Міністерство освіти та науки України

Національний університет “Львівська політехніка”

КАФЕДРА ЕОМ

****

**Дипломна робота**

на тему: “**Мобільний термінал системи передачі медичних зображень.**”

Виконав:

ст.гр. СПРм-21 Сопушинський М. Б.

Перевірив:

асист. Морозов Ю. В.

Львів – 2014 р.

**ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Розробити мобільний термінал системи передачі медичних зображень, який використовуватиметься для підключення до існуючих медичних систем та перегляду медичних зображень.

Розробити програмну реалізацію мобільного терміналу.

Користувач повинен могти переглядати медичні зображення в кількох режимах.

Користувач повинен могти підєднюватись до існуючої системи для завантаження та перегляду медичних зображень.

**АНОТАЦІЯ**

Магістерська кваліфікаційна робота присвячена проведенюю дослідження та розробці програмної реалізації мобільного терміналу системи передачі медичних зображень. Також розглянуто способи побудови медичних систем з використанням мобільних терміналів.

В даній роботі розглянуті питання, пов'язані з покращенням функціонування медичних інформаційних систем. Проведено техніко-економічне обґрунтування проекту і описані питання, що стосуються безпеки медичною інформації.

**ABSTRACT**

The Master's qualification work is devoted to the research and development of software implementation of the mobile terminal of medical images trnsfering systems. Also there were considered the methods of constructing medical systems using mobile terminals.

In this paper there were considered issues related to improving the functioning of medical information systems. Also there were carried out economical analyzing of project and considered issues related to medical information secure.

**ЗМІСТ**

[Найти всі скороченняВСТУП 7](#_Toc403493758)

[ВСТУП 8](#_Toc403493759)

[1. АНАЛІТИЧНИЙ ОГЛЯД 11](#_Toc403493760)

[1.1. Аналіз структури медичної системи 11](#_Toc403493761)

[1.2. Огляд функціональних можливостей сучасних медичних інформаційних систем 13](#_Toc403493762)

[1.2.3. CS PoliBase 13](#_Toc403493763)

[1.2.2. Доктор Eleks 14](#_Toc403493764)

[1.2.3. Eleks Avalon 17](#_Toc403493765)

[1.3. Огляд сучасних медичних стандартів 19](#_Toc403493766)

[1.3.1 Огляд медичного стандарту HL7 19](#_Toc403493767)

[1.3.2 Огляд медичного стандарту DICOM 23](#_Toc403493768)

[1.4 Огляд законодавства у сфері медичної інформації 28](#_Toc403493769)

[1.5 Огляд технологій для забезпечення захисту медичної інформації 29](#_Toc403493770)

[1.5.1 Огляд технології VPN та протолу IPSec 29](#_Toc403493771)

[1.5.2 Огляд протоколу SSL 30](#_Toc403493772)

[Висновок 37](#_Toc403493773)

[2. ОБҐРУНТУВАННЯ ВИБРАНОГО НАПРЯМУ РОБОТИ 39](#_Toc403493774)

[2.1 Дослідження алгоритмів обробки первинного зображення 39](#_Toc403493775)

[Висновок 48](#_Toc403493776)

[3. ПРОЕКТНИЙ РОЗДІЛ 49](#_Toc403493777)

[3.2 Вибір медичного стандарту для медичних зображень 50](#_Toc403493778)

[3.3 Вибір технології для забезпечення безпеки даних 50](#_Toc403493779)

[3.4 Вибір платформи та інструментальних засобів 51](#_Toc403493780)

[3.5 Розробка мобільного терміналу 53](#_Toc403493781)

[3.2.1 Розробка діаграми варіантів використання 53](#_Toc403493782)

[3.2.2 Розробка діаграми класів 56](#_Toc403493783)

[3.2.2 Підсистема комунікації з МІС 60](#_Toc403493784)

[3.2.3 Підсистема зчитування та обробки медичних зображень 60](#_Toc403493785)

[3.2.3 Підсистема взаємодії з користувачем 61](#_Toc403493786)

[3.3 Художньо конструкторське оформлення мобільного терміналу 61](#_Toc403493787)

[Висновок 62](#_Toc403493788)

[4. ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ 63](#_Toc403493789)

[5. ЕКОНОМІЧНИЙ РОЗДІЛ 64](#_Toc403493790)

