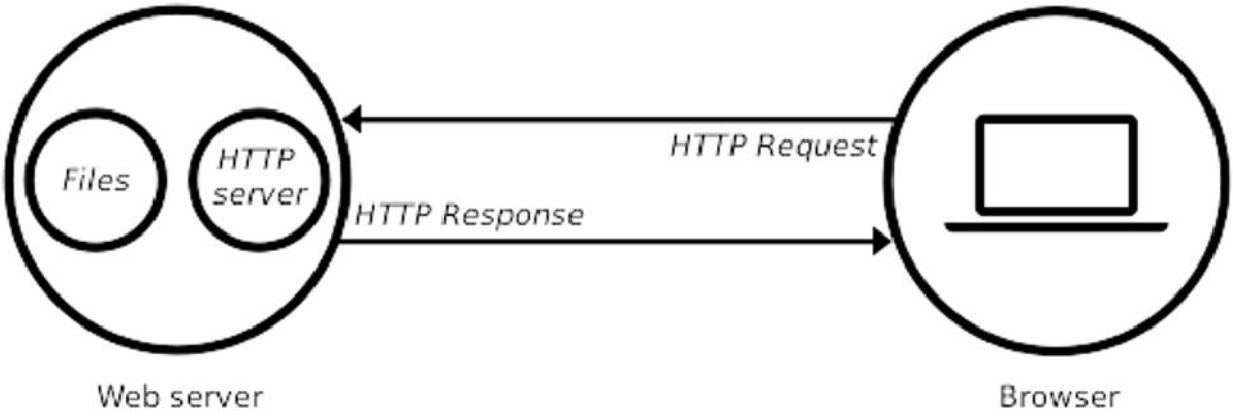


Рисунок 2.2 – Схема роботи веб сервера

Отже, було розглянуто яким чином сервер і клієнтський застосунок мають обмінюватись даними для забезпечення безпеки і приватності даних користувача.

2.4 Вибір мови програмування

Нажаль, через технічні особливості типізації мови JavaScript цю мову не дуже зручно використовувати у великих проектах, як на сервері, так і у браузерному додатку. Через динамічну природу JavaScript програміст може випадково зробити помилку, на виправлення якої піде велика кількість часу, неправильно обробивши данні, або випадково переписавши стандартні елементи мови програмування. Через це, набагато кращим варіантом буде

використати іншу мову програмування, яка після процесу компіляції перетвориться на JavaScript, однією з таких мов є TypeScript.

TypeScript – це мова з відкритим кодом, яка базується на JavaScript, одному з найбільш використовуваних інструментів у світі, додаючи статичні визначення типів. Типи – це спосіб описати форму об’єкта, надаючи кращу документацію та дозволяючи TypeScript перевірити правильність роботи вашого коду. Типи написання можуть бути необов’язковими для TypeScript, оскільки умовивід типу дозволяє отримати багато потужності без написання додаткового коду [12].

Код TypeScript перетворюється в код JavaScript за допомогою компілятора TypeScript або Babel. Цей JavaScript - чистий, простий код, який працює де завгодно JavaScript: у браузері, на Node.js . Саме задля більшої безпечності коду сервера і браузерного додатку варто використовувати саме TypeScript замість JavaScript [12].

Також, код написаний на TypeScript може бути скомпільований у різні версії JavaScript, що дуже важливо, адже не всі браузери підтримують останні версії JavaScript, тому використання TypeScript допоможе написати код у зручному для програміста вигляді і бути при цьому впевненим, що цей код буде працювати у будь-якому браузері.

Отже, було розглянуто мову програмування TypeScript, що дуже підходить для того аби бути в основі хмарного сервісу.

2.5 Середовище розробки

За середовище розробки було вирішено обрати текстовий редактор Visual Stidio Code, з можливістю встановлення розширень для роботи з різними мовами програмування.

Visual Studio Code – це легкий, але потужний редактор вихідного коду, який працює на робочому столі та доступний для Windows, macOS та Linux.

Visual Studio Code оснащений вбудованою підтримкою JavaScript,

TypeScript і Node.js і має багату екосистему розширень для інших мов (таких як C ++, C #, Java, Python, PHP, Go) та режимів виконання (таких як .NET і Unity).

Visual Studio Code це редактор коду, не IDE і має вбудовану підтримку режиму дебагу, підтримку роботи з системою контролю версій GIT і вбудований термінал для вводу команд. Редактор Visual Studio Code представлений на рисунку 2.3.

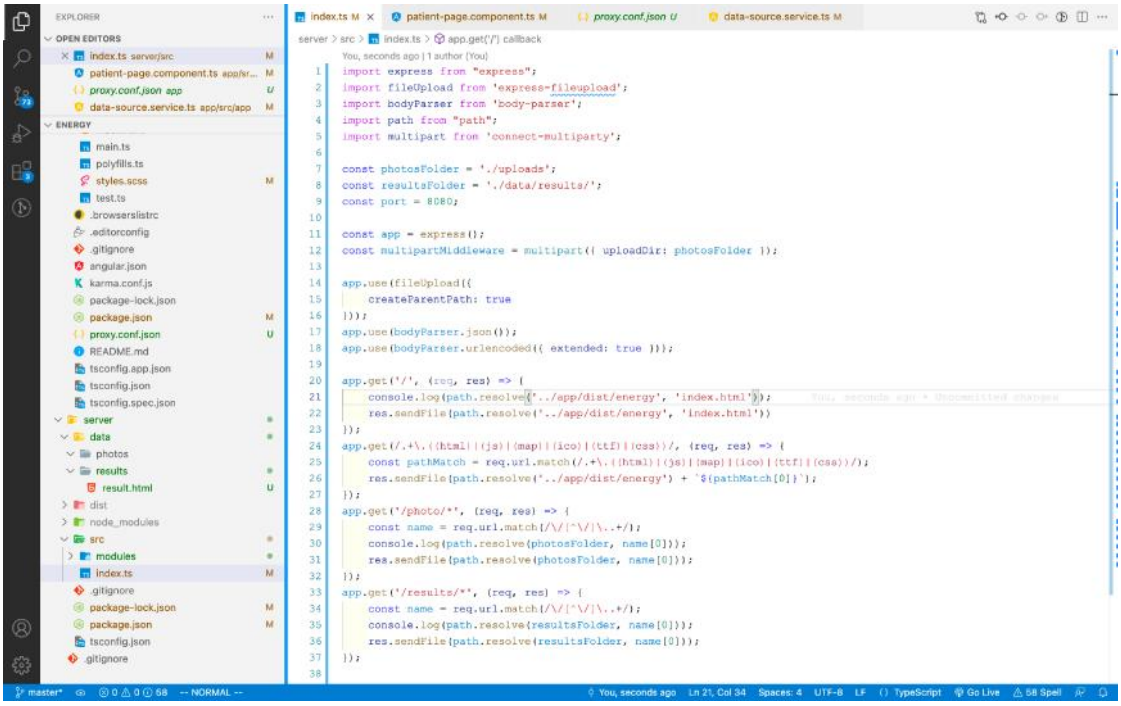


Рисунок 2.3 – Редактор коду Visual Studio Code

Отже, було розглянуто редактор коду, що буде використано для розробки сервісу.

2.6 Система контролю версій

Для організації коду і можливості керувати версіями сервусу необхідним є використання системи контролю версій.

Система контролю версій – це система, що записує зміни у файл або

набір файлів протягом деякого часу, так що ви зможете повернутися до певної версії пізніше.

Як приклад, систему контролю версій можна використовувати для коду програмного забезпечення, хоча насправді можна використовувати контроль версій практично для будь-яких типів файлів [13].

Багато людей в якості одного з методів контролю версій застосовують копіювання файлів в окрему директорію (можливо навіть директорію з відміткою за часом). Даний підхід є дуже поширеним завдяки його простоті, проте він, неймовірним чином, схильний до появи помилок. Можна легко забути в якій директорії ви знаходитеся і випадково змінити не той файл або скопіювати не ті файли, які ви хотіли.

Типи систем контролю версій:

- локальні системи контролю версій;
- централізовані системи контролю версій;
- децентралізовані системи контролю версій.

