КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ РАДІОФІЗИКИ, ЕЛЕКТРОНІКИ ТА КОМП'ЮТЕРНИХ СИСТЕМ

Кафедра радіотехніки та радіоелектронних систем

«На правах рукопису»

Робота допущена до захисту в ЕК
рішенням кафедри радіотехніки та радіоелектронних систем
від __ травня 2024 року, протокол № ___.
Завідувач кафедри доктор фіз.-мат. наук, професор
_____ Ігор АНІСІМОВ

ДИПЛОМНА РОБОТА МАГІСТРА
на тему:
«ЗАСІБ ПЕРЕГЛЯДУ ЗОБРАЖЕНЬ ФОРМАТУ DICOM ДЛЯ ПОТРЕБ WEB РЕАЛІЗАЦІЇ ТЕЛЕМЕДИЧНОГО ЗАСТОСУНКУ»

Виконав:
студент 2-го курсу магістратури
денної форми навчання
спеціальності 172 - Телекомунікації та радіотехніка
ОНП «Інформаційна безпека телекомунікаційних систем і мереж»
Кононов Віктор Михайлович _____

Науковий керівник:
канд. технічних наук, доцент
Жиров Генадій Борисович _____

Рецензент:
завідувач кафедри комп'ютерних наук
Державного університету інформаційно-комунікаційних технологій,
доктор технічних наук, професор
Вишнівський Віктор Вікторович _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів без
відповідних посилань
Студент _____ Віктор КОНОНОВ

Київ – 2024

РЕФЕРАТ

Дипломна робота магістра: 40 с., 9 рис., 2 табл., 4 дод. (16 с.), 31 джерело.

ВІЗУАЛІЗАЦІЯ, DICOM, WEB, КЛІЄНТ-СЕРВЕР, AJAX.

Об'єкт дослідження – візуалізація зображень в клієнт-серверному застосунку.

Предмет дослідження – забезпечення безпечного перегляду медичного діагностичного зображення за рахунок попередньої обробки даних на сервері.

Мета проекту – створення засобів попередньої підготовки на сервері зображень формату DICOM та метаданих для безпечного відтворення на клієнтській частині телемедичного web-застосунку.

Методи – об'єктно-орієнтоване програмування. Для розробки проекту використані мови програмування, C#, JavaScript, PHP.

Була розглянута можливість використання форматування даних DICOM-файлу на сервері для забезпечення візуалізації у телемедичному застосунку графічних засобами браузера та блокування передачі у відкритому вигляді персональних даних, які зберігаються у цьому файлі.

Реалізовані програмні складові для подальшого інтегрування в систему телеконсультацій (дистанційних консультацій) медичного призначення, яка реалізується на основі web-архітектури.

ЗМІСТ

Зміст 3
Вступ 4
1. Архітектура застосунків телемедичного призначення 5
1.1. Телемедицина як використання засобів електронних комунікацій 5
1.2. Вимоги інформаційної безпеки в телемедицині 8
1.3. Основні вимоги до програмного забезпечення для потреб телемедицини .. 10
1.4. Клієнт-серверна архітектура, як основа телемедичних програмних засобів 12
2. Реалізація обробки даних медичного формату DICOM 15
2.1. Вибір технології для доступу до даних формату DICOM 15
2.2. Структура формату даних файлу DICOM 16
2.3. Реалізація обробки текстових даних формату DICOM 21
2.4. Реалізація обробки графічних даних формату DICOM 24
2.5. Фільтрація текстових даних для забезпечення захисту персональних даних 26
3. Реалізація WEB засобів для візуалізації даних файлу формату DICOM 28
3.1. Архітектура застосунку 28
3.2. Розробка серверного боку застосунку 30
3.3. Розробка клієнтського боку застосунку 33
3.4. Тестування застосунку 35
Висновки 37
Перелік джерел посилання 38
Додатки 41

ВСТУП

В сучасному суспільстві використання засобів електронних комунікацій (або телекомунікаційних засобів) є одним з найпотужніших факторів пришвидшення різних видів діяльності. Одним з таких видів діяльності, яка вимагає максимально швидкої обробки інформації є медична діагностика. Вчасно встановлений діагноз та вибір протоколу лікування часто є запорукою збереження здоров'я або, навіть, життя.

Організм людини являє собою доволі складну динамічну систему, яка включає в себе багато систем саморегулювання. Відповідно діагностування стану організму пов'язане з великою кількості інформації, в тому числі такої, яка отримується досить складними методами досліджень і вимагає складного обладнання і подальшої обробки. Але навіть така діагностична інформація далеко не завжди є запорукою правильної її інтерпретації і вимагає залучення кваліфікованих лікарів-діагностів. Забезпечення швидкого дистанційного доступу фахівцю для визначення діагнозу є одним з напрямків розвитку телемедицини – практичної галузі, яка спирається на використання засобів та технологій електронних комунікацій для потреб медицини.

Складність структури медичних діагностичних даних вимагає застосування тегової структуризації даних, як це реалізується в одному з найбільш вживаних форматів для зберігання та пересилання графічних діагностичних даних – DICOM. Більшість сучасних засобів медичної діагностики підтримують такий формат, але його використання у дистанційному режимі вимагає спеціалізованого програмного забезпечення і для реалізації телемедичних систем є додатковим ускладненням. Є цікавим розглянути питання ефективності реалізації засобів перегляду даних з файлів такого формату використовуючи клієнт-серверну архітектуру на основі web-технологій. При цьому слід звернути додаткову увагу на те, що в багатьох випадках треба забезпечити умови медичної таємниці, тобто блокування на клієнтському боці доступу до персональних даних, які знаходяться в DICOM-файлах на сервері.

1. АРХІТЕКТУРА ЗАСТОСУНКІВ ТЕЛЕМЕДИЧНОГО ПРИЗНАЧЕННЯ

1.1. Телемедицина як використання засобів електронних комунікацій

В процесі лікування постійно отримується, накопичується та обробляється певна інформація. Серед основних етапів роботи з такою інформацією можна виділити [1]:

- діагностика – отримання інформації для визначення типу захворювання і його рівня розвитку, для цього широко застосовуються приладові методики (ультразвукове сканування, енцефалографія, декілька видів томографії тощо);
- безпосередньо процес лікування може плануватись або виконуватись з застосуванням складних технічних засобів, в тому числі комп'ютерних;
- контроль стану хворого в процесі лікування зазвичай пов'язаний з моніторингом ключових параметрів або додатковими дослідженнями для визначення правильності та подальшої корекції процесу.

Телемедицина має багато напрямків, серед яких слід виділити такі [2,3]:

- віддалена хірургія;
- консультативна хірургія;
- моніторинг;
- телеконсультації;
- дистанційне навчання.

Основним компонентом віддаленої хірургії [4] є робот з дистанційним керуванням, який використовується для виконання операції. Він складається з механізмів виконання зі спеціальними інструментами, якими керує хірург, камер, що забезпечують хірургу зображення операційного поля. За рахунок цього хірург здає команди роботу через спеціальний інтерфейс і бачить виконувану операцію на екрані.

Віддалена хірургія забезпечує доступ до висококваліфікованих спеціалістів (пацієнти з віддалених регіонів можуть отримати доступ до кращих хірургів) та зниження ризику передачі інфекцій (хірург не має контакту з пацієнтом);

При консультативній хірургії операцію виконує хірург, доступ якого є можливість забезпечити, але з врахуванням підказок від більш досвідченого колеги, який аналогічно до віддаленої хірургії бачить операційне поле, що забезпечують засоби електронної комунікації.

Телемедичний моніторинг [5,6] дозволяє віддаленим чином контролювати ключові діагностичні параметри пацієнта за допомогою автоматизації засобів вимірювання та електронної комунікації. Це дозволяє аналізувати стан пацієнта на відстані, забезпечуючи якісну медичну допомогу у випадках хронічних захворювань та дозволяють скоротити час перебування хворого в стаціонарі за рахунок перенесення нагляду за його станом у домашні умови.

Телеконсультації [7] – це процес проведення медичних консультацій за допомогою довільних засобів віддаленого зв'язку, в тому числі текстового, голосового, відео. Для проведення таких консультацій консультанту можуть надаватись усі види діагностичної інформації. Телеконсультації можуть проводитись ургентно або у відкладеному режимі, можуть охоплювати спілкування між пацієнтом та лікарем, або між лікарями різного фаху або кваліфікації. Для проведення консультацій часто реалізуються спеціалізовані програмно-апаратні комплекси та інформаційні ресурси. Одним з важливих питань таких ресурсів є захист конфіденційності медичних даних пацієнтів. Телеконсультації дозволяють забезпечити доступність медичної допомоги тим, хто знаходиться в віддалених регіонах або має обмежені можливості фізичного звернення до лікаря. Крім того лікарі можуть консультуватися з іншими спеціалістами для отримання додаткових порад або знань.

Дистанційне навчання в медицині [8] – це форма освіти в галузі медицини, де студенти, медичний персонал та медичні фахівці навчаються та підвищують свої знання за допомогою електронних засобів та інтернету. Воно може включати:

- курси, які надаються в електронному форматі (відеоуроки, електронні підручники та ресурси, консультативна допомога викладача тощо), що забезпечує процес навчання у зручному для студента ритмі в зручний час особливо важливо у випадку підвищення кваліфікації лікарів, які можуть суміщати навчання з роботою;
- вебінари, тобто інтерактивні онлайн-заняття з метою забезпечення широкої аудиторії спілкування;
- симуляційні навчання для підвищення медичних навичок.

Таким чином можна побачити, що сучасна медицина для забезпечення вчасної діагностики та ефективного лікування серед іншого спирається на використання цифрового стандартизованого запису діагностичних даних та ефективні засоби телекомунікації, забезпечення юридичної бази, в тому числі захисту медичних даних. В результаті зростання телемедицини відбувається на рівні 20-30% на рік [9].

На сучасний момент [10] 46% ринку телемедицини знаходиться у США. Слідом за Сполученими Штатами за розповсюдженістю телемедичних послуг йдуть Канада та Китай. В Європі найбільш телемедицина використовується у Великій Британії, а також в Данії, Швеції, Швейцарії, Естонії, Франції, Фінляндії та Італії. При цьому телемедицина зазвичай працює як безоплатно, так і на комерційній основі.

Для України [11] впровадження телемедицини надає можливість використання сучасних методів медичного обслуговування, особливо в районах з низькою щільністю населення (в сільській місцевості) та відповідно зменшення потреби з переміщенням пацієнтів і фахівців (економія коштів). Крім того, більш раціонально використовується ресурс високопрофесійних медиків, зменшується ймовірність медичних помилок. Особливо важливим є налаштування обміну медичною інформацією за міжнародними стандартами, в першу чергу з фахівцями ЄС та НАТО.

Слід згадати, що одним з витоків телемедицини була військова медицина [1], а наша країна знаходиться в стані війни з перспективою довготривалої

орієнтації на протидію зовнішнім воєнним викликам. Таким чином є потреба повернення до тісної співпраці з військовою медициною. Загалом розвиток телемедицини в Україні вимагає багатьох напрямків роботи: нормативно-правової підтримки, підготовки та організації персоналу та інформаційного забезпечення. Серед іншого надзвичайно важливим є і розвиток телемедичних систем та ресурсів [12].

1.2. Вимоги інформаційної безпеки в телемедицині

Використання телемедицини надзвичайно актуалізує ще одну задачу – забезпечення безпеки медичної інформації, частіше за все у вигляді лікарської (медичної) таємниці [13]. Така проблема пов'язана з двома факторами:

- відбувається передача загальними каналами зв'язку діагностичної інформації тобто інформації, пов'язаної з конкретним пацієнтом;
- більш широким є використання порівнянь різних блоків діагностичної інформації з метою діагностування (даний випадок порівнюється з раніше діагностованими).

Лікарська таємниця (правило, яке бере свій початок від клятви Гіппократа) традиційно застосовується до розмов між пацієнтами та лікарями при наданні медичної допомоги. З розвитком медичних технологій вона розповсюджується на заборону передачі третім особам будь-якої інформації, яка стосується діагностики або лікування пацієнта. Але в цифровому світі така інформація зберігається на цифровому носії, як запис в комп'ютерній системі, що дозволяє в разі зламу легко її скопіювати та розповсюдити.

Лікарська таємниця є основоположним елементом етики медичної практики та заснований на повазі до прав пацієнтів на приватність та конфіденційність їхніх медичних даних. Слід також брати до уваги, що незаконне отримання медичної інформації зловмисником може використовуватись як спосіб шантажу, тобто дозволяє отримати комерційну вигоду і є привабливим для зловмисника [1].

Основними аспекти лікарської таємниці є:

- конфіденційність медичних записів (інформації про стан здоров'я пацієнта, діагнози, хід лікування тощо);
- згода пацієнта (розголошення медичної інформації стороннім особам передбачає згоду пацієнта або його представника);
- обмеження доступу (доступ до медичної інформації повинен надаватись тільки особам, які безпосередньо пов’язані з медичним обслуговуванням даного пацієнта);
- захист даних (від крадіжки або несанкціонованого розголошення).