[4.1. Розрахунок витрат на розробку програмного забезпечення 64](#_Toc403493791)

[4.2. Вибір та обґрунтування аналогу 68](#_Toc403493792)

[4.3. Визначення експлуатаційних витрат 68](#_Toc403493793)

[4.4. Розрахунок ціни споживання проектного рішення 70](#_Toc403493794)

[4.5. Визначення показників економічної ефективності 71](#_Toc403493795)

[Висновки до економічного розділу 73](#_Toc403493796)

[ВИСНОВКИ 74](#_Toc403493797)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 75](#_Toc403493798)

[ДОДАТОК А 76](#_Toc403493799)

**ПЕРЕЛІК СКОРОЧЕНЬ**

DICOM – Digital Imaging and Communication in Medicine

HL7 – Health Level 7

PACS – Picture Archiving and Communication System

МІС – медична інформаційна система

VPN – virtual private network

SSL – secure socket layer

СКБД – система керування базою даних

УЗД – ультразвукова діагностика

ПЗ ЕКГ ОС API ООП

# Найти всі скороченняВСТУП

За кілька останніх десятиліть рівень застосування комп'ютерів в медицині надзвичайно підвищився. Практична медицина стає все більш і більш автоматизованою. Почало з’являтися багато програм та пристроїв з якими вони взаємодіють для забезпечення функціонування медичних установ.

Виділяють два види комп'ютерного забезпечення: програмне та апаратне. Програмне забезпечення включає в себе системне й прикладне.

Системне програмне забезпечення призначене для функціонування самого комп'ютера як єдиного цілого. Це, в першу чергу, операційна система, а також сервісні програми різного призначення - драйвери, утиліти і т. п. У системне програмне забезпечення входить мережевий інтерфейс, який забезпечує доступ до даних на сервері. Дані, введені в комп'ютер, організовані, як правило, в базу даних, яка, в свою чергу, управляється прикладною програмою системою керування базами даних (СКБД) і може містити, зокрема, історії хвороби, рентгенівські знімки, статистичну звітність зі стаціонару, бухгалтерський облік та іншу інформацію.

Нові вимоги в політиці охорони здоров'я, а також бурхливий розвиток комп'ютерних технологій ставлять перед розробниками програмного забезпечення завдання створення комплексних систем автоматизації діяльності медичних установ та їх оптимізацію. Розробка та впровадження таких систем дозволяє ефективно вирішувати завдання інтеграції всіх наявних джерел інформації як медичної, так і господарської орієнтації, полегшити роботу медичного персоналу. Це виражається в збільшенні швидкості обробки інформації різного типу, підвищення оперативності прийняття рішень. Результатом є зменшення ресурсів потрібних на обслуговування пацієнта.

Медична система збору та обробки інформації в сучасній лікарні повинна виконувати настільки багато різноманітних можливостей, що їх не можна навіть описати. Розробка та впровадження таких систем дозволяє ефективно вирішувати завдання інтеграції всіх наявних джерел інформації як медичної, так і господарської орієнтації, полегшити роботу медичного персоналу. Це виражається в збільшенні швидкості обробки інформації різного типу, підвищення оперативності прийняття рішень.

Основними проблемами при аналізі предметної області є розуміння роботи медичних інформаційних систем, їх внутрішньої структури та способів взаємодії з ними. Також необхідно розуміння специфічних медичних протоколів та стандартів, оскільки вони є доволі специфічними для такої галузі як медицина, тому їх використання необхідно враховувати при розробці програмного забезпечення для медичних систем. Іншою стороною проблеми є забезпечення конфіденційності даних, які передаються в таких системах та відповідність їх вимогам законодавства конкретної країни.

Згідно до статті 32 Конституції України, яка гарантує конфіденційність особистої інформації про людину, статтею 286 Цивільного Кодексу України, яка гарантує кожному право на право на таємницю про стан свого здоров’я, факт звернення за медичною допомогою, діагноз, а також відомості, одержані при медичному обстеженні. Згідно зі ст. 39-1 Основ законодавства України про охорону здоров’я пацієнт має право на таємницю про стан свого здоров’я, факт звернення за медичною допомогою, діагноз, а також про відомості, одержані при її медичному обстеженні. Крім того, згідно зі ст. 7 Закону України «Про захист персональних даних» до обробки персональних даних про здоров’я людини пред’являються особливі вимоги.