Схема роботи локальної системи контролю версій наведена на рисунку 2.4, централізованої на рисунку 2.5 і децентралізованої на рисунку 2.6.

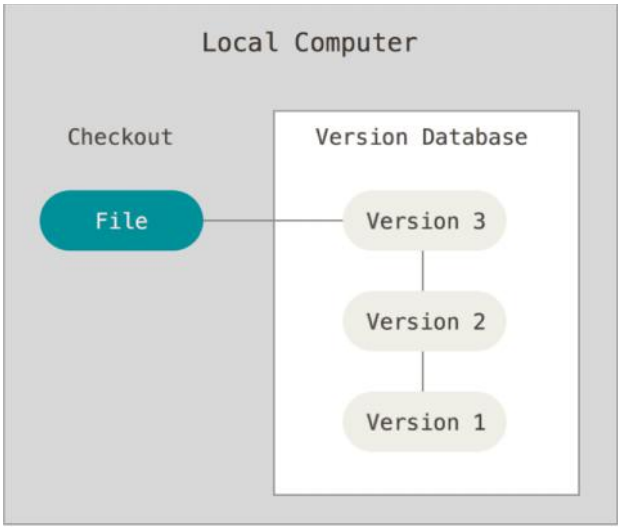


Рисунок 2.4 – Локальні системи контролю версій

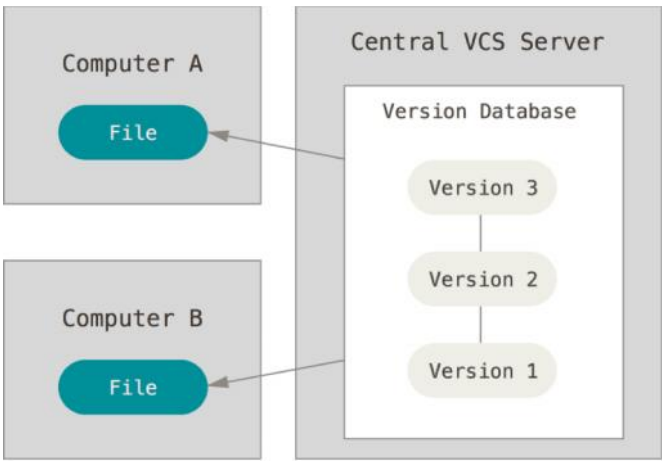


Рисунок 2.5 – Локальні системи контролю версій

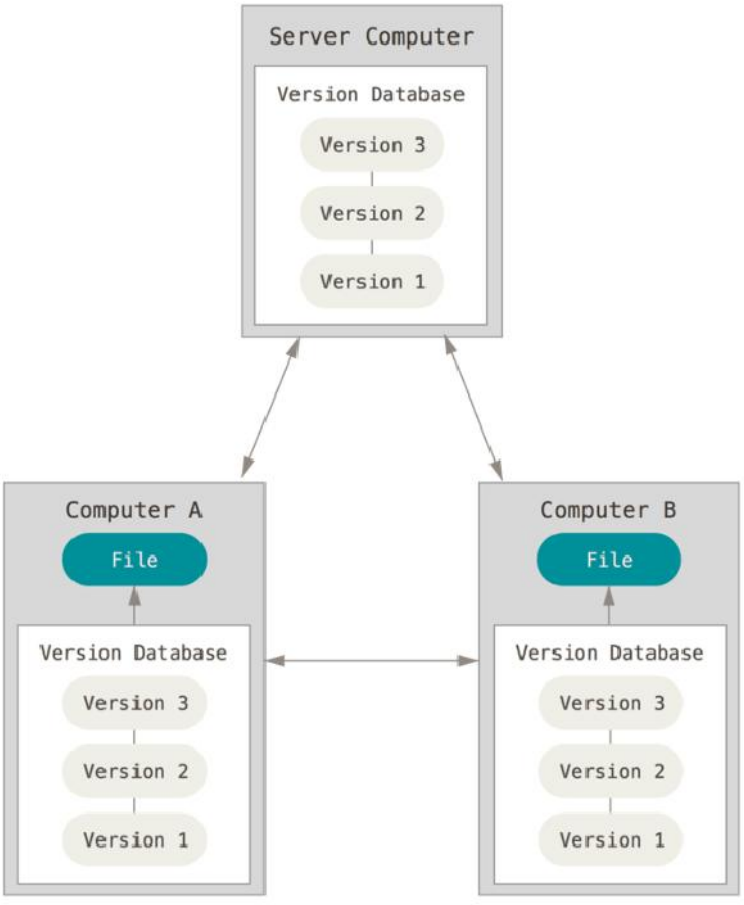


Рисунок 2.6 – Децентралізовані системи контролю версій

Для контролю зміни файлів у проєкті необхідна система контролю версій, найпопулярнішою є система контролю версій GIT, що є

децентралізованою системою контролю версій.

Але для того, аби над проєктом могли працювати декілька розробників, або для більшої безпечності (у разі поламки комп’ютера розробника) необхідно зберігати код на віддаленому сервері. Найкращим варіантом буде використати сервіс для хостингу GIT репозиторіїв, наприклад, GitHub.

GitHub – сервіс, що дозволяє зберегти GIT репозиторій проекту, аби декілька людей могли спільно працювати над проєктом. Представлений на рисунку 2.7.

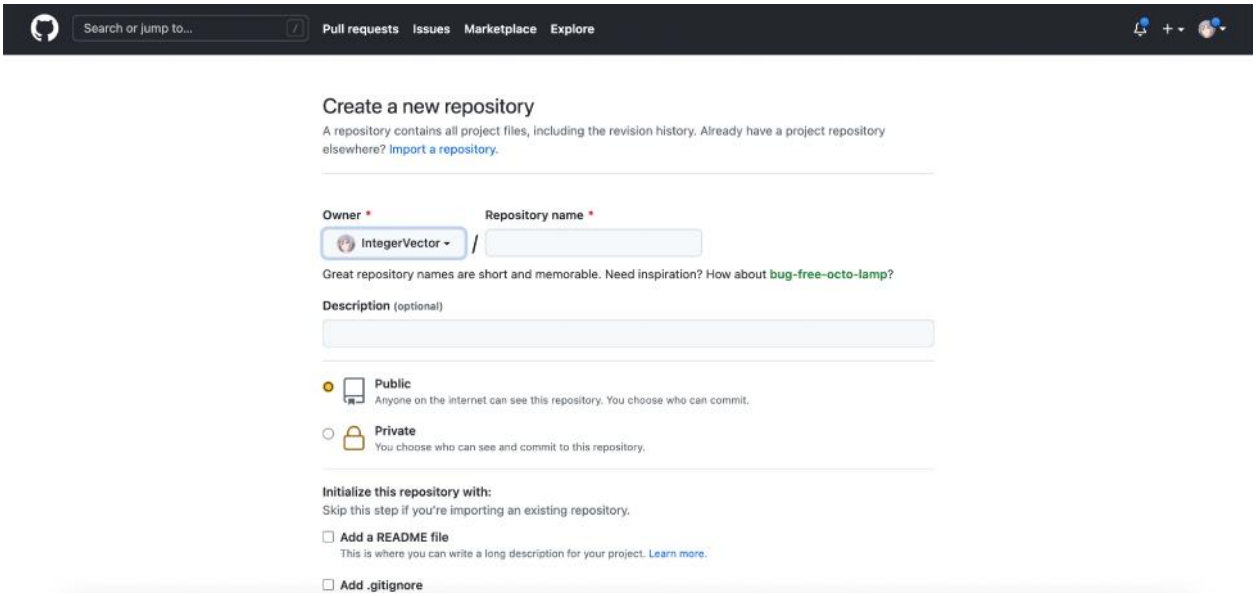


Рисунок 2.7 – Сторінка GitHub для створення нового репозиторію

Після створення нового репозиторію, у користувача сервісу є змога зробити його приватним і обмежити доступ тільки для розробників та власників проекту, додавати нові версії коду проекту, додавати файли документації та опису для проєкта.

Окрім розміщення коду, учасники можуть спілкуватись, коментувати редагування один одного, а також слідкувати за новинами знайомих.