Важливим фактором лікарської таємниці є нерозповсюдження персональних даних, які часто супроводжують процес лікування, в тому числі діагностичні дані. До персональних даних відноситься будь-яка інформація, що пов’язана з конкретною ідентифікованою фізичною особою або дозволяє безпосередньо або опосередковано ідентифікувати таку особу:

- особиста інформація (ім'я, прізвище, дата народження, громадянство, адреса, електронна пошта, номер телефону, тощо);
- ідентифікаційні (номер паспорта, ідентифікаційний номер платника податків, соціальний страховий номер тощо);
- біометричні (відбитки пальців, запис голосу, контрольні точки обличчя, а також *певні медичні особливості, які можуть бути отримані з діагностичних даних*);
- фінансові (реквізити банківського рахунку, його стан, трансакції, інформацію для доступу до нього);
- медичні (діагнози, використовувані лікарських засоби тощо);
- поведінкові (історія перегляду ресурсів в Інтернеті, взаємодія в соціальних мережах, покупки тощо).

Обробка та зберігання таких даних повинні здійснюватися з дотриманням законів і стандартів забезпечення конфіденційності та інформаційної безпеки.

Є багато досліджень, присвячених інформаційній безпеці під час використання телемедицини Для захисту персональних даних пропонують

застосовувати досконалі криптоалгоритми, для затвердження важливої медичної інформації (наприклад діагнозу, лікарської консультації), які зберігаються або передаються в електронному вигляді, застосовувати цифровий підпис. Для захисту авторських прав на зображення, які використовуються для навчання, пропонують застосовувати цифрові водяні знаки. Слід зазначити, що розробка політики безпеки повинна виконуватись для конкретних задач та методів при впровадженні або розробці засобів телемедицини [9].

Забезпечення безпеки даних є надзвичайно важливою для проектів телемедицини, пов'язаних з акумуляцією великих обсягів даних, наприклад реалізацією загальних сервісів збереження даних, в тому числі на основі хмарних технологій. Саме поширення медичних даних у хмарі завдяки простоті доступу та розгортання телемедичних сервісів значно збільшило вразливість даних через ризик атак з метою перехоплення та розголошення інформації, DoS-атак (атак на відмову в обслуговуванні), впровадження в хмару зловмисницького програмного забезпечення, атак типу "людина в середині", спуфінгу. Ці проблеми мають бути розглянуті при проектуванні або розгортанні довільної телемедичної системи [7]. В Україні також приділяється велика увага юридичним питанням захисту інформації при використанні телемедицини [14].

1.3. Основні вимоги до програмного забезпечення для потреб телемедицини

Як вже було зазначено в попередніх параграфах, телемедицина вимагає розробки специфічних програмних засобів, орієнтованих на задану функціональність. Такі засоби повинні орієнтуватись на побудову деякої мережевої інформаційної системи на основі використання загальних засобів комунікації. На перших етапах розвитку телемедицини часто орієнтувались на використання телефонних мереж та створення спеціальних мереж, але з часом практично повністю перейшли на використання Internet [3].

Телемедичні програмні засоби можуть орієнтуватись на використання вузькоспеціальних варіантів архітектури зі специфічними форматами та

протоколами обміну даними [15]. Так, для потреб телеконсультування рівня лікар-лікар може бути організований спеціальний ресурс, який забезпечує як відкладений, так і ургентний режим консультування, має інструментарій формування запиту з можливістю приєднання приладових діагностичних даних, засоби формування відповіді (у випадку відкладеної консультації), або засоби інтерактивного спілкування (ургентний режим). Телеконсультативна система може орієнтуватись або на текстовий режим (орієнтація на випадок повільного зв'язку), або на голосове спілкування. Така система може забезпечити [15]:

- буферизацію інформації, що проходить через сервер;
- роботу з графічною інформацією, зокрема – стандартних медичних форматів, наприклад DICOM;
- мати можливість працювати в умовах повільних каналів зв'язку.

З врахуванням сучасного стану розвитку комп'ютерних мереж та мобільної телефонії часто кажуть навіть про можливість використання для потреб медичних телеконсультацій широковживаних месенджерів, хоча в даному випадку виникає питання медичної таємниці та використання даних у специфічних медичних стандартах запису. Загальновживані месенджери не забезпечують прийнятний рівень захисту персональних даних пацієнта. Для досягнення високого рівня захисту персональних даних, може бути використане об'єднання в деяку логічну мережу з окремим програмним забезпеченням (портал телемедицини). Прикладами таких захищених ресурсів є Telemed24 [16] і додаток пацієнта MedCard24 [17]. Більшість сучасних платформ телемедичного призначення дозволяють забезпечити підтримку роботи з файлами з медичними зображеннями (різні види томографії, рентгенограми, УЗД, ЕКГ тощо). Месенджери ж не мають такої підтримки, тобто для роботи з такими даними вимагають встановлення на комп'ютері медичного фахівця додаткового програмного забезпечення, що не завжди є зручним. Загальноприйнятими для передавання, збереження та інтерпретації медичної графічної інформації разом з текстовими метаданими, пов'язаними з нею, є використання міжнародного стандарту DICOM (ДСТУ ISO 17432:2009) [17].

З врахуванням сучасних підходів до побудови програмних засобів, орієнтованих на узгоджене обслуговування великої кількості користувачів за основу для побудови телемедичних ресурсів треба орієнтуватись на клієнт-серверну архітектуру із максимально можливим залученням стандартних компонентів, в тому числі з максимальною опорою на web. В Україні зараз працює велика кількість медичних Internet-ресурсів [16–18], вони значною мірою орієнтовані на забезпечення взаємодії рівня пацієнт-лікар, але зазвичай на допомогу пацієнту в пошуку прийнятного фахівця, або медичного закладу. Таким чином залишається багато питань в плані розробки, наприклад, телеконсультативних ресурсів з розвиненим інструментарієм. При цьому необхідно враховувати вартість розробки та використання, так, щоб забезпечити економічну прийнятність не тільки для комерційних, а й для державних лікувальних закладів [9]. Крім того, телемедичні програмні засоби повинні мати простий інтерфейс користувача, оскільки користувачам зазвичай не є фахівцем в сфері інформаційних технологій.

1.4. Клієнт-серверна архітектура, як основа телемедичних програмних засобів

Даний проект будемо орієнтувати на роботу з графічними даними, записаними у форматі DICOM. Для роботи з такими даними можуть використовуватись «десктопні» програмні засоби. В цьому випадку може бути забезпечена максимальна функціональність за рахунок того, що складна обробка даних файлу, необхідна для коректної візуалізації інформації, виконується безпосередньо на комп’ютері користувача. Це стосується перекодування метаданих та формування для відтворення пікселів 8-бітової глибини (в градаціях сірого) з масиву 16-бітових відліків у файлі. Альтернативою може бути web-програма, тобто застосунок клієнт-серверної архітектури, в якому тільки частина дій виконується засобами браузера або плагіна для браузера. Щодо DICOM, дані для відтворення на екрані користувача можуть бути частково підготовлені на сервері з остаточною обробкою на клієнті, а можуть повністю оброблятись на

сервері. Останній варіант є цікавим в плані універсальності робочого місця користувача та забезпечення кращої захищеності даних, оскільки за інформаційну безпеку, а відповідно і виконання вимог медичної таємниці повністю відповідає програмне забезпечення серверу, для якого простіше організувати достатній рівень інформаційної безпеки.

Було розглянуто декілька варіантів реалізації [19] програмного забезпечення для перегляду зображень з файлів у форматі DICOM як на основі плагіну, наприклад [20], так і хмарної реалізації [21]. Останній засіб перегляду проаналізуємо детальніше, розглядаючи його як основний аналог розробки, яка виконується в даному проекті. На ресурсі [21] надається можливість демонстраційного перегляду засобу [22]. При вході користувач може побачити список наявних на сервері обстежень з графічними даними, класифікованих за модальністю (способом отримання діагностичних даних). При виборі одного з обстежень відтворюється екран з конкретним кадром з обстеження – рис.1.1.

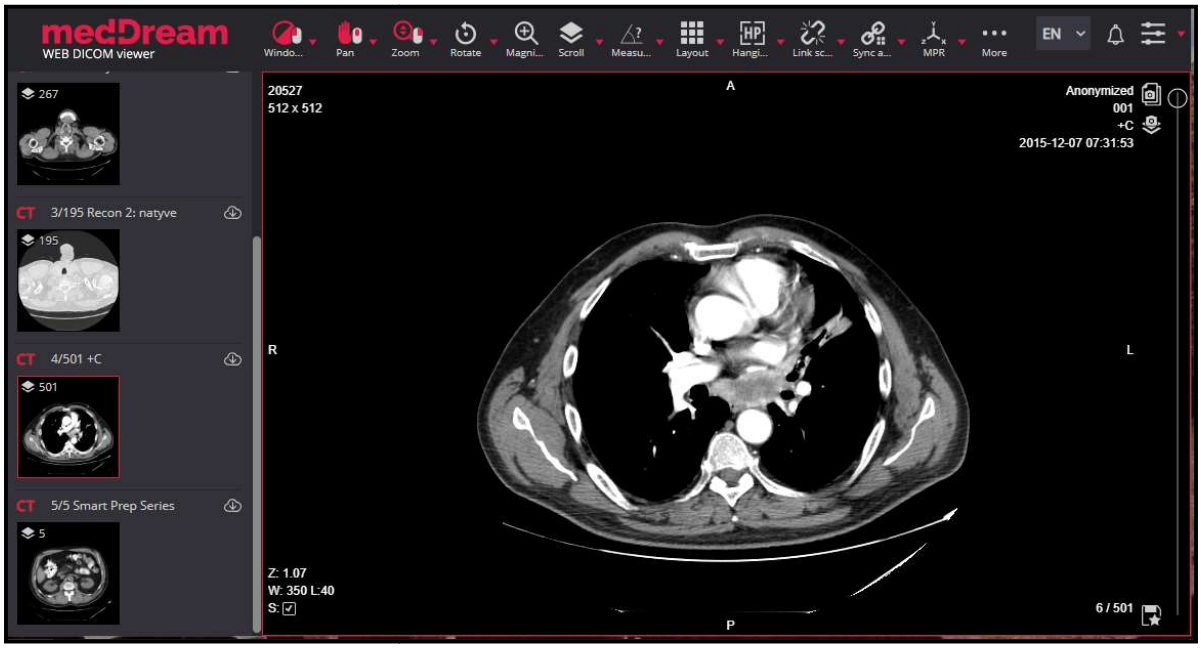


Рисунок 1.1 – Вигляд хмарного засобу перегляду даних DICOM з web-доступом при перегляді одного з обстежень (комп'ютерна томограма) [23].

Ресурс дозволяє завантажувати, переглядати та видаляти файли. Він є сертифікованим та має дозвіл FDA для діагностичного використання, має багатий

набір інструментів (масштабування, панорамування, вимірювання та додаткові інструменти для потреб деяких медичних галузей, наприклад радіології, кардіології. Файли перед завантаженням анонімізуються, завдяки чому персональні дані пацієнта або лікаря не розкриваються. Користувачі можуть ділитися посиланням на перегляд обстеження, наприклад для консультування. Таким чином формально цей ресурс пропонується для потреб телеконсультацій. Портал [21] містить також основні списки даних DICOM – теги, SOP (пари сервіс-об'єкт), модальності.

Серед особливостей слід зазначити можливість настроювання «вікна відображення», тобто способу перерахунку 16-бітових відліків в 8-бітовий рухом мишки. Програма надзвичайно швидко відпрацьовує такий перерахунок, при цьому він залишається доступним для активного кадру при вимкненні мереженого доступу. Тобто перерахунок в даному випадку виконується на клієнті. При перегортанні кадрів програма має помітну затримку при переході на новий кадр, але цієї затримки немає при поверненні до кадру, який раніше виводився на екран, що каже про якісну буферизацію переглянутого. Загалом даний програмний засіб має досить розвинену функціональність. Певним недоліком для потреб використання для потреб телеконсультацій є закритість коду і відсутність прав на його редагування, дещо ускладнене керування програмою, відсутність можливості переглянути метадані (теги DICOM-файлу). Зауважимо також, що даний продукт передбачає ліцензування (узгоджену в договорі кількість ліцензій і одночасних підключень, інтеграцій, інсталяцій тощо) використання за індивідуальною угодою з тримачем ліцензії.

Зазначимо, що наведений в цьому прикладі програмний продукт реалізований максимально професійно і досягти його функціональності та якості в рамках виконання дипломної роботи не є можливим. Продукт розроблено фірмою Softneta (спеціалізується на комунікаційних та візуалізаційних рішеннях медичного призначення), яка була заснована в 2007 році (Каунас) і має понад 16 років досвіду розробки медичних програмних засобів.

2. РЕАЛІЗАЦІЯ ОБРОБКИ ДАНИХ МЕДИЧНОГО ФОРМАТУ DICOM

2.1. Вибір технології для доступу до даних формату DICOM

Завданням даного проекту є реалізація web доступу до діагностичної інформації формату DICOM без використання плагіна для браузера, тобто тільки на основі попередньої підготовки даних на боці сервера. Оскільки браузер підтримує тільки обмежений перелік форматів графічних файлів, дані зображення, які утримуються у файлі даного типу, повинні бути перетворені у графічний файл прийнятного для браузера формату. Особливу увагу треба приділити 16-бітовим зображенням в градаціях сірого, оскільки настроювання візуалізації (перерахунок тіла зображення) потрібно виконувати також на сервері за клієнтським запитом. Крім того є зручним форматування інших даних з файлу в прийнятний для відображення в браузері вид також на боці сервера. Зауважимо, що з метою приховування персональних даних, що є складовою інформаційної безпеки, саме на сервері треба виконати відповідну фільтрацію даних з файлу, що відтворюється.