Згідно вищезгаданого законодавства предмет лікарської таємниці складають:

* стан здоров’я пацієнта;
* хвороби і діагноз;
* огляд і його результати;
* методи лікування;
* відомості, отримані при медичному обстеженні.

Тому мідичні інформаційні системи повинні забезпечувати належний рівень захисту відомостей про пацієнтів.

На сьогоднішній час існує багато таких систем як вітчизняного так і закордонного виробництва. Наприклад:

* CS PoliBase - виробник «Сайбернетик Системс», Росія;
* Доктор Елекс - виробник «Елекс», Україна;
* Eleks Avalon - виробник «Елекс», Україна.

Зараз жодна з них не впроваджена на рівні держави, що робить важливими дослідження в даній предметній області. Через це існує великий інтерес на ринку програмного забезпечення до таких систем. Тобто зараз є актуальним пошук рішень для покращення роботи таких систем.

Хоча єдиної системи на рівні держави немає у лікарнях вже є багато обладнання, яке підтримує основні стандарти галузі. Таке обладнання генерує велику кількість медичних зображень, які необхідно певним чином переглядати. Для цього є два способи: друк таких зображень або перегляд первинних зображень за допомогою спеціального програмного забезпечення. Друк не є надійним способом. Використання друкованих документів від часу стає причиною помилок при фіксуванні даних пацієнтів, проблем зі зберіганням і впорядкуванням результатів аналізів та різноманітних медичних досліджень (рентгенографія, томографія, УЗД тощо) і, як результат, може призвести до непередбачуваних наслідків.

Тому рішення, що мною було запропоновано – це розробка мобільного терміналу системи передачі медичних зображень, що дозволить уникнути вище вказаних проблем з друком. Це дозволить лікарю напряму працювати з первинним зображенням, а також дасть зручний інтерфейс для роботи з різними медичними документами, що, в свою чергу, підвищить точність діагностування, зменшить час потрібний для обслуговування пацієнта, призведе до зменшення навантаження на приймальні медичного закладу.

# 1. АНАЛІТИЧНИЙ ОГЛЯД

Для кращого розуміння функціональних можливостей сучасних медично-інформаційних систем потрібно розглянути та проаналізувати їх загальну структуру. Також необхідно розглянути основні стандарти галузі для кращого розуміння роботи існуючих систем, їх переваг та недоліків.

# 1.1. **Аналіз структури медичної системи**

Як і будь-яка складна система, медична система лікарні буде складатися з багатьох складових частин. Вона буде містити як апаратну так і програмні частини. До апаратної частини такої системи повинно входити обладнання необхідне для проведення медичних маніпуляцій, мережне обладнання, сервер, що забезпечує роботу системи, клієнтські машини, необхідні для роботи з документами, медичними даними пацієнтів і т. д. До програмної частини буде входити програмне забезпечення медичного сервера для забезпечення комунікації між клієнтами мережі, роботи бази даних, драйвери для роботи з медичним обладнанням, принтерами а також клієнтське ПЗ на комп’ютерах лікарів, призначене для перегляду даних медичних пристроїв, історій хвороби пацієнтів, доступу та керування медичним сервером і т. д. Також окремо слід виділяти ПЗ для забезпечення безпеки системи. На рис. 1.1 наведено структурну схему медичної системи.



Рис. 1.1 Структурна схема медичної системи.

Як видно з рис. 1. система повинна включати в себе багато взаємодіючих пристроїв, а також багато функціональних можливостей, які необхідно реалізувати. Тому розробка таких систем була стандартизована, оскільки вони повинні поширюватися не на окремий заклад, а на тисячі закладів країни, багатьох виробників обладнання і т. д. На сьогоднішній день існує декілька найбільш розповсюджених стандартів та багато, які розповсюджені не так широко. Слід зазначити, що не існує якогось головного універсального медичного стандарту. Різні стандарти часто реалізовують різноманітні сторони такої величезної області як медицина, тому спершу потрібно визначити конкретну проблему, яку слід вирішити, а потім підбирати, який стандарт найбільше підходить для її вирішення. Також постає питання про доцільність розробки такої системи з нуля чи використання вже існуючих систем і їх поліпшення. Виходячи з цього, спершу необхідно провести опис медичних стандартів та розробок і визначити сферу застосування кожного з них. Нижче приведений опис таких стандартів та список їх функціональних можливостей.