За допомогою широких можливостей Git програмісти можуть поєднувати свої репозиторії – GitHub дає зручний інтерфейс для цього і може показувати вклад кожного учасника в вигляді дерева.

Для проєктів є особисті сторінки, невеликі Wiki та система відстеження помилок. Прямо на сайті можна дивитись файли проєктів з підсвічуванням синтаксису для більшості мов програмування.

На платних тарифних планах можна створювати приватні репозиторії, доступні обмеженому колу користувачів.

Є можливість прямого додавання нових файлів у свій репозиторій через веб інтерфейс сервісу

Код проєктів можна не лише скопіювати через Git, але й завантажити у вигляді архіву.

На сайті є pastebin-сервіс gist.github.com для швидкої публікації фрагментів коду.

Зберігання документації, включаючи автоматично відтворювані файли README у різних форматах файлів типу Markdown.

Також доступні:

- вкладені списки завдань у файлах;
- візуалізація геопросторових даних;
- 3D-рендеринг файлів, які можна попередньо переглянути, використовуючи новий інтегрований переглядач файлів STL, який зображає файли на «3D canvas». Переглядач підтримує WebGL і Three.js. Внутрішній формат PSD для Photoshop з можливістю попередньо перегляду та порівняння з попередніми версіями того самого файлу [13].

Для потреб проекту достатньо безкоштовних, базових можливостей сервісу.

Отже, було розглянуто технології що будуть використані для керування проектом і організації версій хмарного сервісу.

2.7 Особливість побудови хмарного сервісу для медичних цілей

В медичній сфері необхідно забезпечити безпеку приватних даних клієнтів, тож всі данні в базі даних мають бути зашифрованими, з’єднання

клієнта з сервером також. Шифроване з’єднання клієнта з сервером може бути забезпечене протоколом HTTPS, який вже був розглянутий в попередній частині.

Шифрування даних в базі даних може бути забезпечене на рівні бази даних, або на рівні сервера.

Перший варіант дозволяє отримати стабільний механізм шифрування з гарантіями безпеки від розробників системи контролю базою даних, а другий вимагає власної реалізації на рівні серверу [14].

Функціональність алгоритму AES може здатися складною, але в реальному сенсі її дуже просто зрозуміти. Перший крок - знати, що AES має 3 блочних шифри, а саме:

- AES-128, Цей блочний шифр використовує ключ довжиною 128 біт;
- AES-192, що використовує ключ довжиною 192 біти;
- AES-256, використовує ключ довжиною 256 біт.

Чим довший ключ шифрування, тим складніше зламати алгоритм. Тому AES-256 дуже стійкий до грубих атак, що намагаються підібрати правильне значення в порівнянні з двома попередніми варіантами.

Ключ, який використовується під час шифрування повідомлення, зазвичай є тим самим ключем, який використовується для розшифровки повідомлення на стороні-одержувача. Під час шифрування за допомогою AES користувач може використовувати AES-128, AES-192 і AES-256. Під час цього шифрування виконуються кілька кроків для шифрування повідомлення, і ці кроки виконуються поетапно [15].

На рисунку 2.8 схематично показано як AES шифрує дані.

Шифрування є ключовим у питаннях інформаційної безпеки медичних закладів.

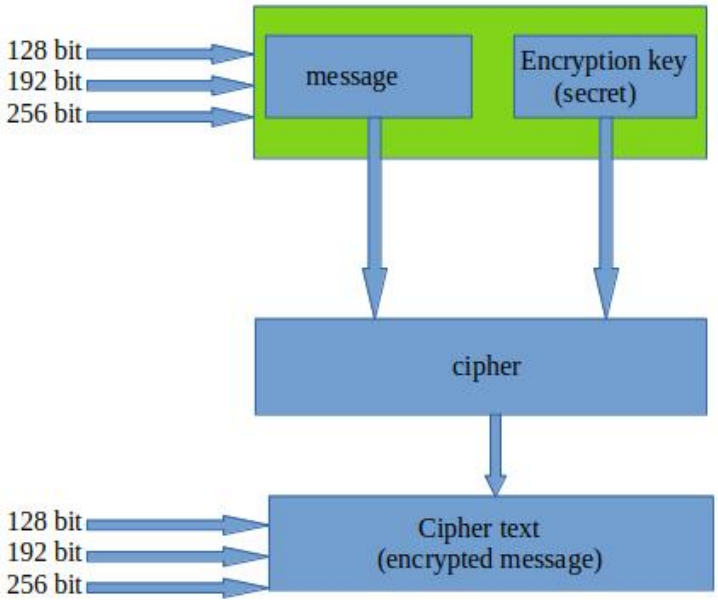


Рисунок 2.8 – Схема роботи AES

2.8 Висновки

В цій частині було розглянуто побудову простого хмарного сервісу, який можна було б використати з метою покращення аналізу енергограм пацієнтів. Оглянуто набір технологій, які можна використати для створення хмарного сервісу.

Для хмарного сервісу буде використаний сервер на базі node.js і фреймворку Express, які підходять завдяки своїй швидкості і дешевизни в розробці. СКБД MySQL для побудови бази даних, через широкий набір інструментів і підтримку у всіх серверах. А також, Angular в якості фреймворку для веб застосунку, через його інструменти підтримки різних браузерів і велику кількість вбудованих функцій. Для розробки як сервера, там і веб застосунку буде використана мова TypeScript, що покращить стабільність сервісу у перспективі.

3 РОЗРОБКА АРХІТЕКТУРИ ХМАРНОГО СЕРВІСУ ОБ’ЄКТУ РОЗРОБКИ

3.1 Схема роботи сервісу

Перш ніж почати розробку архітектури, хорошою практикою є побудувати узагальнюючу схему роботи сервісу.

Для створення хмарного сервісу для медичних і відпочивальних закладів для обробки зображень енергограм пальців рук пацієнта.

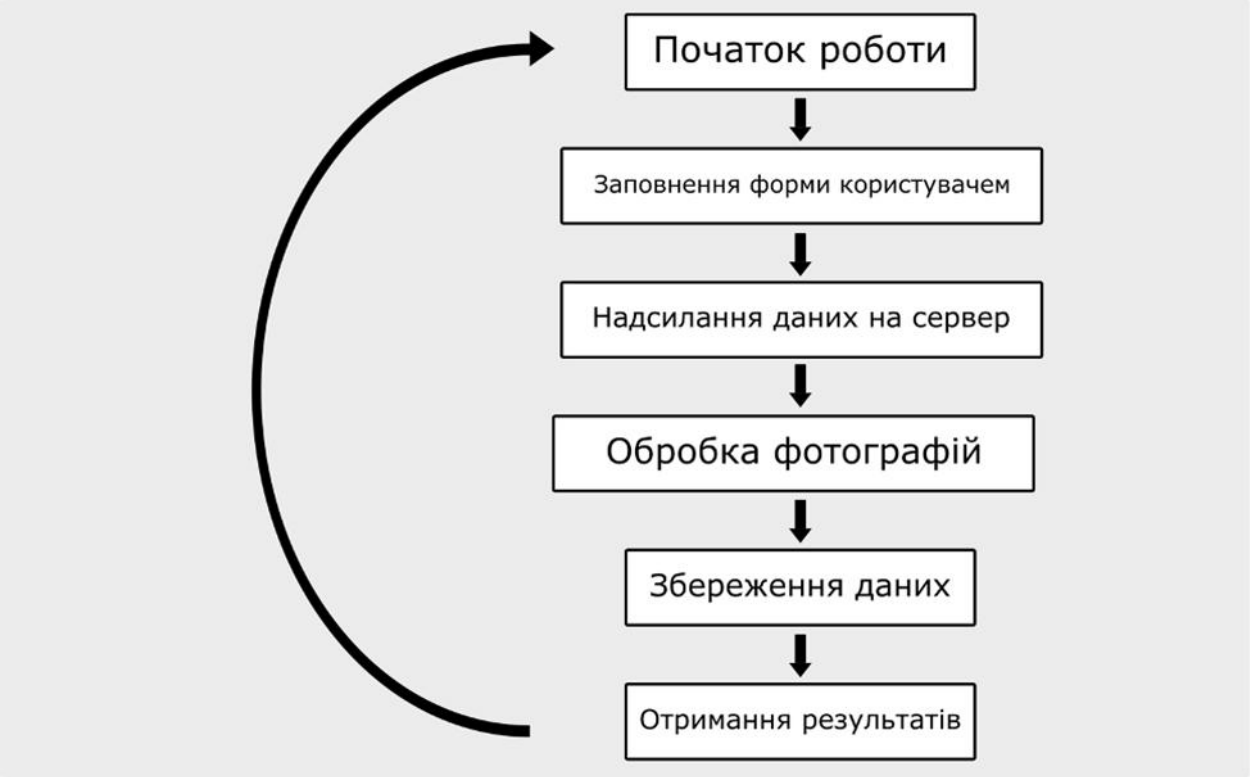


Рисунок 3.1 – Загальна схема роботи сервісу

На рисунку 3.1 зображено узагальнену схему роботи сервісу. На зображені можна помітити що процес роботи достатньо простий і прямий.