Серверна частина, яка обробляє запит, зазвичай реалізується використанням однієї з скриптових мов (PHP, Perl, Python, тощо). В даному проекті обрано PHP. Але значною частиною проекту є складова, яка реалізує безпосередньо підготовку даних потрібного файлу. З врахуванням того, що частиною даної роботи є розрахунок тіла зображення, використання для цього скриптових мов є неприйнятним. Кращим варіантом є передача запита до додаткової складової серверного забезпечення, для якої використана технологія програмування, що підтримує повну компіляцію, наприклад C++. Оскільки формат кваліфікаційної роботи передбачає обмежений термін розробки, як компроміс (програш у швидкодії приблизно в 1,5 рази) для обробки DICOM-файлу було обрано реалізацію додаткової складової сервера на основі Microsoft.Net (мова C#) [24].

З врахуванням такої гібридної архітектури серверного боку програмного засобу необхідно забезпечити обмін даними між його складовими. Найпростішим

способом є використання допоміжних файлів – для команди (завдання на обробку) та результатів її виконання. Такий спосіб обміну даними дозволяє досить просто масштабувати проект для одночасної роботи сервера з багатьма користувачами запуском потрібної кількості обробників DICOM-даних з виділенням для них окремих директорій для кожного сеансу. В цьому випадку такий обробник може бути реалізований в однопотоковому режимі, а повноцінність використання ресурсів сервера забезпечується саме запуском багатьох процесів. Альтернативою такої архітектури може бути багатопотоковий обробник з виділенням окремого потоку на кожний сеанс, але така архітектура дещо ускладнює розробку самого обробника і для скорочення часу розробки було обрано саме перший варіант.

2.2. Структура формату даних файлу DICOM

DICOM [25] є абревіатурою від Digital Imaging and Communications in Medicine. Це загальновживаний галузевий стандарт у медицині для запису отриманих приладових даних, їх зберігання, доставки через мережу та візуалізації включеною в обстеження графічною інформацією, яка отримана приладовим способом. Для приладових діагностичних зображень використовується DICOM 3.0. Стандарт було розроблено і підтримується Комітетом DICOM, авторське право закріплене за Національною асоціацією виробників електротехніки США (NEMA). Відповідно посилання на нього можна зустріти як NEMA PS3 та ISO 12052:2017. У стандарті крім формату файлу визначений протокол мережевого зв’язку, реалізований на прикладному рівні стеку TCP/IP.

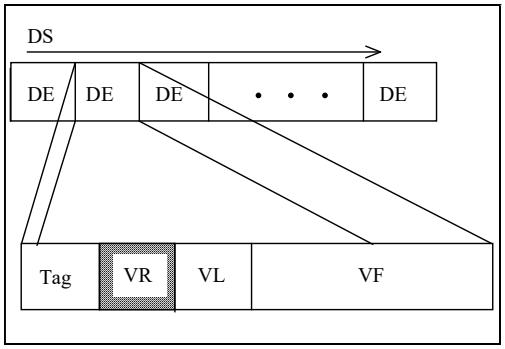
При внесенні даних у файл визначається модальність (тип), для якої будується відповідна сукупність мета-даних та даних для візуалізації. Стандарт визначає близько 40 модальностей, як приклад, наведемо деякі з них:

- CR – (Computed Radiography) Комп'ютерна рентгенографія;
- CT – (Computed Tomography) Комп'ютерна томографія;
- MR – (Magnetic Resonance) Магнітно-резонансна томографія;
- NM – (Nuclear Medicine) Ядерна медицина;

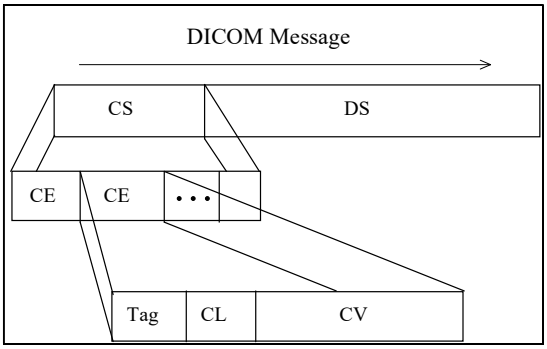
- PT – (Positron-emission Tomography) Позитронно-емісійна томографія;
- RF – (Radiofluoroscopy) Рентгенофлюороскопія;
- US – (Ultra Sound) ультразвукова діагностика;
- XA – (X-Ray Angiography) Рентгенівська ангіографія;

Дані у форматі DICOM мають доволі складну структуру, що значною мірою ускладнює повноцінну і універсальну реалізацію візуалізатора даних.

Специфікація стандарту DICOM визначає, що потік даних (DS – data set) впорядковуються у вигляді лінійної послідовності елементів (DE – data element), як показано на рис.2.1.а.



а



б

Рисунок 2.1 – Структура даних формату DICOM.

Стрілкою на рисунку показано напрямок збільшення адрес байтів, або послідовність мережевої передачі. Кожен з DE має в своєму складі 3 або 4 поля:

- Tag – ідентифікатор (заголовок) тегу;
- VR – тип тегу (необов’язкове поле);
- VL – довжина даних тегу (поля VF) ;
- VF – дані тегу.

Ідентифікатор тегу є впорядкованою парою 16-бітових беззнакових чисел. Перше з них визначає номера групи, друге – номер елемента в групі. Тобто групування тегів має дворівневу ієрархію. В межах кожної з груп теги слідують за зростанням номерів, а самі групи також вибудовують послідовність за зростанням номерів групи. За кожним конкретним номером групи та конкретним номером тегу закріплено певне змістове значення, тобто ідентифікатор тегу однозначно визначає призначення його змісту. Порушувати порядок слідування стандартом заборонено, але групи або теги, які не є потрібними для запису даних конкретного файлу, пропускаються. За рахунок зазначеного впорядкування при читанні даних легко виділяється блок з потрібним змістом.

В форматі DICOM реалізовано можливість розширення типів даних поза стандартом. Така вимога пов'язана з тим, що навіть детальний аналіз потреб окремого виробника медичного обладнання та фахівців, які використовують відповідні приладові дані, не дозволяє врахувати всієї різноманітності можливих даних. Відповідно, не порушуючи принцип впорядкування груп за зростанням номерів, групи були розділені на стандартизовані (мають парні номери) та нестандартизовані (непарні номери). Тобто за потреби після стандартної групи для конкретного приладу можна включити «дописування» розширення цієї групи специфічними для цього приладу даними, створивши групу з номером на одиницю більше. За семантику стандартних груп відповідає комітет DIMSE, вони повинні підтримуватись засобами універсального доступу до даних. Нестандартні групи можуть бути враховані програмними засобами, орієнтованими на конкретний діагностичний прилад, а інші програми просто ігнорують їх.

Поле VR є двобайтовим символьним. За його допомогою явним чином задається тип даних із стандартного переліку. Оскільки тип даних конкретного тегу в більшості випадків передбачається неявним чином, присутність цього поля в тегу є необов'язковою.

Далі включається поле довжини у байтах даних тегу (VF). При наявності поля VR ця довжина записується 16-бітовим числом, при відсутності VR – 32-бітовим. Цим автори формату заклали деяку проблему синтаксичного аналізу, яку

доводиться вирішувати при розробці програми доступу до даних, оскільки необхідно враховувати можливість різної інтерпретації двох байтів тегу, які можуть бути або символами, що визначають тип, або частиною чотирибайтового числа, що ускладнює алгоритм синтаксичного аналізу вхідних даних.

Останнім з полів тегу є поле даних VF, яке може містити число (форматоване або неформатоване), текст, зображення тощо. Довжина цього поля повинна бути вирівняна до парного числа.

Для мереженого сполучення визначається структура повідомлень (DICOM Message). Ця структура складається з сукупності команд (CS), за якою слідує послідовність даних (DS) (Рис.2.1.б). Сукупність команд є впорядкованою послідовністю командних елементів (CE), які мають по три поля:

- Тег – однозначно ідентифікує команду, задається впорядкованою парою 16-бітових беззнакових чисел, а саме номеру групи і номера елемента;
- CL – довжина у байтах поля CV;
- CV – значення команди (вирівнюється до парного числа).

Стандарт передбачає складну семантику можливих даних. Для прикладу деякі з них наведені в табл.2.1 (повну таблицю можна подивитись у главі 6 опису стандарту [25]). Значна частина даних, які вносяться в теги, записується у вигляді коду, що записується у текстовому форматі як ієрархічне об’єднання (через точку) стандартизованих ідентифікаторів. Приклад деяких з таких даних наведений в табл.2.2. Для зручності перегляду даних замість цих кодів необхідно відтворювати текст-пояснення, що вимагає при реалізації програми для доступу до даних використовувати досить великий за розміром словник, що забезпечує таку відповідність.

Підтримка графіки в форматі DICOM є досить розвиненою. Крім зображень в градаціях сірого з 8-бітовою глибиною пікселя підтримуються інші графічні формати, наприклад 24-бітовий RGB. З врахуванням того, що сучасні діагностичні прилади забезпечують динамічний діапазон відліку, що перевищує обсяг 8 бітів, досить часто використовується запис даних зображення в градаціях

сірого з глибиною пікселя 16 бітів. Серед іншого підтримується, хоча в реальних даних зустрічається доволі рідко, JPEG стиснення.

Таблиця 2.1 – Приклад деяких стандартизованих DICOM тегів, необхідних для роботи з діагностичними зображеннями.

Група	Елемент	Назва за стандартом
Характеристика обстеження		
0x0008	0x0000	Group 0008 Length
0x0008	0x0016	SOP Class UID
0x0008	0x0018	SOP Instance UID
0x0008	0x0020	Study Date
0x0008	0x0023	Acquisition Date
0x0008	0x0070	Manufacturer
0x0008	0x0080	Institution Name
0x0008	0x0081	Institution Address
0x0008	0x1041	Slice Location
0x0008	0x0090	Physician's Name
Дані для візуалізації зображення		
0x0028	0x0000	Group 0028 Length
0x0028	0x0002	Samples per Pixel
0x0028	0x0004	Photometric Interpretation
0x0028	0x0010	Rows
0x0028	0x0011	Columns
0x0028	0x0040	Image Format
0x0028	0x1050	Window Center
0x0028	0x1051	Window Width
0x0028	0x0060	Compression Code
0x0028	0x0100	Bits Allocated
0x0028	0x0101	Bits Stored
0x0028	0x0102	High Bit
0x0028	0x0103	Pixel Representation
0x0028	0x0030	Pixel Spacing

Реалізація програм для візуалізації DICOM-зображень може мати проблеми з окремими варіантами реалізаціями формату. Окремі розробники діагностичних приладів, наприклад Siemens, зберігають майже половину інформації, що входить в результати приладового обстеження, в нестандартних блоках (тобто з непарними номерами груп). Відповідно програмний засіб, який не є

спеціалізованим щодо цього виробника обладнання, не може отримати коректного доступу до цих даних.

Таблиця 2.2 – Приклад деяких даних для тегу ***SOP Class UID*** **(**0x0008; 0x0016**).**

Дані	Значення
1.2.840.10008.5.1.4.1.1.1	Computed Radiography Image
1.2.840.10008.5.1.4.1.1.2	CT Image
1.2.840.10008.5.1.4.1.1.3	Ultrasound Multi-frame Image
1.2.840.10008.5.1.4.1.1.4	MR Image
1.2.840.10008.5.1.4.1.1.5	Nuclear Medicine Image
1.2.840.10008.5.1.4.1.1.6	Ultrasound Image
1.2.840.10008.5.1.4.1.1.7	Secondary Capture Image
1.2.840.10008.5.1.4.1.1.8	Standalone Overlay
1.2.840.10008.5.1.4.1.1.9	Standalone Curve
1.2.840.10008.5.1.4.1.1.10	Standalone Modality LUT

2.3. Реалізація обробки текстових даних формату DICOM

Для реалізації підготовки даних файлу формату DICOM було реалізовано окремий C# [24] клас `public class Dicom` , який виконує структуризацію даних файлу у вигляді списку та усі допоміжні функції, потрібні для запису графічних даних у файл прийнятного формату. З врахуванням основного завдання проекту є потреба забезпечити зручний доступ до довільних тегів. Таким чином основою даних даного класу є список спеціально для цього визначених (рис.2.2) структур `public List<minitag> minitagList;` Докладніше – в Додатку А. Для заповнення цього списку з метою подальшого послідовного перегляду реалізований метод класу `public bool GetDicimMiniTags(string FileName)` .