1.2. Огляд функціональних можливостей сучасних медичних інформаційних систем

Для кращого розуміння функціональних можливостей сучасних інформаційно-аналітичних систем, нижче наводиться декілька готових розробок як українських так і закордонних виробників, з коротким описом їх функціональних можливостей.

# 1.2.3. CS PoliBase

Медична інформаційна система (МІС) CS PoliBase є ефективним програмним засобом комплексної автоматизації медичних закладів, клінік загального профілю, поліклінік, косметологічних центрів, стоматологічних клінік, стаціонарів, санаторіїв, лабораторій, відділень швидкої допомоги, відділень або кабінетів приватних лікарів. Система є масштабною і дозволяє автоматизувати багатопрофільні медичні установи, які мають у своєму складі різні відділення, підрозділи та служби.

Ця інформаційна система забезпечує комплексну автоматизацію роботи медичного закладу: облік пацієнтів і лікарського персоналу, ведення розкладу роботи та запису на прийом, облік діагнозів, облік медичних послуг, облік медикаментів, розрахунок вартості послуг і облік оплати, розрахунок зарплати лікарського персоналу, облік ефективності реклами, а також володіє потужним і зручним механізмом статистичної обробки інформації та підготовки будь-яких звітів про роботу лікарів, реєстраторів, відділень і клініки в цілому.

CS PoliBase підтримує ієрархічну структуру підприємства будь-якого рівня вкладеності: Всі основні дані і довідники системи настроюються у відповідності з ієрархічною структурою підприємства. Механізм управління доступом дозволяє гнучко налаштовувати робочі місця і права користувачів по їх приналежності до підрозділу підприємства.

Даний продукт реалізує оригінальну методику оцінку ефективності роботи медичного закладу, лікарів і реєстраторів за об'єктивними показниками, підтримує роботу з рентгенівськими знімками і фотографіями і дозволяє реєструвати, зберігати й обробляти практично необмежений обсяг інформації про лікування пацієнтів і роботі клініки, забезпечуючи при цьому швидкий доступ до будь-якої необхідної інформації при дотриманні прав доступу і вимог безпеки.

# 1.2.2. Доктор Eleks

Системні показники

Спеціалізація. Медична інформаційна система, розроблена для автоматизації роботи сучасних медичних закладів, незалежно від їх розміру і спеціалізації

Системні вимоги. Вимоги до системи (робоче місце ):

Комп'ютер : і386 сумісний процесор з тактовою частотою 1 ГГц, 256 Мб оперативної пам'яті, роздільна здатність монітора 800x600, підключення до мережі.

Безпека, конфіденційність даних. Існують облікові записи користувачів і груп користувачів, можливе управління правами доступу до документів.

Статистичні звіти.

Підсистема статистичних вибірок.

Підсистема статистичних вибірок дає можливість кожному лікарю одержати і проаналізувати потрібну інформацію про пацієнтів, процес лікування, частоту надходження хворих з певним діагнозом, кількості витрачених медикаментів і т. д. Підсистема статистичних вибірок дозволяє користувачу за допомогою спеціального редактора побудувати фразу, що описує необхідну інформацію. Далі побудована фраза транслюється в запит до бази даних і користувач отримує необхідні дані

Підсистема складання звітів.

Для отримання роздруків виписок, направлень, звітів в систему вбудований редактор звітів. Він дозволяє створення шаблону звіту, який пізніше можна буде використовувати для виводу конкретної інформації. Редактор звітів дозволяє створювати шаблони для генерації звітів на різних мовах.

Після завершення лікування лікар проводить завершальний огляд пацієнта і виписує його.

При цьому він створює новий документ – виписку. Вона містить короткий опис всіх оглядів, аналізів, процедур, а також рекомендації щодо подальшого оздоровлення, профілактики, режиму роботи і дієти. При формуванні рекомендацій система Avalon дає можливість лікарю враховувати різні аспекти і обставини.

Архітектура (підтримка модульної /компонентності, тип моделі клієнт-сервер (якщо використовується ). Підсистеми (модулі):

- реєстратура - прийом, облік пацієнтів, прийом на лікування, поселення пацієнтів

- робоче місце лікаря - спрощує пошук інформації про пацієнта, введення даних огляду та створення для пацієнта будь-яких документів. Окрім того, система дозволяє вводити дані лабораторних досліджень, записувати відео і зображення безпосередньо з медичного устаткування, редагувати ключові кадри та включати їх у документи пацієнтів.