Користувач починає роботу з сервісом починаючи з мобільного веб застосунку. Після запуску мобільного веб застосунку користувач має завантажити фотографії зображень енергетичного поля всіх пальців рук.

Після завантаження зображень, користувач має ввести дані пацієнта для того, аби результати було коректно збережено, після чого сервіс починає обробку зображень і повертає результат.

Краще за все буде повернути результат в графічній формі, наведеній в попереднії частинах, але достатньо буде і текстового результату.

Отже, була розглянута загальна схема роботи хмарного сервісу, з всіма етапами роботи користувача з сервісом.

3.2 Вибір інфраструктури сервісу

Для даного сервісу не має сенсу розгортати власну структуру комп’ютерів, достатньо скористатись вже готовими рішеннями. Немає значення який сервіс буде обраний, головними критеріями вибору залишиться надійність і поєднання ціни і якості обслуговування.

Кожен медичний заклад, чи заклад відпочинку може сам обрати необхідного провайдера послуг. А враховуючи, що деякі заклади вже мають власні веб сторінки чи навіть сервери, вибір провайдера серверної інфраструктури може бути вже вирішеним питанням для таких закладів.

Отже, для хмарного сервісу достатньо орендувати готову до використання інфраструктуру. Вибір конкретної залежить від побажань і від фінансових можливостей кожного конкретного медичного закладу, іабо закладу відпочинку.

3.3 Структура бази даних

Структура бази даних сильно залежить від конкретних потреб медичного закладу, чи закладу відпочинку. В рамках цієї роботи буде запропонована проста структура бази даних, що дозволить зберігати базові дані пацієнта і результати його діагностики.

Для цього, необхідна наявність хочаб трьох таблиць.

Перша таблиця «Episodes» необхідна для збереження контактних і базових даних пацієнта. Кожен епізод, це кожен випадок проведення діагностики. Епізод містить дані пацієнта і дані про сам епізод, наприклад, дату проведення діагностики. У випадку конкретних медичних закладів така таблиця може бути розширена і містити додаткові дані пацієнта.

Друга таблиця «EnergyGraphs» необхідна для прив’язки збережених зображень енергетичног поля до конкретного пацієнта і його епізода. Таблиця містить шляхи до всіх зображень пальців пацієнта.

Третя таблиця «Results» містить шлях до файлу з результатами діагностики, а також слугує таблицею для поєднання даних епізоду з файлами зображень енергетичного поля пальців рук пацієнта.

Схема цієї бази даних наведена на рисунку 3.2.

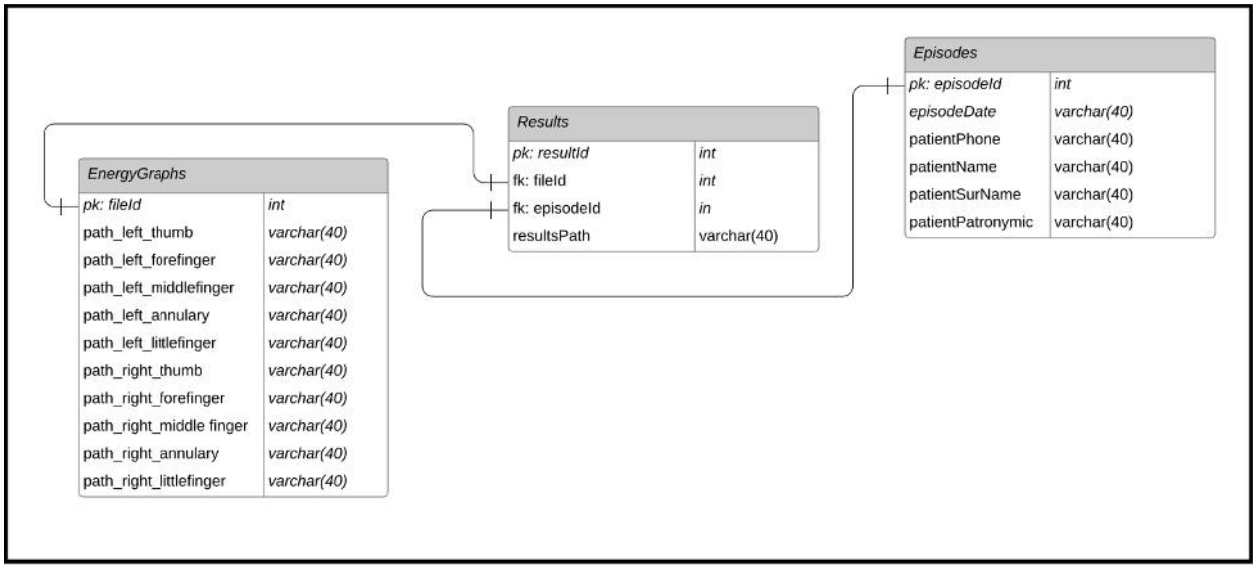


Рисунок 3.2 – Схема бази даних сервісу

Отже, була створена схема бази даних для хмарного сервісу для аналізу зображень енергограм пальців пацієнта медичного закладу, чи закладку відпочинку.

3.4 Програма діагностики

Для проведення діагностики за зображеннями енергетичного поля необхідно запустити програму аналізу.

Програма аналізу зображення є окремою програмою, а не частиною сервера, але саме сервер і буде запускати її.

Для цього необхідно написати програму-обгортку, що дозволить передавати дані з сервера в програму і у зворотньому напрямку. Далі, ця програма-обгортка має стати частиною модулів сервера, аби сервер міг напряму комунікувати з програмою обгорткою. Після цього сервер зможе викликати програму діагностики і отримувати її результати.

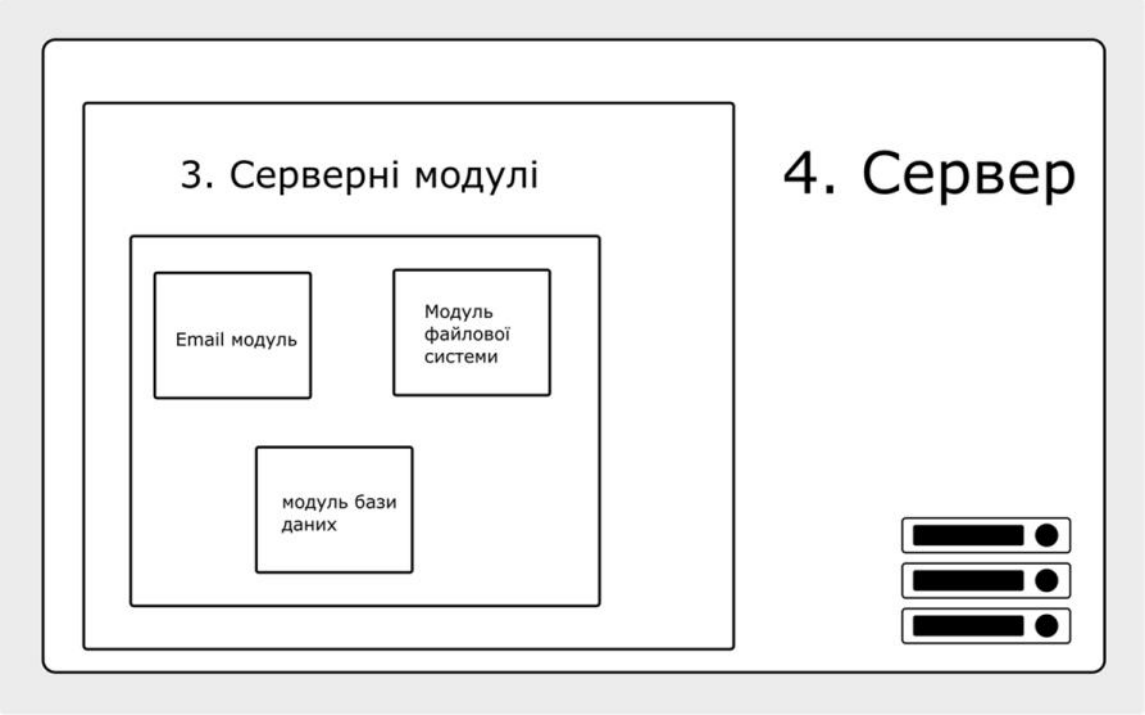


Рисунок 3.3 – Схема зв’язку сервера з програмою діагностики

Схема зв’язків сервера і програми діагностики наведена на рисунку 3.3.