З врахуванням різноманіття форматів полів даних тегів та особливостей кодування рядкових даних в MS.Net (використовується юнікод на відміну від однобайтового кодування в файлі формату DICOM) до зазначеного основного класу додано клас допоміжних інструментів `public static class Tools` , структура якого показана на рис. 2.3.

```
public struct minitag
{
    public int g;
    public int t;
    public string vr;
    public int vl;
    public int pTag;
    public int pData;
    //..
    public minitag(int pTag, int g, int t, string vr, int vl, int pData)
    {
        this.pTag = pTag;
        this.g = g;
        this.t = t;
        this.vr = vr;
        this.vl = vl;
        this.pData = pData;
    }
}
```

Рисунок 2.2 – Структура, яка є елементом списку тегів у класі Dicom.

```
public static class Tools
{
    static public string ByteToDoubleChar(Byte[] DataIn)...
    static public Byte[] DoubleCharToByte(string DataIn)...
    public static string ClearStringFronZeroBytes(string S)...
    static int MyRegistrationNum = 0;
    public static void ReadMyRegistrationNum()...

    public static bool debug = true;
    public static bool trace = !true;
    public static string msgText;
    public static bool start = true;
    //------------------------
    static Tools()...
    public static void ReadOnly(string FileName, bool Set)...
    public static bool WriteFile(string FileName, string Buffer)...
    public static bool ReadFile(string FileName, ref string Buffer)...
    static bool DecimalIsPoint = false;
    public static double ToDouble(string D)...
    public static long ConvertBytes(byte[] D, int index, int N, bool HiByteIsRigh
    public static string ConvertToChar(byte[] D, int index, int N)...
    public static UInt16 ConvertHex(string D)...
    public static void msg(string S)...
    public static void debug_msg(string S)...
    public static string GetMsg()...
}
```

Рисунок 2.3 – Структура допоміжного «інструментального» класу.

Призначення методів цього класу визначається назвами методів. Найбільш важливими є читання файлу в рядок та запис текстового файлу з наперед сформованого рядка єдиним блоком. Це дозволяє виконувати подальший синтаксичний аналіз вже з рядком, а не з файлом, та буферизацію даних, що формуються для виводу, що суттєво скорочую час обробки даних.

Таким чином DICOM-файл, що відкривається, переписується в рядок. Далі виконується послідовна обробка полів усіх тегів відповідно до п.2.2. Для цього проходимо послідовно по цьому рядку виділяючи поточний тег, з якого виділяються номери групи і тегу. Далі аналізується поле VR на наявність одного зі стандартних дволітерних кодів. При наявності такого коду як довжина даних (VL) тегу береться два наступних байти рядка, інакше – чотири (разом з байтами, що перевірялись на приналежність до VR. Далі записуються дані (VF).

Зазначимо, що введені таким чином дані тегів є прийнятними і зручними для маніпуляції з ними та виділення зі списку даних для візуалізації зображення, але є неприйнятними для виведення користувачу на екран значень тегів, оскільки маємо коди полів, а не їх текстові значення (такі, як в табл.2.2). Для перекодування можна використати формування і використання деякого словника. Через велику ресурсоємність такої архітектури програми було обрано інший шлях. Для формування тексту для подальшої візуалізації виконується виклик OpenSource програми, яка виділяє і перекодовує коди в значення для усіх тегів. Ця програма є зручною для візуалізації, але є неприйнятною для маніпуляції з даними і відтворення зображення. І саме комбінація двох алгоритмів обробки вхідних даних забезпечую повноту виконання завдання.

Для формування тексту, який надається користувачу, використовується метод класу Dicom `public int GetTagList(string FileName)`. Цей метод створює інший список `List<DICOM_Tag>()`, дані в якому дозволяють формувати рядок візуалізації вже в прийнятному для перегляду користувачем вигляді і одночасно з тим за потреби не включати в потік візуалізації окремі теги. Формування списку тегів для відправки на клієнтський бік виконує метод класу Dicom `public void PrepareAllTagList(String TagListFileName)`.

2.4. Реалізація обробки графічних даних формату DICOM

В межах задачі візуалізації зображення, дані якого містяться у файлі формату DICOM, була забезпечена підтримка 8- та 16-бітових зображень в градаціях сірого. В більшості випадків обстеження, яке має серії зображень, записується як сукупність відповідної кількості окремих зображень, згрупованих у деяку папку і зв'язаних між собою спільним ідентифікатором обстеження. При цьому об'єднання файлів в обстеження часто позначається спільною частиною імен цих файлів, хоча самим стандартом така вимога не регламентується. Але, як варіант, стандарт підтримує багатокадрові файли (Multi Frame), які при спільному наборі інших тегів мають об'єднаний тег графічних даних, тобто підряд в одне поле записані відліки усіх кадрів багатокадрового обстеження. Для підтримки роботи з цим варіантом реалізації файлу обстеження у відповідний метод класу Dicom додано врахування номеру активного (того, що зараз буде відтворюватись) кадру та його розмір, Ці два значення дозволяють коректно сформувати зсув при зчитуванні даних кадру.

Параметри для візуалізації зображення знаходяться в групі 0x0028 (табл..2.1). Для забезпечення їх виділення в зручній формі реалізовано метод `bool GetPicDataIndexes(ref List<minitag> minitagList_, Byte[] Bytes)` ,який перевіряє кожен тег списку методом `void GetPicParams(minitag mt)`. Ця обробка даних виконується при зчитування даних з файлу разом з формуванням списку тегів. Безпосередньо створення графічного файлу стандартного формату для подальшої відправки на клієнтський бік виконується функцією класу Dicom `void DataToBmp(String FileName)`. Цей метод використовує один з бібліотечних кодерів, в даному випадку – у формат JPEG. Для роботи цього кодера з даних, отриманих з файлу, попередньо формується масив відліків (тіло зображення). Для випадку 8-бітового DICOM-зображення достатньо їх скопіювати з поля VF відповідного тегу, але для 16-бітового випадку необхідно виконати перерахунок двобайових відліків в однобайтові з врахуванням позиції та

ширини вікна (значень, які формують лінійну функцію перерахунку). Параметри вікна знаходяться в самому файлі, як параметри рекомендованого відображення.

Стандарт DICOM має досить складну структуру кодування параметрів перерахунку в файлі в залежності від модальності (п.2.2). Врахування даних вікна з файлу не є принципово складним для включення в проект, але вимагає врахування багатьох особливостей стандарту і на даному етапі не було передбачено. Замість цього для перевірки ефективності роботи користувача на програмному засобі обраної в проекті архітектури було реалізовано автоматичний вибір на основі аналізу гістограми відліків та корекція вікна за запитом з клієнтського боку.

Для розрахунку параметрів гістограми було реалізовано нескладний клас `public static class Histo`, текст якого можна подивитись в Додатку Б. Єдиний метод цього класу розраховує середнє значення (використовується як позиція вікна) та дисперсію, через яку розраховується розмір вікна. При такому розрахунку враховується можливість як прямого (big-endian) так і зворотнього (little-endian) порядку слідування байтів в двобайтовому числі, оскільки обидва варіанти підтримуються стандартом DICOM.

Для подальшої відправки на клієнтський бік за запитом користувача в клас Dicom крім методу `void DataToBmp(String FileName)`, який забезпечує для випадку 16-бітових даних зображення автоматичний розрахунок параметрів вікна (використовується для виконання команди відкриття файлу), включені додаткові методи, які за заданим алгоритмом зменшують або збільшують розмір вікна та його позицію і формують вихідний графічний файл зі зміненим вікном (рис. 2.4).

```
public void DataToBmp_WP_Plus(String OutFileName)...
public void DataToBmp_WP_Minus(String OutFileName)...
public void DataToBmp_WS_Plus(String OutFileName)...
public void DataToBmp_WS_Minus(String OutFileName)...
```

Рисунок 2.4 – Обробники розміру вікна.

2.5. Фільтрація текстових даних для забезпечення захисту персональних даних

Інформаційна безпека (п.1.2) крім забезпечення надійності збереження інформації, заради чого використовується саме серверне збереження даних (сервер легше адмініструвати, в тому числі забезпечити резервне копіювання даних), передбачає обмеження доступу до даних для небажаних осіб. В медицині є термін «лікарська таємниця» – правове та етичне поняття, що визначає заборону медичному працівнику повідомляти третім особам інформацію про стан пацієнта, результати його обстеження або діагноз. Зазначимо, що інформація про людину, отримана за рахунок несанкціонованого доступу до його медичних даних може бути використана як інструмент шантажу, тобто в розвиток методів перехоплення такої інформації зламникам варто вкладати достатні ресурси. Відповідно виникає необхідність виділяти відповідні ресурси і для захисту від подібного зламу. Основним методом для такого захисту інформації є обмеження прав доступу до серверу з медичними даними.

Слід зазначити, що сучасна медицина для діагностики часто використовує порівняння результатів поточного обстеження пацієнта з раніше виконаними обстеженнями, з результатами яких вже пов'язаний характер перебігу хвороби та лікування. Таке порівняння забезпечує більш ранню та надійну діагностику. Крім того певні характерні екземпляри діагностичних даних використовуються для навчання медичного персоналу. Слід зауважити, що для зазначених задач крім безпосередньо графічних даних можуть бути корисними інші дані з обстеження, яке зберігається в форматі DICOM. Зрозуміло, що в багатьох випадках при відтворення таких даних користувачу частина інформації повинна бути прихована, тобто в рамках загальної архітектури даного проекту вилучатись при підготовці сервером даних для візуалізації на клієнтському боці.

В рамках даного проекту в класі Dicom крім зазначеного в п.2.3. методу `public void PrepareAllTagList(String TagListFileName)` було реалізовано метод `public void PrepareShortTagList(String TagListFileName)`. Цей

метод враховує попередньо завантажений в даний програмний модуль серверної складової програмного засобу список тегів, які відтворюються, і, як опція, список тегів, які відтворюються із заміною даних. Таким чином обстеження, що візуалізується, анонімізується по відношенню до пацієнта, до якого це обстеження відноситься. Зауважимо, що використання заміни значень тегів, в яких зберігаються персональні дані пацієнта або лікаря (за даними лікаря можна спробувати вирахувати пацієнта), з перезаписом самого DICOM-файлу також забезпечують анонімізацію, що може бути використано для створення бази діагностичних файлів для потреб порівняльної діагностики та навчання. Тобто розроблений клас може використовуватись для розв’язання подібної задачі. Для завантаження зазначеного налаштування в «інструментальному» класі Tools, про який було згадано в п.2.3, були реалізовані методи підтримки формату ini-файлу.

3. РЕАЛІЗАЦІЯ WEB ЗАСОБІВ ДЛЯ ВІЗУАЛІЗАЦІЇ ДАНИХ ФАЙЛУ ФОРМАТУ DICOM

3.1. Архітектура застосунку

В даному проекті на клієнтському боці (в браузері) виконується тільки візуалізація графічних та метаданих з обстеження у форматі DICOM, яке зберігається на сервері. Виділення даних та перенесення їх у прийнятний для браузера формат даних повністю виконується на серверному боці. Таке розділення функціональності між клієнтом та сервером забезпечує кращу швидкодію обробки даних, оскільки на серверному боці можуть бути задіяні технології значно швидші за можливості скриптової програми, яка виконується в браузері. Крім того попередня підготовка даних саме на сервері дозволяє виконувати фільтрацію даних в більш захищеному середовищі з метою блокування відтворення на клієнтському боці небажаної інформації, наприклад персональних даних з обстеження, що відповідає принципам інформаційної безпеки.

Для виконання зазначеного ланцюжка дій в рамках клієнт-серверної архітектури web-застосунку на клієнтському боці повинен формуватись запит з однією з команд, виконання яких виконує обробник даних формату DICOM (***далі DICOM-обробник***) та вказуванням даних до команди. На серверному боці програма, яку запускає web-сервер за даним клієнтським запитом (***далі web-обробник***), повинна її передати DICOM-обробнику, який обробляє запит та повідомляє web-обробнику про завершення виконання. Після цього результат передається на клієнтський бік. З врахуванням того, що перегляд даних з DICOM-файлу може бути тільки однією з багатьох задач робот з обраним обстеженням, необхідно забезпечити виконання запиту без денонсації web-сторінки в браузері, для чого використовується технологія AJAX [26].

В загальному випадку web-реалізація програмного засобу передбачає можливість роботи сервера з багатьма запитами одночасно (паралельно). При

виконанні відокремленої дії з даними, що зберігаються на серверному боці можливе використання запуску обробника та його закриття після підготовки ним даних для відправки на клієнтський бік. В даному випадку структура DICOM-сховища, а саме ієрархічне вкладення директорій з включенням обстеження у вигляді певної сукупності директорій, наповнених зв'язаними між собою серіями файлів, вимагає збереження поточного стану обробника запиту на певне обстеження та поточного стану його відтворення на клієнті. Тобто на кожного користувача в межах сесії повинен виділятись окремий екземпляр обробника. Для цього може використовуватись:

- запуск окремої нитки (потоку або процесу) для кожного з клієнтів;
- передача для конкретного клієнта однієї нитки з раніше запущеного пулу обробників.

DICOM-обробник може бути використаний в одному з цих режимів, але більш простим в реалізації є перший з них. Для цього в нього під час запиту (через параметр функції Main) є необхідним передати ідентифікатор (номер) сеансу, для якого створюється окрема робоча директорія (видаляється при завершенні сеансу). За рахунок запуску багатьох таких обробників забезпечується повноцінне використання багатоядерної архітектури процесору на серверному боці. Саме тому розробка DICOM-обробника була виконана без залучення багатопотоковості.