- підсистема складання звітів - технологія побудови звітів базується на аналізі кількості документів, створених за певним типом шаблонів, а також на основі аналізу певних вузлів шаблону, згідно заданих критеріїв генерації звітів. Кожній спеціальності відповідає свій тип звіту.- підсистема формування рекомендацій

- видача рекомендацій з урахуванням різних аспектів : діагнозу, віку пацієнта, протипоказань і т.д.

- підсистема PACS - автоматизована телемедична радіологічна інформаційна система, що є PACS-системою (Picture Archiving and Communication System), забезпечує автоматизацію процесу вводу, обробки, збереження та передачі результатів досліджень через канали зв’язку (а саме, діагностичних зображень, протоколів досліджень, тощо)

- лабораторія - дає лікарю-лаборанту можливість швидко і легко вводити дані лабораторних досліджень у документи пацієнта. У випадках відхилення від норми інформація виділяється візуально.

- підсистема обробки графічних зображень та відео - аналіз і обробка одержаних зображень в результаті проведення УЗД, флюорографії, ЕКГ.

- адміністрування - підсистема створення і настройки призначеного для користувача інтерфейсу і управління системою

Платформа (сервери, СКБД, ОС, мови програмування ). СКБД : MS SQL. Операційна система (на АРМ ): Microsoft Windows 2000

Відкритість системи. Підтримка стандартів мед. інформатики, інтеграція з зовнішнім медичним ПЗ і МІС, інтеграція з мед. устаткуванням і лаб. системами . Наявність відкритого API (підтримка OLE, DCOM, інтеграція /зв'язок із зовнішнім ПЗ ). -

Робота з графічними даними. Після проведення флюорографії або УЗД одержаний результат у вигляді графічного зображення потрібно проаналізувати або додатково обробити . Система пропонує наступні можливості :

* зберегти лише необхідну частину зображення;
* використовувати спеціальні медичні позначки;
* додавати текст в зображення;
* малювати по зображенню, відмічати певні області.

Можливості налаштування системи (на установу, на підрозділ, на користувача). Налаштування АРМ користувачів. Наявність шаблонів звітів, виписок.

Віддалений доступ. Система Доктор Елекс може використовувати Web-інтерфейс

Планування, управління ресурсами (ліжка, розклад лікарів, устаткування і т.д.). Підсистема розкладів програми Доктор Елекс передбачає індивідуальне планування робочого дня кожного працівника, враховуючи побажання лікарів та пацієнтів. Для зручності користувачів розклади відображаються за день, тиждень чи місяць.

# 1.2.3. Eleks Avalon

Системні показники

Спеціалізація. Eleks Avalon автоматизує роботу персоналу основних підрозділів медичних установ.

Системні вимоги. Вимоги до системи (робоче місце ):

Комп'ютер : Pentium 4 1000 Мгц, 128 Мб оперативної пам'яті, роздільна здатність монітора 800x600, підключення до мережі.

Безпека, конфіденційність даних. Існують облікові записи користувачів і груп користувачів, можливе управління правами доступу до документів.

Підсистема статистичних вибірок дає можливість кожному лікарю одержати і проаналізувати потрібну інформацію про пацієнтів, процес лікування, частоту надходження хворих з певним діагнозом, кількості витрачених медикаментів і т. д. Підсистема статистичних вибірок дозволяє користувачу за допомогою спеціального редактора побудувати фразу, що описує необхідну інформацію. Далі побудована фраза транслюється в запит до бази даних і користувач отримує необхідні дані

Для отримання роздрукованих виписок, направлень, звітів в систему вбудований редактор звітів. Він дозволяє створення шаблону звіту, який пізніше можна буде використовувати для виводу конкретної інформації. Редактор звітів дозволяє створювати шаблони для генерації звітів на різних мовах. Також система Avalon дозволяє створювати звіти в Microsoft Excel97.

Після завершення лікування лікар проводить завершальний огляд пацієнта і виписує його.