Для створення програми обгортки підійде пакет «node-gyp» який дозволить автоматично зібрати пакет програми обгортки і перетворити його на модуль для сервера.

Версія утиліти node-gyp входить у комплект і поширюється з Node.js як частина npm. Ця версія не доступна для розробників і призначена лише для підтримки можливості використання команди npm install для компіляції та встановлення доповнень. Розробники, які бажають використовувати node-gyp, можуть безпосередньо встановити його за допомогою команди «npm install -g node-gyp».

Встановлення цього інструменту може зайняти час, але це один з не багатьох способів виконати необхідну нам програму діагностики.

Отже, програма діагностики буде запускатись за допомогою серверу, через програму обгортку.

3.5 Опис приватного API

Для передачі даних між браузерним веб застосунком і сервером необхідно створити інтерфейс програмної розробки (application programing interface, далі буде скорочено до API).

API є описом того як сервер може комунікувати з браузерним веб застосунком.

Для потреб даного сервісу достатньо дуже простого API, але кожен конкретний заклад оздоровлення чи відпочинку може розширити його для своїх власних потреб.

В цьому випадку буде достатньо трьох запитів, доступних для використання застосунком. Запити наведені на рисунку 3.4.

На рисунку 3.4 можна побачити розроблене API.

Браузерний веб застосунок може наліслати на сервер POST запит з зображнням енергетичного поля пальців рук людини, а у відповідь сервер надішле шлях, який дозволить завантажити це зображення в застосунку. Це потрібно для того, аби користувач міг побачити завантажене зображення і переконатись що все працює.

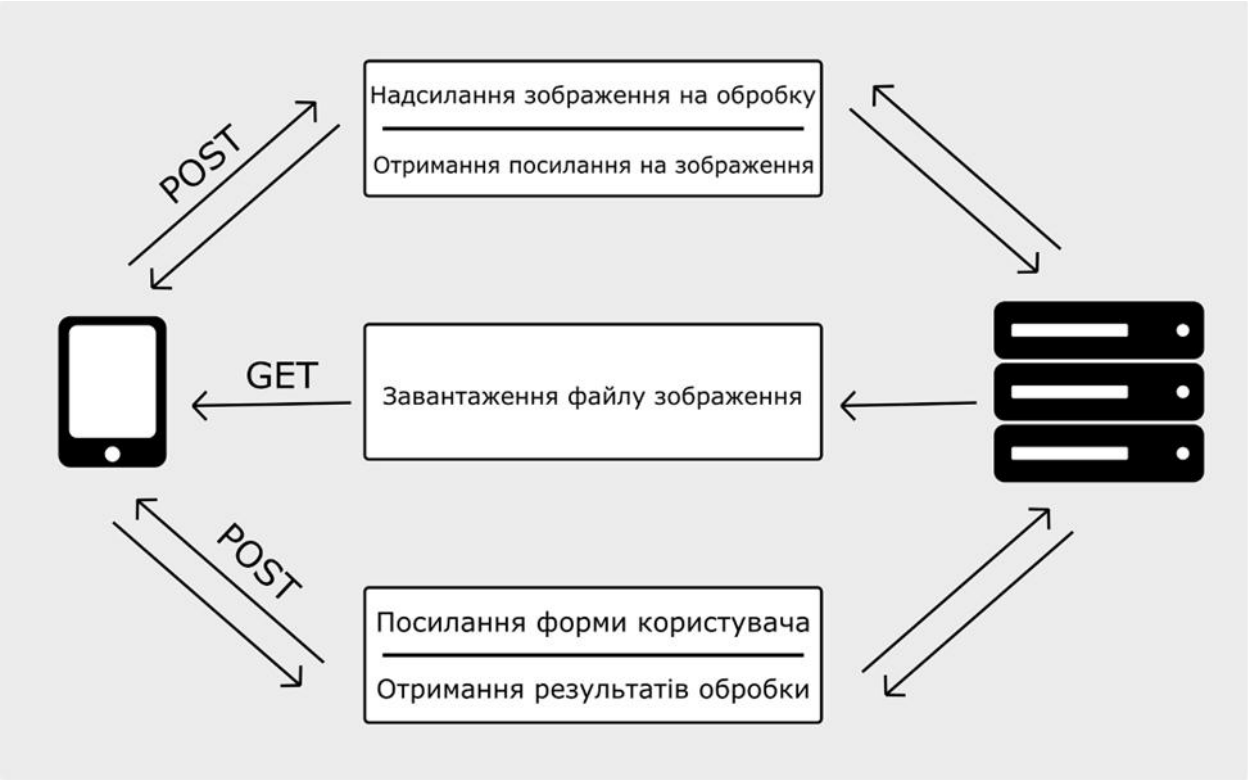


Рисунок 3.4 – API сервіса

Далі, GET запит застосунок може використати отриманий шлях для завантаження файлу зображення для перегляду користувачем.

Останній POST запит в нижній частині рисунка дозволяє надіслати дані пацієнта, у відповідь на які сервер запустить процес аналізу даних пацієнта і надішле результат.

Дане API є простим і надійним. Для забезпечення безпеки дані мають передаватись в зашифрованому вигляді, а саме API має бути приватним, тобто доступним лише через мобільний веб застосунок. Для того, аби зробити API приватним, немає необхідності в жодних додаткових діях, адже за замовчуванням сервер не додає дозвіл на запити з інших доменних імен і браузер блокує такі запити.

Отже, було розроблено API для комунікації серверу з мобільним застосунком, що дозволяє завантажувати файли зображень енергограм пацієнтів на сервер і отримувати результати аналізу.

3.6 Архітектура веб застосунку

Мобільний веб застосунок має бути простим і швидким. Для цього його не варто перевантажувати зайвими елементами інтерфейсу і всі операції в застосунку мають бкти простими і очевидними для користувача.

Схема роботи застосунку наведена на рисунку 3.5.

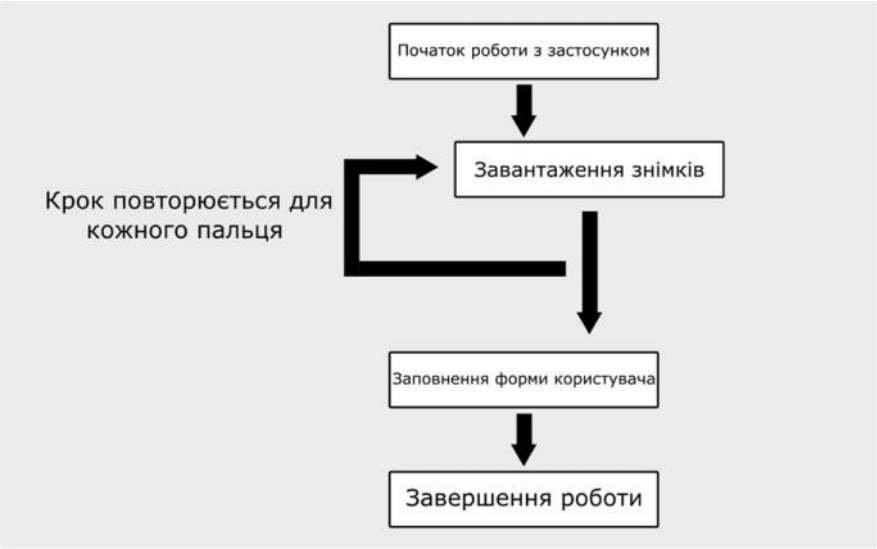


Рисунок 3.5 – Схема роботи застосунку

На початку роботи користувач має завантажити фотографії енергетичних полів всіх пальців рук пацієнта. Після чого застосунок має запропонувати користувачу ввести додаткові дані пацієнта для того, щоб почати обробку. Після отримання результатів користувач завершує свою роботу.