Для запуску нового процесу (DICOM-обробника) при відкритті нового сеансу може використовуватись постійно працюючий на серверному боці окремий менеджер запуску. Саме до нього при відкритті сесії повинен звертатись web-обробник (через файл-команду аналогічно до того, як забезпечується звернення до DICOM-обробника). В рамках даної дипломної роботи менеджер запуску не розроблявся, але він не є особливо складним і завдяки загальній модульності проекту може бути легко інтегрований в проект, так само як і разом з наведеною розробкою DICOM-обробника може бути включений у складніший проект з більш широкою функціональністю..

Зауважимо, що основною задачею даного проекту є саме DICOM-обробник. Інша частина є «обгорткою» для перевірки можливості інтегрування цього DICOM-обробника в іншу web-програму та загальної працездатності технології (в першу чергу прийнятності часу затримки оновлення даних в браузері). Нагадаємо, що з врахуванням обмеженості часу реалізації проекту для DICOM-обробника була обрана дещо компромісна технологія «часткової компіляції» вибором MS.Net замість C++.

3.2. Розробка серверного боку застосунку

DICOM-обробник, який за запитом з клієнтського боку виконує форматування графічних даних в форматі, прийнятному для браузера (в проекті використано JPEG) та переведення метаданих в HTML, було розроблено на основі класів, реалізація яких була описана в главі 2. Для взаємодії з клієнтом до функціональності зазначених класів додається деякий набір команд, самодостатній для роботи з даними DICOM-файлу:

- ***вибір папки*** (в ній можуть бути файли, які є серією в межах обстеження, або інші папки);
- ***вибір DICOM-файлу*** в папці;
- ***зміна «вікна відтворення»*** (яскравості та контрастності) для зображень в градаціях сірого з глибиною пікселя 16 бітів.
- ***перегортання кадрів*** (вперед та назад) для багатофреймового файлу.
- ***зміна повноти відтворення мета даних*** – скорочений список DICOM-тегів виводиться разом з обраним зображенням, повний список (з врахуванням налаштованої на сервері заборони показу захищених тегів) за окремим запитом.

DICOM-обробник (PrepareDicomData.exe) було реалізовано у вигляді MS.Net проекту архітектури Windows Forms [27], в який були включені зазначені вище класи та реалізований обробник перелічених команд. Зазначимо, що застосунки такої архітектури безпроблемно портуються на Linux використанням Mono, що забезпечує деяку кросплатформність проекту. Команди ззовні

передаються в DICOM-обробник через окремий текстовий файл «_TMP\Commands.txt». Після читання отриманої команди DICOM-обробник видаляє цей файл, виконує парсинг рядка з командою та забезпечує підготовку даних у вигляді файлів в папці «_DICOM_DATA». Ознакою завершення дії є запис відповідним методом класу Dicom файлу «_TMP\OK.txt».

В штатному режимі форма (вікно), яка є основою застосунку, запускається в прихованому вигляді, але для зручності відлагодження та як допоміжний інструмент розробника при включенні DICOM-обробника в більш складний проект, передбачено режим її візуалізації. Крім того на форму додані кнопки, які прямим чином (без файлу-команди) запускають тестові команди (рис. 3.1)

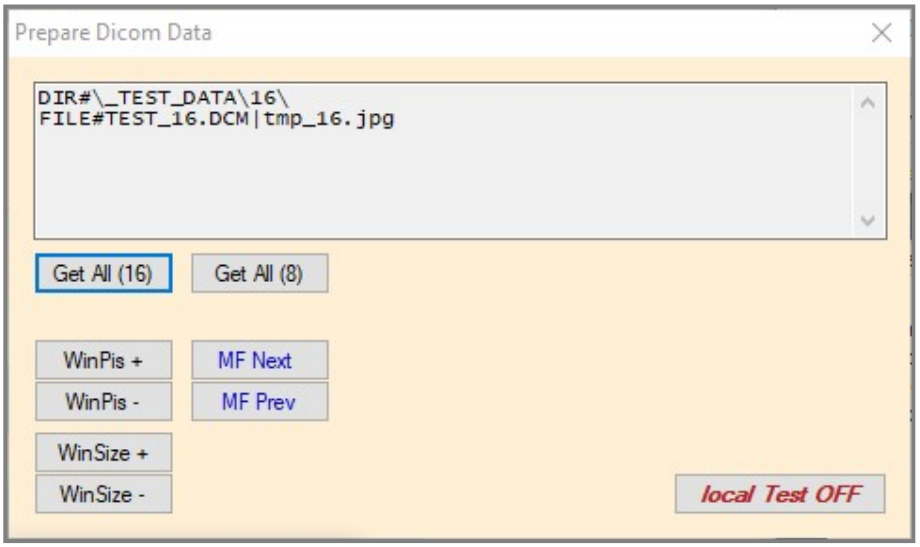


Рисунок 3.1 – Форма DICOM-обробника в тестовому (неприхованому) режимі.

Докладніше текст можна подивитись в Додатку В.

Web-обробник (на основі використання мови PHP [28]) було реалізовано у вигляді трьох окремих PHP файлів:

- select_dir.php – на основі заданої в папки повертає список файлів (або папок);
- get_image.php – завдання конкретного DICOM-файлу для подальшої роботи, повертається сформоване зображенні у форматі JPEG та скорочений список DICOM-тегів;

- change_data.php – зміна відображення даних для обраного файлу, видача повного списку DICOM-тегів або зміна вікна налаштування зображення для даних, збережених в 16-бітовій шкалі градацій сірого (вибір файлу в даному випадку не змінюється). Для коректного відтворення на клієнті графічних даних в ім’я JPEG-файлу включено номер, що інкрементується.

Ці модулі є схожими за функціональністю, хоч і мають певні відмінності, пов’язані з різницею структури даних, які формує DICOM-обробник. Текст одного з цих web-обробників наведено на рис.3.2. Суть роботи такого модуля є надзвичайно простою. Вона зводиться зо запису отриманої від клієнта команди у файл для DICOM-обробника і далі очікування стану готовності за появою файлу-прапорця (`while (!file_exists($OK)) usleep(20000);`)

```php
<?php
    $img = $_POST['text'];

    $commands = "DCM\\_TMP\\commands.txt";
    $OK = "DCM\\_TMP\\OK.txt";
    $result = "DCM\\_DICOM_DATA\\Result.htm";
    $imgToDel = "DCM\\_DICOM_DATA\\" . $_POST['imgToDel'];

    if (file_exists($OK)) unlink($OK);

    file_put_contents($commands, $img);

    while (!file_exists($OK)) usleep(20000);
    unlink($OK);
    unlink($imgToDel);

    $metaData=file_get_contents($result);
    echo $metaData;
?>
```

Рисунок 3.2 – Приклад одного з web-обробників для отримання даних від DICOM-обробника.

Формат команди зводиться до об’єднання у єдиний рядок слова-команди, яке відділяється символом « # » та додаткових даних для команди, які між собою розділяються символом « | ». Наприклад, для вибору DICOM-файлу з попередньо заданої папки задається команда FILE, до якої додаються імена файлу, з якого

будуть братись дані та файлу в форматі JPEG, в якому буде сформоване зображення, що буде відтворене на клієнті (в браузері).

3.3. Розробка клієнтського боку застосунку

Клієнтський бік в рамках даного проекту є джерелом запитів на вимогу користувача та засобом відтворення результату їх обробки сервером – виведення графічної інформації, що міститься в діагностичному DICOM-файлі та списку тегів, які є метаданими щодо цього зображення. В даному проекті досліджується сама технологія перенесення усієї обробки на серверний бік. Відповідно клієнтський бік було реалізовано у дещо спрощеному вигляді, без частини функціональності, яка за потреби реалізується розвитком даного програмного засобу, наприклад реалізація вимірювань. Крім того, одним із засобів пришвидшення роботи при перегортанні кадрів певної серії або зміни налаштування відтворення є буферизація на клієнтському боці того, що було переглянуто раніше. Запропонована архітектура застосунку дозволяє використовувти таку буферизацію, оскільки очищення робочої папки з підготовленими JPEG-файлами робиться тільки при зміні джерела даних. Таким чином для подальшого вдосконалення буферизації є достатнім дещо ускладнити принцип формування імен JPEG-файлів, які формується на сервері, і перевірку наявності вже підготовленої графічної інформації при формуванні на клієнті запиту. Це дозволить при певних діях користувача виводити оновлене зображення без додаткового запиту до серверу.

Стартовою точкою клієнтського боку є перегляд та входження в деяку робочу директорію. Оскільки, повторюємось, реалізація клієнта носить тестовий характер, а робота з папками є окремою (нескладною, але додатковою) частиною програми, як вхідна точка було обрано дві стартові папки (на сервері), в які попередньо вносяться потрібні тестові файли. З врахуванням особливостей роботи з 16-бітовими графічними даними для зручності тестування перед виконується сортування саме за цією ознакою. За відповідною дією користувача (вибір однієї з тестових папок) в браузері виводиться у вигляді інтерактивного

списку зміст відповідної папки (кожен рядок – ім’я файлу). При виборі користувачем одного з рядків цього списку виводиться графічна та супровідна текстова інформації з DICOM-файлу, який було обрано (рис.3.3).

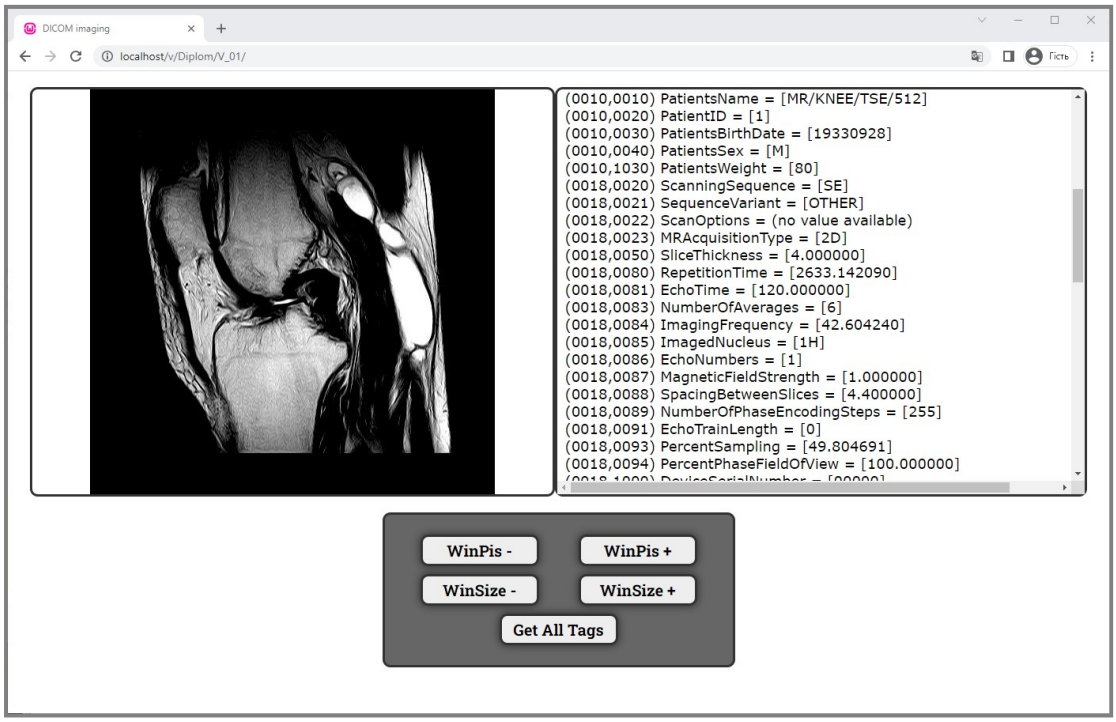


Рисунок 3.3 – Скріншот вікна браузера з відображенням медичного діагностичного зображення та списком тегів, що містяться у файлі формату DICOM.

Клієнтський бік реалізовано як нескладний HTML [29] код з винесенням сукупності стилів в окремий CSS [30] файл та з приєднанням окремого файлу, в якому використанням JavaScript [31] реалізовано усю програмну обробку на основі використання AJAX. Текст JS файлу наведений в Додатку Г.

На наведеному скріншоті (рис.3.3) можна побачити два вікна виведення даних (зображення та метадані) та кнопки вибору додаткових дій :

- Зміна параметрів «вікна відображення» 16-бітових графічних даних (візуально сприймається як зміна налаштування яскравості та контрастності);
- виведення повного списку DICOM-тегів.

Зауважимо, що графічна інформація виводиться масштабованим чином до заповнення відповідного контейнера по висоті. За потреби при перевищенні

ширини контейнера активується горизонтальний скролбар. Такий режим виведення дозволяє максимально зручно переглядати кадри обстеження при їх перегортанні. За потреби повноцінного (немасштабованого перегляду) на клієнтському боці повинна бути додана опція, яка в даному проекті не була реалізована (також через значною мірою тестовий характер клієнта).

3.4. Тестування застосунку

Для попереднього відлагодження основного модуля застосунку (DICOM-обробника) було реалізовано додаткову тестову форму (PrepareDicomDataTest.exe), яка формує у файлі для передачі команди відповідний тестовий запис для усіх потрібних в даному проекті випадків. З врахуванням того, що основний модуль запускається в прихованому режимі додатково додані команди візуалізації модуля та його закриття. Вигляд допоміжної форми наведений на рис.3.4.