При цьому він створює новий документ – виписку. Вона містить короткий опис всіх оглядів, аналізів, процедур, а також рекомендації щодо подальшого оздоровлення, профілактики, режиму роботи і дієти. При формуванні рекомендацій система Avalon дає можливість лікарю враховувати різні аспекти і обставини.

Архітектура (підтримка модульної /компонентности, тип моделі клієнт-сервер (якщо використовується ). Підсистеми (модулі):

- реєстратура - прийом, облік пацієнтів, прийом на лікування, поселення пацієнтів

- робоче місце лікаря - проведення оглядів пацієнта і ведення картки пацієнтів

- підсистема складання розкладів - складання графіків відвідин пацієнтами лікарів з урахуванням певних обмежень

- підсистема статистичних вибірок - аналіз інформації з бази даних

- підсистема складання звітів - генерація обхідних листів, виписок і різноманітних призначених для користувача звітів за допомогою вбудованого редактора

- підсистема формування рекомендацій - видача рекомендацій з урахуванням різних аспектів : діагнозу, віку пацієнта, протипоказань і т.д.

- робота з медичною апаратурою - обмін даними з сучасними медичними апаратами різних фірм

- підсистема обробки графічних зображень - аналіз і обробка одержаних зображень в результаті проведення УЗД, флюорографії, ЕКГ.

- адміністрування - підсистема створення і настройки призначеного для користувача інтерфейсу і управління системою

Платформа (сервери, СУБД, ОС, мови програмування ). СУБД : Oracle 7/8, Interbase. Операційна система (на АРМ ): Microsoft Windows 95/98 або Windows NT 4.0 (SP3)

Відкритість системи. Підтримка стандартів мед. інформатики, інтеграція з зовнішнім медичним ПЗ і МІС, інтеграція з мед. устаткуванням і лаб. системами . Наявність відкритого API (підтримка OLE, DCOM, інтеграція /зв'язок із зовнішнім ПЗ ). У Eleks Avalon передбачена робота з різними медичними пристроями виробництва фірм Aloka, Siemens, Humolyser.

Робота з графічними даними. Після проведення флюорографії або УЗД одержаний результат у вигляді графічного зображення потрібно проаналізувати або додатково обробити . Eleks Avalon пропонує наступні можливості :

- зміна масштабу (цілого знімка або його частини ), контрасту, яскравості і т.д.;

- фарбування в псевдокольори для виділення окремих деталей;

- отримання негативу зображення;

- згладжування деяких деталей за допомогою різних графічних фільтрів;

- градуювання;

- денситометрія (визначення густини різних крапок);

- морфологія (вимірювання лінійних розмірів деталей знімка).

Шаблони документів. Існують шаблони звітів, виписок.

Можливості налаштування системи. Налаштування АРМ користувачів. Наявність шаблонів звітів, виписок.

Планування, управління ресурсами (ліжка, розклад лікарів, устаткування і т.д.). Підсистема складання розкладів. Складання розкладів процедур і оглядів пацієнтів. При складанні розкладів можна враховувати різноманітні аспекти (час роботи лікарів, кабінетів, їх перерви; спеціалізацію лікарів і кабінетів; призначення процедур групам пацієнтів; місткість кабінетів ). Для зручності в складанні розкладів підсистема забезпечена безліччю сервісних функцій, що дозволяють вносити коректування вручну.

# 1.3. Огляд сучасних медичних стандартів

# 1.3.1 Огляд медичного стандарту HL7

HL7, Health Level 7 («Сьомий рівень») - стандарт обміну, управління та інтеграції електронної медичної інформації.

«Сьомий рівень» - аналогія з вищим рівнем комунікаційної моделі відкритих систем (OSI). Сьомий рівень підтримує виконання таких завдань як:

* структурування переданих даних
* можливості проектування систем
* досягнення узгодженості передач
* безпека
* ідентифікація учасників
* доступність

Розробкою, розвитком, впровадженням, накопиченням медичних знань займається потужна безприбуткова добровольча організація Health Level Seven, Inc, створена в 1987 році, штаб-квартира розташована в місті Енн Арбор шт. Мічиган (Ann Arbor, MI).

В США стратегічна медична ініціатива реалізується з кінця 70-х років в наступних програмах:

* UMLS (довідники в широкому сенсі слова - віртуальний світ медицини)
* МИС В.А. Vista (повна реалізація всіх контурів управління лікарнею - Управління у справах ветеранів МО)
* HL7 - однойменна корпорація.