Для поавног переходу між етапами роботи сервісу він має бути розбитим на сторінки.

Для кожного етапу має бути створена своя сторінка в мобільному застосунку.

Це необхідно для того, аби користувач міг на кожному етапі повернутись назад і замінити вказані дані.

Така архітектура також дозволяє пришвидшити розробку застосунку і зменшити потенційні помилки при оновленні інтерфейсу застосунку.

Користувач має змогу додати тільки необхідні файли зображень енергограм пальців рук пацієнта.

Отже, була розроблена архітектура мобільного застосунку для завантаження зображень енергограм пацієнта на сервер і отримання результатів обробки даних пацієнта.

3.7. Можливості покращення сервісу

Наведена архітектура сервісу є простим прикладом того, як може працювати сервіс. В реальних задачах може знадобитись розширити дану архітектуру сервісу.

Варіантами покращення бази даних може бути додання шифрування даних в базі даних. Це дозволить забезпечити приватність пацієнтів у випадку втрати даних.

Сервіс також може не тільки збирати дані пацієнта, а і може бути розширеним функцією аналізу цих даних. Наприклад, можна аналізувати частоту виникнення тих чи інших проблем зі здоров’ям пацієнтів, або надавати інформацію про зміну прогресії звороби з часом.

3.8 Висновки

В цій частині було побудовано схему роботи і побудови всіх компонентів хмарного сервісу. Архітектура майбутнього сервісу буде складатись з БД, серверу для обробки даних, програми аналізу енергограм і веб застосунку, схема взаємозв’язку між якими була наведена. Дана схема є звичною для хмарних сервісів, окрім частини з програмою для аналізу. Дана архітектруа вимагає наявності операторa для проведення аналізу і здатна сильно пришвидшити процес аналізу даних клієнта.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ХМАРНОГО СЕРВІСУ ОБ’ЄКТУ РОЗРОБКИ

Для реалізації сервісу буде використано інструменти описані в попередній частині. В даній частині увага буде приділена найважливішим моментам в розробці серверу і мобільного веб додатку.

4.1 Файлова структура

Для того, щоб весь сервіс був в одному місці варто тримати проект в одній папці.

Для файлів зображень енергограм пальців рук пацієнта необхідно створити окрему папку, так само, як і для файлів з результатами аналізів пацієнтів.

Клієнтський додаток має бути в своїй підпапці в середині папки хмарного сервісу. З цієї підпапки буде доступен весь клієнтський застосунок для надсилання через сервер.

Отже, файлова структура проекту має залишатись простою.

4.2 Налаштування проекту

Для клієнтського додатку і серверу необхідно створити файли налаштувань, які будуть налаштовувати збірку проекту і його особливості.

Перший файл необхідний для налаштування поведінки angular проекту, генерується фреймворком автоматично і є обовєязковим у проекті. Цей файл налаштовує поведінку компілятора, поведінку мови програмування, директорії для збереження результатів компіляції і набір команд для виконання через інструменти фреймворку. Файл налаштувань наведено в прикладі на лістингу 4.1.

Лістинг 4.1 – Фрагмент файлу налаштувань angular

```
"build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/energy",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.app.json",
            "inlineStyleLanguage": "scss",
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "development": {
            }
          },
          "defaultConfiguration": "production"
        },
        "serve": {
          "builder": "@angular-devkit/build-angular:dev-server",
          "configurations": {
            "production": {
              "browserTarget": "energy:build:production"
            },
            "development": {
              "browserTarget": "energy:build:development"
            }
          },
          "defaultConfiguration": "development"
        },
        "extract-i18n": {
          "builder": "@angular-devkit/build-angular:extract-
i18n",
          "options": {
            "browserTarget": "energy:build"
          }
        },
        "test": {
          "builder": "@angular-devkit/build-angular:karma"
          }
        }
```

Для браузерного застосунку має бути створено окремий файл налаштувань, для налаштування доступного набору команд для виконання.

Файл налаштувань node.js середовища для браузерного застосунку наведено в прикладі лістингу 4.2.

Лістинг 4.2 – Фрагмент файлу налаштувань застосунку

```
{
  "name": "energy",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "watch": "ng build --watch --configuration development"
```

Файл налаштувань node.js середовища для сервера наведено в прикладі лістингу 4.3.

Лістинг 4.3 – Файл налаштувань серверу

```
{
  "main": "dist/index.js",
  "scripts": {
    "build": "tsc",
    "start": "node ."
  },
  "dependencies": {
    "body-parser": "^1.19.0",
    "express": "^4.17.1",
    "express-fileupload": "^1.2.1"
  },
  "devDependencies": {
    "@types/express": "^4.17.13",
    "@types/express-fileupload": "^1.2.0",
    "@types/node": "^16.11.10",
    "typescript": "^4.5.2"
  }
}
```

Оскільки для сервера використовується мова TypeScript, так само як і для клієнтського застосунку, то необхідно створити файл налаштувань для цієї мови. Файл буде вказувати опції для компілятора, поведінку з типами та парку для вихідних файлів.

Файл налаштувань наведено у прикладі 4.4.

Лістинг 4.4 – Файл налаштувань мови програмування

```
{
    "compilerOptions": {
        "module": "commonjs",
        "esModuleInterop": true,
        "target": "es6",
        "noImplicitAny": true,
        "moduleResolution": "node",
        "sourceMap": true,
        "outDir": "dist",
        "baseUrl": ".",
        "paths": {
            "*": [
                "node_modules/*"
            ]
        }
    },
    "include": [
        "src/**/*"
    ]
}
```

Отже, проекти серверу і клієнтського застосунку були налаштовані. Були налаштовані команди для середовища node.js, поведінка компілятора. Після налаштування стає можливо розпочати саму розробку компонентів хмарного сервісу.

4.3 Розробка серверу

Для серверу необхідно лише реалізувати три запити з розробленого раніше API.

Але перед цім сервер має бути налаштований рацювати з файлами зображень і даними що будуть надсилатись в форматі json. Для цього необхідно вказати серверу на необхідність використовувати додаткові інструменти. Інструменти організовані у вигляді об’єктів стратегій, і можуть бути легко додані до функціоналу серверу.

Код реалізуючий налаштування сервера наведено у прикладі 4.5.

Лістинг 4.5 – Налаштування сервера

```
import express from "express";
import fileUpload from ’express-fileupload’;
import bodyParser from ’body-parser’;
import path from "path";

const photosFolder = ’./data/photos/’;
const resultsFolder = ’./data/results/’;
const port = 8080;

const app = express();

app.use(fileUpload({
    createParentPath: true
}));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
```

Для того, аби клієнтський додаток міг отримати зображення енергограм пацієнта, необхідно реалізувати відповідний елемент API. Код реалізації наведено в прикладі 4.6.

Лістинг 4.6 – Реалізація отримання зображення

```
app.get(’/photo/*’, (req, res) => {
    const name = req.url.match(/\/[^\/]\..+/);
    res.sendFile(path.resolve(photosFolder, name[0]));
});
```

Для того, аби застосунок міг надіслати зображення енергограм пацієнта на обробку, необхідно реалізувати відповідну частину API. Реалізація цього наведена у прикладі 4.7.

Лістинг 4.7 – Завантаження енергограм на сервер

```
app.post(’/photo’, (req, res) => {
    const data = JSON.parse(req.body);
    const finger = req.files.finger;
    const fileName =
‘${data.type}_${Math.round(Math.random()*10000)}‘;
```

```
    (<any>finger).mv(photosFolder + fileName);

    res.send({
        status: true,
        data: {
            type: data.type,
            name: fileName
        }
    });
});
```

4.4 Розробка застосунку

Для того, аби у користувача була можливість завантажити зображення енергограм, необхідно додати відповідний функціонал. Але для того, щоб не змішувати код варто розподілити функціонал застосунку на окремі сторінки.