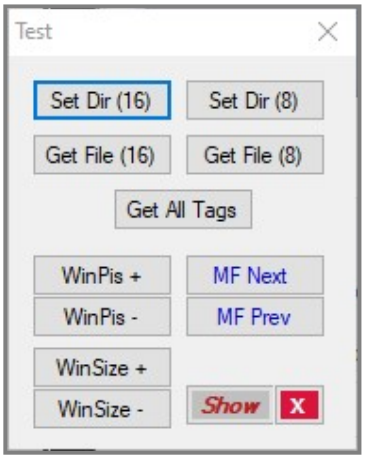


Рисунок 3.4 – Допоміжна форма для тестування DICOM-обробника.

Використання цього тестового модуля дозволило виконати попереднє тестування DICOM-обробника PrepareDicomData.exe (п.3.2) без web-складової даного проекту. Основним етапом тестування проекту загалом була перевірка працездатності на різних зразках діагностичних даних в форматі DICOM. Загалом було використано вибірку більше ніж з 50 файлів, вибраних з різних за модальністю обстежень (магніто-резонансна та комп’ютерна томографія,

ультразвукове сканування, рентгеноскопія). Особливу увагу було приділено файлам, які містять графічні дані в градаціях сірого 16-бітового формату. Розроблений програмний засіб показав повну працездатність на усіх файлах з обраної вибірки.

Найбільш цікавим було оцінити часову затримку, яка виникає при виконанні запиту до сервера для вибору іншого файлу або зміні параметрів «вікна візуалізації», оскільки виконання цих запитів є відносно ресурсоємним через потребу перерахунку усіх пікселів зображення. Тестування виконувалось на комп'ютері з ОС Windows 10 21H2 та процесором Intel Core i5 10400, тактова частота 2,9/4,3 Ггц. Паралельність обробки даних (6 ядер / 12 потоків) в даному випадку не має значення, оскільки основний модуль програмного застосунку (DICOM-обробник PrepareDicomData.exe) скомпільований в однопотоковому режимі. Нагадаємо, що клієнт-серверна архітектура в рамках даного проекту передбачає використання багатоядерності сервера на рівні одночасного звернення декількох клієнтів за рахунок запуску окремих екземплярів DICOM-обробника для кожного з сеансів. В даному тестуванні web-сервер (Apache 2.4.23) та браузер (Google Chrome 114.0.5735.199) були запущені на тому самому комп'ютері, що виключає затримку за рахунок мереженого сполучення. Тобто затримка, яка спостерігається визначається саме часом підготовки даних сервером.

Затримка від запиту до оновлення на браузер до відображення тексту (списку тегів) візуально взагалі не спостерігається. Затримка для оновлення діагностичного зображення не перевищує 0,1 – 0,2 сек., тобто не викликає особливого дискомфорту користувача. Таким чином технологія трансляції запиту клієнта у DICOM-обробник через файл практично не призводить до сповільнення роботи. Обробка даних зображення з використанням програмного засобу, реалізованому за технологією компіляції в проміжний код також забезпечує прийнятну швидкість роботи в рамках поставленої задачі. Зауважимо, що за швидкодією підготовки даних є резерв приблизно в 1,5 рази при умові реалізації аналогічної функціональності засобами з повною компіляцією (C++).

ВИСНОВКИ

1. Розроблено власний клас для доступу до текстових та графічних даних медичного формату даних DICOM. Цей клас дозволяє виконати перерахунок графічного блоку даних для забезпечення відтворення засобами браузера та формує блок текстових даних з вилученням персональних даних, що містяться у файлі.

2. Показано, що перерахунок на боці сервера (за запитом на зміну параметрів візуалізації) від клієнта даних зображення 16-бітової глибини пікселя, яке міститься у файлі формату DICOM, має прийнятний для користувача час затримки .

3. Програмні складові, розроблені в цій роботі, є прийнятними для використання при побудові системи віддалених медичних консультацій.

Публікація за матеріалами роботи:

Kononov V., Kononov M. DICOM viewer for web implementation of telemedicine application // Proceedings of the 19 International scientific conference “Electronic and applied physics”, Oct. 17–21, 2023, Kyiv, Ukraine. P.95-96.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кононов М. В., Новоселець М. К., Судаков О. О. Телемедицина. К.: Київський університет, 2003. 178 с.
2. Баязітов М. Р., Смирнов І. В., Адеїнка М., Ігельнік С. М. Сучасні інформаційні технології в роботі лікувально-профелактичних закладів: Перспективи для регіону. // Одеський медичний журнал. – 2005. – №4 (90). – С.9–12.
3. Владзимирский А. В. Телемедицина. – Донецк: ООО «Цифровая типография», 2011. 437 с.
4. Баязитов Н. Р. Телехирургия: Новые возможности лапороскопических технологий. // Досягнення біології та медицини. – 2008. – №2(12). – С. 84–97.
5. Gaikwad R, Warren J. The role of home-based information and communications technology interventions in chronic disease management: a systematic literature review. // Health informatics journal. – 2009. – 15. – P.122–146.
6. Bower P, Cartwright M, Hirani SP et al. A comprehensive evaluation of the impact of telemonitoring in patients with long-term conditions and social care needs: protocol for the Whole Systems Demonstrator cluster randomised trial. // BMC Heal Serv Res. – 2011. – 11(184).
7. Alenoghena C.O., Ohize H.O., Adejo A.O.. et al. Telemedicine: A Survey of Telecommunication Technologies, Developments, and Challenges. // Journal of Sensors and Actuator Networks. – 2023. – 12. – P. 20 – 39.
8. Сілкова О. В., Лобач Н. В. Використання ресурсів телемедицини в медичному освітньому процесі. // Педагогіка формування творчої особистості у вищій і загальноосвітній школах. –2021. – Т. 3. – № 75. – С.65–68.
9. Дубчак Л. О. Телемедицина: Сучасний стан та перспективи розвитку. // Системи обробки інформації. – 2017. – вип.1(147). – С.144–146.

10. Запорожан Л. П., Тренда Н. О., Литвинова О. Н. та інші. Необхідність розвитку української телемедицини за сучасних умов. // Вісник соціальної гігієни та організації охорони здоров'я України. 2020. – № 2 (84). – С.65–71.
11. Телемедицина в Україні. URL:https://telemed24.ua/articles/telemedecina-v-ukraini (дата звернення: 18.01.2024).
12. Закон України. Про підвищення доступності та якості медичного обслуговування у сільській місцевості. // Відомості Верховної Ради. – 2018. – № 5, ст.32). URL:https://zakon.rada.gov.ua/laws/show/2206-VIII#Text (дата звернення: 18.01.2024).
13. Москаленко К. Розголошення лікарської таємниці: деякі питання теорії та практики. // Цивільне право і процес. – 2006. – 8. – С. 27-31.
14. Булеца С. Телемедицина: Переваги і недоліки в правовому полі. // Право України. – 2020. – №3. – С. 49-60 – DOI: 10.33498/louu-2020-03-049. URL:https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/31317/1/ПУ_03_2020_сайт_17_04_2020-pages-49-60.pdf (дата звернення: 18.01.2024).
15. Охременко Л. О., Новоселець М. К., Кононов М. В. та інші – Система телемедичних консультацій, орієнтована на активне використання діагностичних зображень. // Реєстрація, зберігання і обробка даних. – 2000,. – Т.2.– №4. – С.48–57.
16. Український телемедичний портал Telemed24. URL:https://telemed24.ua/ (дата звернення: 18.01.2024).
17. Телемедична платформа Medcard24. URL:https://medcard24.net/ (дата звернення: 18.01.2024).
18. Електронна система охорони здоров'я в Україні. URL: https://ehealth.gov.ua/ (дата звернення: 18.01.2024).
19. What are the best Free DICOM Viewer to Open your Medical images. URL:https://www.imaios.com/en/resources/blog/5-best-dicom-viewer
20. Chrome DICOM image viewer. URL: https://chrome.google.com/webstore/detail/dicom-image-viewer/ehppmcooahfnlfhhcflpkcjmonkoindc (дата звернення: 18.01.2024)

21. DICOM Library online viewer URL:https://www.dicomlibrary.com/viewer-online/ (дата звернення: 18.01.2024).
22. Demo of DICOM Library online viewer (вхід). URL:https://demo.softneta.com/search.html (дата звернення: 20.03.2024)
23. Demo of DICOM Library online viewer (завантажене CT обстеження). URL:https://demo.softneta.com/?study=1.2.840.113619.2.55.3.4271045733.996.1449464144.595 (дата звернення: 20.03.2024).
24. C# documentation. URL:https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 20.03.2024).
25. DICOM. URL:https://www.dicomstandard.org/current (дата звернення: 18.01.2024).
26. W3schoolsUA (AJAX). URL:https://w3schoolsua.github.io/js/js_ajax_intro.html (дата звернення: 20.03.2024).
27. Desktop Guide (Windows Forms .NET). URL:https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-7.0 (дата звернення: 20.03.2024).
28. W3schoolsUA (PHP). URL:https://w3schoolsua.github.io/php/index.html (дата звернення: 25.03.2024).
29. W3schoolsUA (HTML). URL:https://w3schoolsua.github.io/html/index.html (дата звернення: 25.03.2024).
30. W3schoolsUA (CSS). URL:https://w3schoolsua.github.io/css/index.html (дата звернення: 25.03.2024).
31. W3schoolsUA (JavaScript). URL:https://w3schoolsua.github.io/js/index.html (дата звернення: 25.03.2024).