Кожна з сторінок має приблизно однаковий функціонал, тож немає сенсу копіювати один і той самий код. Достатньо створити одну базову сторінку і створити ієрархію з наслідування кожною сторінкою базової і доповненням своїм функціоналом. Код базової сторінки наведено у прикладі 4.8.

Лістинг 4.8 – Код компонента базової сторінки

```
@Component({
  selector: 'app-base-finger-page',
  templateUrl: './base-finger-page.component.html',
  styleUrls: ['./base-finger-page.component.scss']
})
export class BaseFingerPageComponent {
  public title = '';
  public nextPageUrl = '';
  public previewImg = '';

  protected fingerType = '';

  private fileToUpload: File | null = null;

  @ViewChild('fileInput')
  private fileInput!: ElementRef;

  constructor(protected router: Router) { }
```

```
  public next(): void {
    this.router.navigate([‘/${this.nextPageUrl}‘]);
  }
  handleFileInput(target: any) {
    const files: FileList = target.files;
    if (files) {
      this.fileToUpload = files.item(0);
    }
  }
  public fileSelection(): void {
    console.log(this.fileInput);
    this.fileInput.nativeElement.trigger(’click’);
  }
```

Стилі кожної сторінки ідентичні, тому немає сенсу створювати окремі файли реалізації стилів, можна створити всі css стилі в одному місці. Код реалізації css стилів наведено в прикладі 4.9.

Лістинг 4.9 – Стилі для сторінок

```
html, body {
    margin: 0;
    padding: 0;
    height: 100%;
    width: 100%;
    font-family: Arial, Helvetica, sans-serif;
    font-size: x-large;
    color: #464545;
}
h1 {
    font-size: xx-large;
    font-weight: bold;
}
input {
    min-width: 6rem;
    height: 1rem;
    border: 2px solid #7a7a7a;
    border-radius: 0.2rem;
}
```

Реалізація елементів сторінки теж може бути зроблена лише в одному файлі для більшості сторінок. HTML реалізація базової сторінки наведена у прикладі 4.10.

Лістинг 4.10 – Код розмітки базової сторінки

```
<div class="page-container">
    <h1>{{ title }}</h1>
    <img src="{{ previewImg }}">
    <div class="command-bar">
        <button
            (click)="fileSelection()">
            Обрати
        </button>
        <button
            [disabled]="!!previewImg"
            (click)="next()">
            Наступне
        </button>
    </div>
    <form action="">
        <input
            #fileInput
            name="imageSelector"
            type="file"
            accept=".jpg, .jpeg, .png, .bmp"
            (change)="handleFileInput($event.target)"
            style=""/>
    </form>
</div>
```

Але деякі сторінки відрізнятимутся від базової, тож яхня розмітка має бути зроблена в окремих файлах. Приклад реалізації сторінки пацієнта наведено в прикладі 4.11.

Лістинг 4.11 – Код розмітки сторінки даних пацієнта

```
<div class="patient-container">
    <h1>Дані пацієнта</h1>
    <div class="row">
        <label for="name">Ім’я</label>
        <input name="name" type="text">
    </div>
    <div class="row">
        <label for="surname">Прізвище</label>
        <input name="surname" type="text">
    </div>
    <div class="row">
        <label for="patr">Побатькові</label>
        <input name="patr" type="text">
```

```
    </div>
    <div class="row">
        <label for="phone">Номер телефону</label>
        <input name="phone" type="tel">
    </div>
    <div class="command-container">
        <button>Надіслати на аналіз</button>
    </div>
</div>
```

Отже, було розроблено веб застосунок для надсилання файлів зображень енергограм пацієнтів для обробки на сервер.

4.5 Висновки

В цій частині було розроблено сервер і веб застосунок для проведення діагностики пацієнтів медичних закладів і закладів відпочинку, шляхом аналізу енергограм пальців рук пацієнта.

Розроблений хмарний сервіс допоможе пришвидшити процес аналізу знімків енергограм [16]. Були наведені фрагменти коду, що ілюструють реалізацію того, чи іншого функціоналу сервісу.

Дана реалізація

Були наведені фрагменти коду, що ілюструють реалізацію того, чи іншого функціоналу сервісу.

5 ІНСТРУКЦІЯ КОРИСТУВАЧА

На початку роботи з застосунком користувача очікує сторінка привітання, зображена на рисунку 5.1.

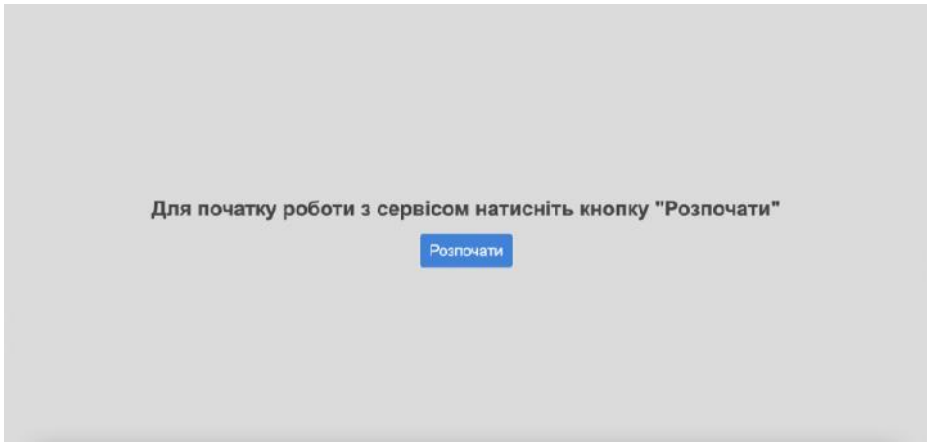


Рисунок 5.1 – Комп’ютерна сторінка привітання

Після натискання кнопки «Розпочати» користувач може обрати файл зображення енергограми для аналізу.

Сторінка завантаження зображення зображена на рисунку 5.2.

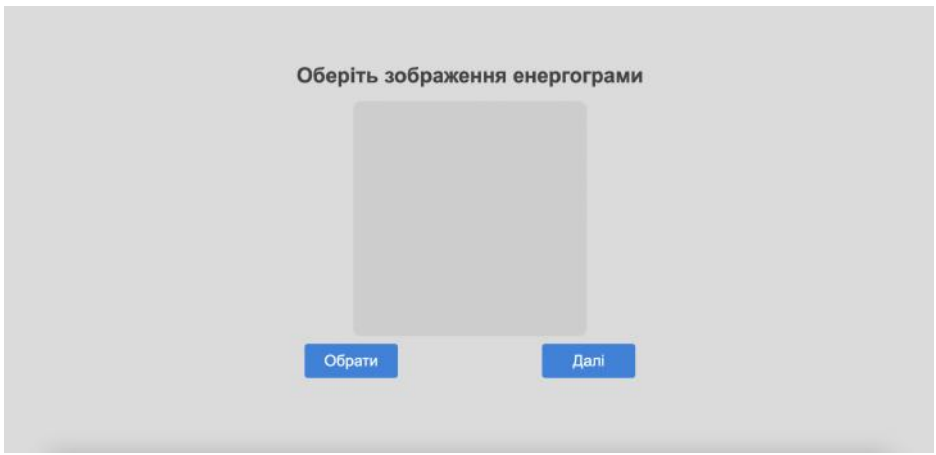


Рисунок 5.2 – Сторінка завантаження файлу

Для завантаження зображення енергограми необхідно натиснути кнопку «Обрати». Вікно вибору файла наведено на рисунку 5.3.

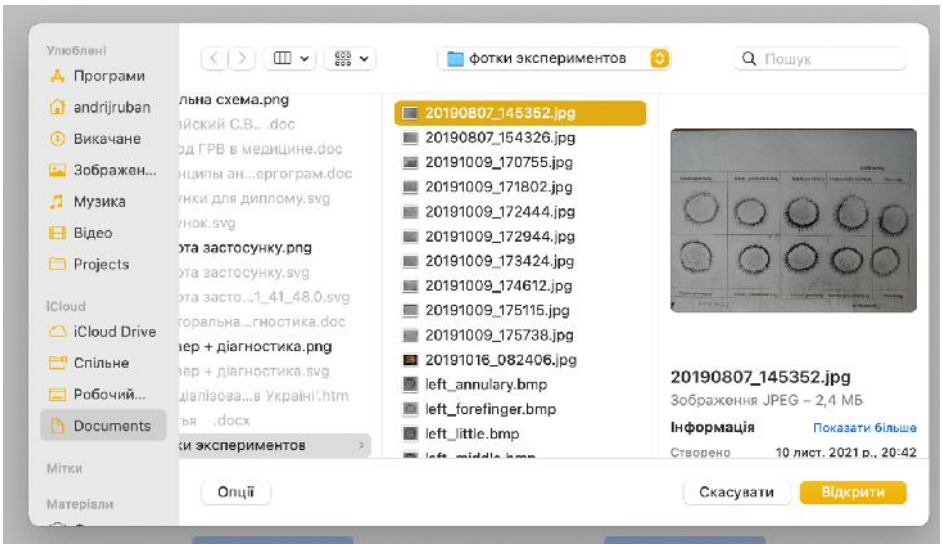


Рисунок 5.3 – Вікно вибору файла

Після того, як файл завантажено, можна побачити його на сторінці. Для продовження необхідно натиснути кнопку «Наступне».

Остання сторінка, є сторінкою заповнення даних пацієнта. Стоінка даних пацієнта наведена на рисунку 5.4.

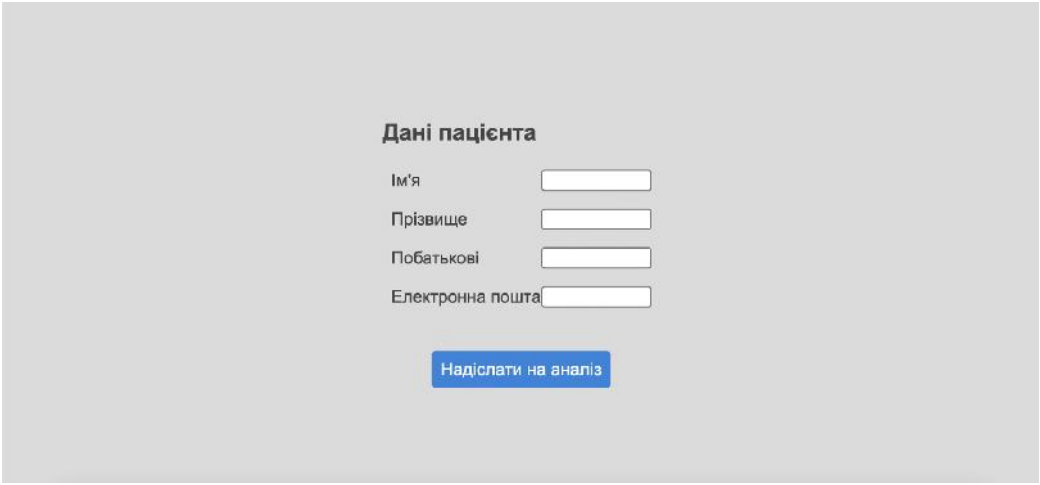


Рисунок 5.4 – Сторінка даних пацієнта на комп’ютері

Після натискання кнопки «Надіслати на аналіз» залишається лише чекати на лист з результатами, що буде надіслано на електронну пошту. Формат представлення даних наведено на рисунку 5.5.

Рисунок 5.5 – Представлення результатів

Лист прийде через деякий час, адже оператору потрібен час на проведення аналізу. На цьому робота сервісу завершується.

ВИСНОВКИ

В результаті роботи було проаналізовано два існуючих методи оцінки стану пацієнта, що відносяться до енергетичної діагностики і базуються на аналізі зображень енергограм пацієнта. Для автоматизації був обраний метод аналізу на основі поляризації дієлектриків, через його дешевизну. На основі обраного методу було побудовано хмарний сервіс для автоматизації оцінки стану пацієнтів, що дозволяє пришвидшити процес аналізу енергограм.

В роботі було проаналізовано з чого складається типовий хмарний сервіс і побудовано архітектуру сервісу для автоматизації обраного методу енергетичної діагностики. Було створено сервіс на основі розробленої архітектури і описано інструкцію користувача.

Розроблений сервіс може бути покращений, шляхом представлення результатів діагностування у графічному, а не текстовому форматі.

Практичне значення отриманих результатів полягає в пришвидшенні існуючого методу оцінки стану пацієнта, методу, що задяки своїй дешевизні і швидкості може сильно розвантажити існуючі оздоровчі і відпочивальні заклади і дозволити не професійним людям проводити діагностику.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Панфілова Г. Л., Панфилова А. Л., Бобошко Л. Г., Гусаченко А. А. Аналіз епідеміологічної ситуації по COVID-19 та відповідних протипандемічних заходів в Україні : матеріали міжнар. наук.-практ. конф., м. Харків, 26-27 лист. 2020 р. Київ, 2020. С. 187.

2. Попович І.Л., Бабелюк В.Є., Дубкова Г.І. Зв’язки між параметрами біоелектрографії (кірліанографії) та варіабільності серцевого ритму і артеріального тиску / І.Л. Попович, В.Є. Бабелюк, Г.І. Дубкова // Медична гідрологія та реабілітація. – 2010. – Т. 8, № 1. – С. 4-16.

3. Дроздов Д.А., Шацилло О.И. анализ ГРВ - биоэлектрографических изображений с позиций вегетологии. Труды международной конференции «Наука, Информация, Сознание», СПб, 2005, С. 99-104

4. Коротков К. Г. Основы ГРВ биоэлектрографии / К. Г. Коротков. – 2001. – 356 с.

5. Бистрова А. Метод ГРВ в медицині / А. Бистрова..

6. Спосіб аналізу стану біологічних та технічних об’єктів на основі явища роляризації дієлектриків : пат. 116491 Україна : G01N 27/62 / Завизіступ Ю.Ю., Свиридов А.С., Макаревич В.С. – № a201607179 ; заявл. 02.07.2016 ; опубл. 26.03.2018, Бюл. № 6/2018. – 3 с.

7. Пісоцька Л. А., Лелека О. В., Науменко О. Ю., Ковальчук Л. В., Баранова Н. О. Кірліан-діагностика в практичній охороні здоров’я // Вісник проблем біології і медицини. 2014. №3. URL: https://cyberleninka.ru/article/n/kirlian-diagnostika-v-praktichniy-ohoroni-zdorov-ya.

8. Волченко О. А. Методі і моделі класифікації зображень кільцевої форми / О. А. Волченко..

9. Tanenbaum A. S. Computer networks / A. S. Tanenbaum, D. J. Wetherall. – Boston: Pearson Education, Inc., 2011. – 905 с.

10. Introduction to the Angular Docs [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/docs.

11. MDN Web Docs [Електронний ресурс] // Mozilla and individual contributors. – 2005. – Режим доступу до ресурсу: https://developer. mozilla.org/en-US/.

12. Documentation TypeScript [Електронний ресурс] // Microsoft. – 2012. – Режим доступу до ресурсу: https://www.typescriptlang.org/docs/home.html.

13. Pro Git, 2014. – 421 с. – (Apress). – (Pro Git; вип. 2).

14. MySQL Enterprise Transparent Data Encryption [Електронний ресурс] – Режим доступу до ресхурсу: https://www.mysql.com/ products/enterprise/tde.html.

15. Understanding AES and RSA Encryption Algorithms [Електронний ресурс] – Режим доступу до ресурсу: https://www.section.io/engineering-education/aes-rsa-encryption/.

16. Завизіступ Ю.Ю., Рубан А.М. Аналіз методів пошуку інформації для персональних електронних бібліотек. Проблеми інформатизації : зб. тез доп. IX міжнар. наук.-практ. конф. том 2, м. Харків, 18-19 листопада 2021 р. Харків, 2021. С. 86.