ДОДАТОК А

C# КЛАС ДЛЯ ДОСТУПУ ДО ДАНИХ У ФОРМАТІ DICOM

```
public class Dicom
{
    public string TxtResultFileName = "Result.htm";
    string DirFileName = "Result.htm";

    public bool OK = false;
    string TagString;
    string IniString;
    public bool visualise = true;
    DicomIni IniView;
    DicomIni IniRewrite;
    string dump = "_TMP\\dicom_dump";
    public List<DICOM_Tag> TagList;
    string WrkDir = "";
    public string DirForFileMode = "";
    public bool isDicomFile = true;
    public bool setDicomFileInfo = false;
    private String ActiveDir;
    private String FileName;
    private String ShortFileName;
    private String MyDir;

    //------------------------
    public Dicom()
    {
        MyDir = Directory.GetCurrentDirectory();
        IniRewrite = new DicomIni("_INI\\Rewrite.ini");
        IniRewrite.ReadIni();
        IniView = new DicomIni("_INI\\view.ini");
        IniView.ReadIni();
        OK = false;
        Clear();
        WrkDir = Directory.GetCurrentDirectory();
    }

    //==================== User Interface Functions ====================================begir
    public void OpenDir(String ActiveDir)
    {
        this.ActiveDir = ActiveDir;
        string[] FilesArr = Directory.GetFiles(ActiveDir);
        for (int n = 0; n < FilesArr.Length; n++)
            FilesArr[n] = Tools.GetLastEntryName(FilesArr[n]);

        string json = "[";
        for (int n = 0; n < FilesArr.Length; n++)
        {
            json += "\"";
            json += FilesArr[n];
            json += "\"";
            if (n < FilesArr.Length-1) json += ",";
        }
        json += "]";
        Tools.WriteFile(MyDir + "\\_DICOM_DATA\\" + DirFileName, json);
        OkMsg = "JSON";
    }
```

```
        public void OpenFile(String InFileName, String OutFileName)
        {
            ClearOutDataDir();
            this.ShortFileName = InFileName;
            FileName = ActiveDir + ShortFileName;
            GetTagList(FileName);
            GetPicData();
            SetDicomWindow();  //auto
            DataToBmp(MyDir + "\\_DICOM_DATA\\" + OutFileName);
            PrepareShortTagList(TxtResultFileName);
        }
        public void GetBitmap(int winSize, int winPosition, String OutFileName)
        {
            SetDicomWindow(winSize, winPosition);
            DataToBmp(MyDir + "\\_DICOM_DATA\\" + OutFileName);
        }
        public void NextInMultiFrame(String OutFileName)...
        public void PrevInMultiFrame(String OutFileName)...
        public void PrepareShortTagList(String TagListFileName)
        {
            WriteTagListString(false);
            Tools.WriteFile(MyDir + "\\_DICOM_DATA\\" + TagListFileName, StringOfTags);
        }
        public void PrepareAllTagList(String TagListFileName)
        {
            WriteTagListString(true);
            Tools.WriteFile(MyDir + "\\_DICOM_DATA\\" + TagListFileName, StringOfTags);
        }

        public string OkMsg = "-";
        public void WriteOK()...

        //==================== User Interface Functions ==============================
        public struct minitag...
        //-----
        public List<minitag> minitagList;
        int ptr;

        //for visualisation
        public byte[] FileBytes;
        public int PicDataIndex;

        public int PH = -1;         //0x0028 0x0010
        public int PW = -1;         //0x0028 0x0011
        public int PbirsAllocated = -1;//0x0028 0x0100
        public int PbirsStored = -1;//0x0028 0x0101
        public int Phibit = -1;     //0x0028 0x0102
        public double PwinPos = -1;    //0x0028 0x1050
        public double PwinWdth = -1;   //0x0028 0x1051
        public double Pb = -1;       //0x0028 0x1052
        public double Pa = -1;       //0x0028 0x1053
        public int PicBody = -1;  //0x7fe0,0x0010
        public int PicLen = -1;

        public bool PicDataIsNew = false;
        public bool ClearPic = false;
```

```
        //---------------------------------
        public bool GetDicimMiniTags(string FileName)
        {
            if (PicDataIsNew) return false;
            bool rez = false;
            if (!File.Exists(FileName)) return false;
            try
            {
                Tools.ReadOnly(FileName, false);
                FileStream fs = File.Open(FileName, FileMode.Open);  //FileMode.Create
                long FLen = fs.Length;
                FileBytes = new byte[FLen];
                fs.Read(FileBytes, 0, (int)FLen);
                fs.Close();
            }
            catch
            {
                Tools.debug_msg("ReadForVisualise");
            }
            rez = GetMiniTagList(FileBytes, out minitagList);
            return rez;
        }

        public bool GetMiniTagList(Byte[] Bytes, out List<minitag> minitagList_)
        {
            bool ok = false;
            minitagList_ = new List<minitag>();
            ok = GetPicDataIndexes(ref minitagList_, Bytes);
            PicDataIsNew = true;
            return ok;
        }

        ///----Get Tags for Picture Visualisation---------------------
        bool GetPicDataIndexes(ref List<minitag> minitagList_, Byte[] Bytes)
        {
            bool isDICOM = true;
            try
            {
                int maxGroupNum = 0;
                ptr = FindDataBegin(Bytes);
                bool ok = false;
                int L = Bytes.Length;
                while (!ok && ptr < L)
                {
                    //     if (tag == 0x1140)   //test!!!
                    //     tag = 0x1140;        //test!!!
                    GetNextTag(Bytes);
                    minitag mt = new minitag(pTag, group, tag, vr, vl, pData);
                    if (group < maxGroupNum) return false; //not DICOM !!!!
                    if (group > maxGroupNum) maxGroupNum = group;
                    minitagList_.Add(mt);
                    GetPicParams(mt);
                    if (ptr > L + 10) isDICOM = false;
                }
            }
            catch
            {
                Tools.debug_msg("GetPicDataIndex");
                return false;
            }
            return isDICOM;
        }
        //------
```

```
        void GetPicParams(minitag mt)
        {
            int p = mt.pData;
            //set parameters for Picture
            if (mt.g == 0x0028 && mt.t == 0x0010)
                PH = (int)Tools.ConvertBytes(FileBytes, p, 2, true);
            if (mt.g == 0x0028 && mt.t == 0x0011)
                PW = (int)Tools.ConvertBytes(FileBytes, p, 2, true);
            if (mt.g == 0x0028 && mt.t == 0x0100)
                PbirsAllocated = (int)Tools.ConvertBytes(FileBytes, p, 2, true);
            if (mt.g == 0x0028 && mt.t == 0x0101)
                PbirsStored = (int)Tools.ConvertBytes(FileBytes, p, 2, true);
            if (mt.g == 0x0028 && mt.t == 0x0102)
                Phibit = (int)Tools.ConvertBytes(FileBytes, p, 2, true);
            string buf = "";
            if (mt.g == 0x0028 && mt.t == 0x1050)
            {
                buf = Tools.ConvertToChar(FileBytes, p, mt.vl);
                try
                {
                    int i = buf.IndexOf('\\');
                    if (i > 0) buf = buf.Substring(0, i - 1);
                    PwinPos = Tools.ToDouble(buf);
                }
                catch { Tools.debug_msg("format tag-1050"); }
            }
            if (mt.g == 0x0028 && mt.t == 0x1051)
            {
                buf = Tools.ConvertToChar(FileBytes, p, mt.vl);
                try
                {
                    int i = buf.IndexOf('\\');
                    if (i > 0) buf = buf.Substring(0, i - 1);
                    PwinWdth = Tools.ToDouble(buf);
                }
                catch { Tools.debug_msg("format tag-1051"); }
            }
            // Picsel recalc   F(x)=Pa*x+Pb
            if (mt.g == 0x0028 && mt.t == 0x1052)
            {
                try
                {
                    buf = Tools.ConvertToChar(FileBytes, p, mt.vl);
                    Pb = Tools.ToDouble(buf); ;
                }
                catch { Tools.debug_msg("format tag-1052"); }
            }
            if (mt.g == 0x0028 && mt.t == 0x1053)
            {
                try
                {
                    buf = Tools.ConvertToChar(FileBytes, p, mt.vl);
                    double Pa = Tools.ToDouble(buf);
                }
                catch { Tools.debug_msg("format tag-1053"); }
            }
            //Pic Body
            if (mt.g == 0x7fe0 && mt.t == 0x0010)
            {
                PicBody = p;
                PicLen = mt.vl;
            }
        }
```

```
//------
 int group;
 int tag;
 string vr;
 int vl;
 int pTag;
 int pData;
 //...
 void GetNextTag(Byte[] FileBytes)...
 //-----
 bool test_vr_isLong()
 {
     if (vr == "SQ")//test on ierarchic !!!!
     {
     }
     if (vr == "OB" || vr == "OW" || vr == "OF" || vr == "SQ" || vr == "UT" || vr == "UN")
     { ptr += 4; return true; }
     else { ptr += 2; return false; }
 }
 //-----
 int zeroHeaderLen = 0;
 int FindDataBegin(Byte[] FileBytes)...
 //-------------------------------------------------------------


 public void Clear()...

//--------Get Tag List for View (by dcmdump) ----------------
string oldFileName = "";
//---

public int GetTagList(string FileName)
{
    int nnn = oldFileName.CompareTo(FileName);
    if (nnn == 0)
    {
        return 0;
    }
    oldFileName = FileName;
    Clear();
    //  string mydir = Directory.GetCurrentDirectory();
    Directory.SetCurrentDirectory(WrkDir);
    bool ok = false;
    try
    {
        int c = 0;
        while (!ok && c++ < 20)
        {
            if (!File.Exists(dump)) ok = true; else File.Delete(dump);
        }
    }
    catch
    {
        Tools.debug_msg("GetTAgList");
    }
    if (!ok)
        DicomIni.msg = "Can't kill temporary dump file !";

    Process p = new Process();
    p.StartInfo.Arguments = @"/c " + "_DCMTC\\dcmdump.exe \"" + FileName + "\" > _TMP\\dicom_dump";
    p.StartInfo.FileName = "cmd.exe";
    p.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
    p.Start();
```

```
        int counter = 0;
        int counter_max = 200;
        OK = false;
        while (!OK && ++counter < counter_max)
        {
            OK = ReadFile("_TMP\\dicom_dump", ref TagString);
            if (!OK) Thread.Sleep(50);
        }
        if (OK) PrepareTagList();
        isDicomFile = true;
        setDicomFileInfo = true;
        if (!OK) { isDicomFile = false; ClearPic = true; return 0; }
        PicDataIsNew = false; // for off the  block of ReadForVisualisation
        if (visualise) GetDicimMiniTags(FileName);
        string[] Smtl = new string[minitagList.Count];
        for (int n = 0; n < minitagList.Count; n++)
        {
            minitag mt = minitagList[n];
            string line = mt.g.ToString() + "  " + mt.t.ToString() + "  " + mt.vr
                + "  " + mt.vl.ToString();
            Smtl[n] = line;
        }
        if (Tools.debug) cmpLists(FileName);
        return 0;
    }

    //------------------------
    bool ReadFile(string FileName, ref string Buffer)...

    //------------------------
    bool IsNotDicom = false;
    void PrepareTagList()...

    Bitmap bmp;
    //Rectangle rect;
    bool DoublePixel = false;
    int FrameSize = -1;
    int ActiveFrame = 0;
    int N_frames = 1;
    bool Hi15 = true;

    void GetPicData()
    {
        //userControl12.InitPic(PW, PH);
        int DPix = 0;
        int W = PW;
        int H = PH;
        ScreenData = new UInt16[W, H];
        FrameSize = PW * PH;
        if (PbirsAllocated > 8) FrameSize *= 2;
        N_frames = PicLen / FrameSize;
        ActiveFrame = 0;

        Hi15 = true;
        double a = 1;
        double b = 0;
        bool rescale = false;
        if (Pa > 0)
        {
            a = Pa;
            b = Pb;
            rescale = true;
        }
```

```
            if (PbirsAllocated > 8)
            {
                int L = PW * PH;
                for (int N = 0; N < L; N += 2)
                {
                    if (FileBytes[N + 1] > 127) FileBytes[N + 1] = 0;
                }
            }

            if (Phibit == 0) Hi15 = false;


            //WPos = 1;
            //WWidth = 1;
            if (PbirsAllocated > 8)
            {
                DoublePixel = true;
                for (int y = 0; y < H; y++)
                    for (int x = 0; x < W; x++)
                    {
                        int N = ActiveFrame * FrameSize + y * 2 * W + 2 * x + PicBody;
                        if (Hi15) DPix = ((FileBytes[N + 1] * 256 + FileBytes[N]));
                        else DPix = ((FileBytes[N] * 256 + FileBytes[N + 1]));
                        ScreenData[x, y] = (UInt16)DPix;
                       // WPos = (int)(0.5 + PwinPos);
                       // WWidth = (int)(0.5 + PwinWdth);
                    }
            }
            else
            {
                DoublePixel = false;
                for (int y = 0; y < H; y++)
                    for (int x = 0; x < W; x++)
                        ScreenData[x, y] = FileBytes[ActiveFrame * FrameSize + y * W + x + PicBody];
            }
        }


        void SetDicomWindow()
        {
            Histo.Calc(ref FileBytes, ActiveFrame * FrameSize + PicBody, FrameSize, DoublePixel, Hi15);
            winSize_ = (int)(Histo.sko*3 + 0.5);
            winPosition_ = (int)(Histo.mid + 0.5);
        }

        void SetDicomWindow(int winSize, int winPosition)
        {
            winSize_ = winSize;
            winPosition_ = winPosition;
        }


        public void ClearOutDataDir()
        {
            //Directory.CreateDirectory(MyDir + "\\_DICOM_DATA");
            DirectoryInfo dir = new DirectoryInfo(MyDir + "\\_DICOM_DATA");
            foreach (FileInfo fi in dir.GetFiles())
            {
                try
                {
                    fi.Delete();
                }
                catch (Exception) { }
            }

        }
```

```
        public void DataToBmp_WP_Plus(String OutFileName)
        {
            if (winPosition_ == 32000 || winPosition_ == 0) return;  //file not opened
            if (winPosition_ > Position_max * 0.8) return;
            winPosition_ = (int)(winPosition_ * 1.1);
            DataToBmp(MyDir + "\\_DICOM_DATA\\" + OutFileName);
        }
        public void DataToBmp_WP_Minus(String OutFileName)
        {
            if (winPosition_ == 32000 || winPosition_ == 0) return;  //file not opened
            if (winPosition_ > Position_max * 0.8) return;
            winPosition_ = (int)(winPosition_ * 0.9);
            DataToBmp(MyDir + "\\_DICOM_DATA\\" + OutFileName);
        }

        public void DataToBmp_WS_Plus(String OutFileName)
        {
            if (winPosition_ == 32000 || winPosition_ == 0) return;  //file not opened
            if (winSize_ > Position_max * 0.5) return;
            winSize_ = (int)(winSize_ * 1.4);
            DataToBmp(MyDir + "\\_DICOM_DATA\\" + OutFileName);
        }

        public void DataToBmp_WS_Minus(String OutFileName)
        {
            if (winPosition_ == 32000 || winPosition_ == 0) return;  //file not opened
            winSize_ = (int)(winSize_ / 1.4);
            DataToBmp(MyDir + "\\_DICOM_DATA\\" + OutFileName);
        }

       void DataToBmp(String FileName)
       {
           size.X = PW;
           size.Y = PH;
           Rectangle rect = new Rectangle(0, 0, PW, PH);
           int size_X = size.X;
           if (size.X % 4 != 0) size_X += (4 - size.X % 4);
           bmp = new Bitmap(PW, PH, PixelFormat.Format8bppIndexed);
           ColorPalette cp = bmp.Palette;
           for (int k = 1; k < 256; k++) cp.Entries[k] = Color.FromArgb(255, k, k, k);
           bmp.Palette = cp;

           BitmapData bmpData = bmp.LockBits(rect, ImageLockMode.ReadWrite, bmp.PixelFormat);
           int pixels = size_X * size.Y;
           IntPtr ptr = bmpData.Scan0;
           byte[] Values = new byte[pixels];
           System.Runtime.InteropServices.Marshal.Copy(ptr, Values, 0, pixels);
           double K = 256.0 / winSize_;
           if (DoublePixel)
           {
               for (int y = 0; y < size.Y; y++)
               {
                   for (int x = 0; x < size.X; x++)
                   {
                       int Vnew = (int)((ScreenData[x, y] - winPosition_) * K) + 128;
                       if (Vnew < 0) Vnew = 0;
                       if (Vnew > 255) Vnew = 255;
                       Values[y * size_X + x] = (byte)Vnew;
                   }
               }
           }
```

```
            else
            {
                for (int y = 0; y < size.Y; y++)
                {
                    for (int x = 0; x < size.X; x++)
                    {
                        Values[y * size_X + x] = (byte)ScreenData[x, y];
                    }
                }
            }
            System.Runtime.InteropServices.Marshal.Copy(Values, 0, ptr, pixels);
            bmp.UnlockBits(bmpData);
            bmp.Save(FileName, System.Drawing.Imaging.ImageFormat.Jpeg);
        }
        //----------- Set TagList to File  -----------------
        public String StringOfTags;
        void WriteTagListString(bool ShowAll)
        {
            if (!OK) return;
            try
            {
                StringOfTags = "<br>"; //"<br>\r\n";
                for (int N = 0; N < TagList.Count; N++)
                {
                    if (ShowAll)
                    {
                        StringOfTags+= TagList[N].webLine;
                       // StringOfTags += "<br>\r\n";
                    }
                    else
                    {
                        if (TagList[N].NeedShow)
                        {
                            StringOfTags += TagList[N].webLine;
                        //    StringOfTags += "<br>\r\n";
                        }
                    }
                }
            }
            catch
            {
                Tools.debug_msg("Catch  PrepareList");
            }
        }
    }
```

ДОДАТОК Б

ДОПОМІЖНІ КЛАСИ ДЛЯ ДОСТУПУ ДО ДАНИХ У ФОРМАТІ DICOM

```
    public static class Tools
    {
        //-----------------------------------------------------
         static public string ByteToDoubleChar(Byte[] DataIn)...
        //--------
         static public Byte[] DoubleCharToByte(string DataIn)...
        //----------------------
        public static string ClearStringFronZeroBytes(string S)...
        //------------------------
        static int MyRegistrationNum = 0;
//        public static int NextNum = 0;
        //--
        public static void ReadMyRegistrationNum()...
        //------------------------
        /* ...

        //------------------------

        public static bool debug = true;
        public static bool trace = !true;
        public static string msgText;
        public static bool start = true;
        //------------------------
        static Tools()...

        //------------------------
        public static void ReadOnly(string FileName, bool Set)...

        //------------------------
        public static bool WriteFile(string FileName, string Buffer)...
        //------------------------
        public static bool ReadFile(string FileName, ref string Buffer)...


        //------------------------
        static bool DecimalIsPoint = false;
        public static double ToDouble(string D)...
        //------------------------
        public static long ConvertBytes(byte[] D, int index, int N, bool HiByteIsRight)...

        //------------------------
        public static string ConvertToChar(byte[] D, int index, int N)...
        //------------------------
        public static UInt16 ConvertHex(string D)...
        //------------------------
```

```
        public static void msg(string S)...
        //-----------------------
        public static void debug_msg(string S)...
        //-----------------------
        public static string GetMsg()...

        //------------------------------
        public static string GetLastEntryName(string Dir)...

        //------------------------------
        public static string GetFilePath(string Dir)...
    }
    //************************ IniUnit ******************************
    public struct IniUnit...
    //************************ DicomIni *****************************
    public class DicomIni...
    //************************** DICOM_Tag **************************
    public class DICOM_Tag...
    //********************** Dicom ******************************
public static class Histo
{
    public static int[] H0;
    public static int[] H1;
    public static int[] HH;
    public static double mid;
    public static double sko;
    public static bool Hi15; //Hi bit is 15 !!!
    static bool file_Hi15;
    public static bool Hi_OK;
    static int noise_0 = 50;          // !!!

    public static void Calc(ref byte[] Data, int begin, int len, bool DoublePixel, bool _Hi15)
    {
        if (Data == null) return;
        file_Hi15 = _Hi15;
        int n0 = begin;
        int max = begin + len;
        H0 = new int[256];
        if (DoublePixel)
        {
            H1 = new int[256];
            HH = new int[256];
            int S0 = 0;
            int S1 = 0;
            byte max0 = 0;
            byte max1 = 0;
            for (int n = n0; n < max; n += 2)
            {
                byte pix0 = Data[n];
                byte pix1 = Data[n + 1];
                S0 += pix0;
                S1 += pix1;
                H0[pix0]++;
                H1[pix1]++;
                if (max0 < pix0) max0 = pix0;
                if (max1 < pix1) max1 = pix1;
            }
            double mid0 = (double)S0 / max;
            double mid1 = (double)S1 / max;
            if (mid0 > mid1 && max0 > max1) Hi15 = true;
            else if (mid0 < mid1 && max0 < max1) Hi15 = false;
            Hi_OK = (Hi15 == file_Hi15);
            //if(!Hi_OK)
            Hi15 = file_Hi15;
```

```
            int MAX = max0 * 256 + 255;
            if (Hi15) MAX = max1 * 256 + 255;
            int noise = noise_0;// *MAX / 256;
            if (Hi15)
                for (int n = n0; n < max; n += 2)
                {
                    int dpix = (Data[n + 1] * 256 + Data[n]) * 256 / MAX;
                    if (dpix > 255) dpix = 255;
                    HH[dpix]++;
                }
            else
                for (int n = n0; n < max; n += 2)
                {
                    int dpix = (Data[n] * 256 + Data[n + 1]) * 256 / MAX;
                    if (dpix > 255) dpix = 255;
                    HH[dpix]++;
                }
            int S = 0;
            int M = 0;
            for (int n = noise; n < 255; n++)
            {
                M += HH[n];
                S += HH[n] * n;
            }
            if (M < 100)
            {
                noise = 0;
                S = 0;
                M = 0;
                for (int n = 0; n < 255; n++)
                {
                    M += HH[n];
                    S += HH[n] * n;
                }

            }
            if (M == 0) M = 1;
            mid = (double)S / M;
            double DS = 0;
            for (int n = noise; n < 255; n++)
            {
                double d = n - mid;
                DS += HH[n] * d * d;
            }
            sko = Math.Sqrt(DS / M);
            mid = mid * MAX / 256;
            sko = sko * MAX / 256;

        }
        else
        {
        }
    }

}
```

ДОДАТОК В

СЕРВЕРНИЙ БІК (ОБРОБНИК ДАНИХ ФАЙЛУ ФОРМАТУ DICOM)

```
namespace PrepareDicomData
{

    public partial class Form1 : Form
    {
        //*****************************************
        bool VisibleMode = true;                // !!!
        bool BlockCommandDoubling=false;        //
        //*****************************************

        Dicom dcm=new Dicom();
        const int Position_max = 64000;
        String MyDir="c:\\tmp";
        String Commands="";
        String CommandsOld="";
        String[] CoomandsListArray;
        String[] CommandFields;
        String[] CommandDataArray;
        char[] sepCommands = { '^' };
        char[] sepData = { '#' };
        char[] sepElements = { '|' };

        public Form1()...
        private void GetAll_16_Click(object sender, EventArgs e)...
        private void GetAll_8_Click(object sender, EventArgs e)...
        private void WP_Plus_Click(object sender, EventArgs e)...
        private void WP_Minus_Click(object sender, EventArgs e)...
        private void WS_Plus_Click(object sender, EventArgs e)...
        private void WS_Minus_Click(object sender, EventArgs e)...

        bool isVisible = false;
        private void timer1_Tick(object sender, EventArgs e)...
        private void Form1_Shown(object sender, EventArgs e)  //Hide form...


        void DoCommands()
        {
            if (Commands == "SHOW#")
            {
                Commands = "";
                if (VisibleMode) Show();
                WindowState = System.Windows.Forms.FormWindowState.Normal;
            }
            CoomandsListArray = Commands.Split(sepCommands);
            CommandsList.Clear();
            Commands = "";
```

```
            if (VisibleMode) Show();
            WindowState = System.Windows.Forms.FormWindowState.Normal;
        }
        CoomandsListArray = Commands.Split(sepCommands);
        CommandsList.Clear();
        Commands = "";

        foreach (String S in CoomandsListArray)
        try {  CommandsList.AppendText(S+"\r\n");    }
        catch (Exception) { }

        foreach (String S in CoomandsListArray)
        try
        {
            CommandFields = S.Split(sepData);
            if (CommandFields.Length == 2)
            {
                dcm.OkMsg = "OK";
                switch (CommandFields[0])
                {
                    case "DIR":
                         dcm.OpenDir(MyDir + CommandFields[1]);
                         break;
                    case "FILE":
                         CommandDataArray = CommandFields[1].Split(sepElements);
                         if (CommandDataArray.Length==2)
                         dcm.OpenFile(CommandDataArray[0], CommandDataArray[1]);
                         break;
                    case "TALL":
                         dcm.PrepareAllTagList(dcm.TxtResultFileName);
                        break;
                    case "WPP":
                        dcm.DataToBmp_WP_Plus(CommandFields[1]);
                        break;
                    case "WPM":
                        dcm.DataToBmp_WP_Minus(CommandFields[1]);
                        break;
                    case "WSP":
                        dcm.DataToBmp_WS_Plus(CommandFields[1]);
                        break;
                    case "WSM":
                        dcm.DataToBmp_WS_Minus(CommandFields[1]);
                        break;
                    case "NXT":
                        dcm.NextInMultiFrame(CommandFields[1]);
                        break;

                   case "PRV":
                       dcm.PrevInMultiFrame(CommandFields[1]);
                       Close();                                    //for test
                       break;
                   case "STOP":
                       Close();
                       break;
               }

            }
            dcm.WriteOK();
        }
        catch (Exception) { }
        CoomandsListArray = new String[0];
    }
```

ДОДАТОК Г

КЛІЄНТСЬКИЙ БІК (JSON-ДОСТУП ДО ДАНИХ У ФОРМАТІ DICOM)

```
function selectDir(text) {
  let xhr = new XMLHttpRequest();
  let url = "select_dir.php";

  xhr.open("POST", url, true);
  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

  xhr.onreadystatechange = function() {
    if (xhr.readyState === XMLHttpRequest.DONE) {
      if (xhr.status === 200) {
        let responseData = xhr.responseText;
        let dirContent = JSON.parse(responseData);
        let data = document.getElementById("data");
        data.innerHTML = "<div id='dir' class='directory unselectable'></div)";
        let dir = document.getElementById("dir");
        let i = 0;
        while (dirContent[i]) {
          let imgName = dirContent[i].substring(0, (dirContent[i].length - 4));// + "jpg";
          dir.innerHTML += '<p onclick="getImage(\''+dirContent[i]+'\',\''+imgName+'\')">' + dirConte
          i++;
        }
      } else {
        console.error("ERROR:", xhr.status);
      }
    }
  }

  xhr.send("text=" + encodeURIComponent(text));
}

function getImage (dcm, jpg) {
  let text = "FILE#"+dcm+'|'+jpg+'.jpg';

  let xhr = new XMLHttpRequest();
  let url = "get_image.php";

  xhr.open("POST", url, true);
  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

  let imageIndex=document.getElementById("imageIndex").value;
  let dataInsides = '<div class="display">'+
  '<div id="imageContainer"></div>'+
  '<div id="metaData"></div>'+
'</div>'+

  '<div class="board">'+
      '<div class="pis">'+
          '<button onclick="changeData(\'WPM#\', \''+jpg+'\')"> WinPis - </button>'+
          '<button onclick="changeData(\'WPP#\', \''+jpg+'\')"> WinPis + </button>'+
      '</div>'+

      '<div class="size">'+
          '<button onclick="changeData(\'WSM#\', \''+jpg+'\')"> WinSize - </button>'+
          '<button onclick="changeData(\'WSP#\', \''+jpg+'\')"> WinSize + </button>'+
      '</div>'+

      '<div class="get">'+
          '<button onclick="changeData(\'TALL#\')"> Get All Tags </button>'+
      '</div>'+
  '</div>'+
'</div>';
```

```
  xhr.onreadystatechange = function() {
    if (xhr.readyState === XMLHttpRequest.DONE) {
      if (xhr.status === 200) {
        let responseData = xhr.responseText;
        let data = document.getElementById("data");
        data.innerHTML = dataInsides;
        let imageContainer = document.getElementById("imageContainer");
        let metaData = document.getElementById("metaData");
        metaData.innerHTML = responseData;
        imageContainer.innerHTML = "<img src='DCM\\_DICOM_DATA\\" + jpg + "'>";
      } else {
        console.error("ERROR:", xhr.status);
      }
    }
  };

  xhr.send("text=" + encodeURIComponent(text));
}

function changeData(cmd, jpg=0) {
  let text;
  let newImage;
  let imgToDel;
  if (!jpg) text = cmd;
  else {
    let imageIndex=document.getElementById("imageIndex").value;
    imgToDel = jpg + '_' + imageIndex + '.jpg';
    document.getElementById("imageIndex").value++;
    text = cmd + jpg+ '_' + imageIndex + '.jpg';
    newImage = jpg + '_' + imageIndex + '.jpg';
  }
  let xhr = new XMLHttpRequest();
  let url = "change_data.php";

  var params = 'text=' + encodeURIComponent(text) + '&imgToDel=' + encodeURIComponent(imgToDel);

  xhr.open("POST", url, true);
  xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

  xhr.onreadystatechange = function() {
    if (xhr.readyState === XMLHttpRequest.DONE) {
      if (xhr.status === 200) {
        let responseData = xhr.responseText;
        let imageContainer = document.getElementById("imageContainer");
        let metaData = document.getElementById("metaData");
        if (text == "TALL#")
          metaData.innerHTML = responseData;
        else if (text != "TALL#")
        imageContainer.innerHTML = "<img src='DCM\\_DICOM_DATA\\"+newImage+"'>";
      } else {
        console.error("ERROR:", xhr.status);
      }
    }
  };

  xhr.send(params);
}
```