Основні складові частини стандарту HL7 зображено на рис. 1.2



Рис. 1.2 Основні складові частини HL7

**RIM** (Reference Information Model, Еталонна Інформаційна Модель) Ключовий елемент ідеології HL7. RIM - інформаційна модель медицини - основне джерело змісту даних усіх HL7- повідомлень і документів. Елементи інформаційної моделі - класи, переходи станів класів, типи даних і накладені обмеження - використовуючи системні концепції та графічне вираження UML.

Типи інформаційних моделей:

**USAM** - Unified Service Action Model - загальна модель службових дій - об'єктна модель всіх клінічних послуг-дій, частина RIM. Дія має модуси (mood) - дефініцію, цілепокладання (план, намір), порядок виконання в заданому контексті, критерії виконання, спеціалізовані модуси. «Охорона здоров'я - послідовність дій виконаних для блага пацієнта». Крім інформаційної моделі є також моделі повідомлень MIM (Message Information Model) і контекстно-прив'язана модель R-MIM (Refined Message Information Model).

**StoryBoard** - функціональна модель - в термінах системного проектування, UML. Концепція розкадровки (storyboard) взята з кіноіндустрії і дозволяє представити засобами HL7 значущі моменти передачі повідомлень як кадри. У кожному кадрі описані ключові учасники та їх взаємодія. Комплект кадрів являє як передачу повідомлення, так і функціонування великої системи. Опис роботи тригерів (тригер) запускаючих подій (наприклад форма після заповнення переходить в стан "заповнена" і / або "підписана"). Кожна взаємодія описується розкадровкою (в UML діаграма послідовностей).

**Vocabulary** – словники, значення словника концепція предметної області, а не слово або код (ідеологія UMLS - словник є тезаурусом, онтологією) Атрибут в RIM-описі може бути елементом словника.

Словники можуть бути:

* Багатоколонкова, побудована на принципах метатезауруса UMLS таблиця описана засобами HL7
* LOINC, SNOMED, HIPAA, місцеві, національні словники.

**HMD** – Hierarchical Message Descriptor - визначник ієрархічної структури повідомлення.

Принципи HMD:

* система передачі повинна розуміти генезис класів.
* повідомлення при передачі шикуються в лінійну структуровану послідовність.

**CDA** Архітектура Клінічного документа (АКД, CDA, Clinical Document Architecture) Стандарт сфери HL7, затверджений ISO (ISO / HL7 27932: 2009 Data Exchange Standards - HL7 Clinical Architecture Document, Release 2). В АКД визначений синтаксис і комплекс структур (база) для повного вираження семантики клінічного документа. АКД використовує мову розмітки інформаційних об'єктів XML.

Специфікація клінічного документа створюється на основі довідника даних RIM - іншими словами, сенс клінічного документа при машинній обробці виходить з RIM. CDA визначає розмітку (розмітки) клінічного документа, його структуру і семантику. Клінічний документ по CDA є повним інформаційним об'єктом, з повністю визначеними компонентами. Він може містити текст, зображення, звук та інший мультимедійний зміст.

HL7 почав розвиватися як стандарт повідомлень. Клінічний документ може бути переданий в повідомленні, або існувати незалежно. Природа документа та повідомлення різні. Повідомлення: тимчасово, запускається зовнішньою подією, існує певний період часу, має адресата, найчастіше непризначений для читання людиною. Документ: стабільний, повний, має авторство, визначення правил доступу, призначений для перегляду людиною.

**EHR** System (Electronic Health Record Systems - Система Електронної історії хвороби (див. ГОСТ Р 52636-2006)).

Опис повного функціоналу EHR що складається з розділів Управління наданням медичної допомоги (Care Management), Клінічний документообіг (Clinical Support), Інформаційна інфраструктура (Information Infrastructure) - всього 125 функцій.

**Арден синтаксис** (Arden Syntax) Специфікація прийнята HL7 для визначення та розповсюдження медичних знань. Арден синтаксис є мовою Медичних Логічних Модулів (Medical Logic Modules) кодування медичних знань. Кожен модуль містить достатню інформацію для прийняття медичного рішення. МЛМ використовується для генерації сигналів тривоги, розуміння медичних даних, діагностики, фільтрації медичних даних і адміністративних завдань. За певних умов може бути розроблена комп’ютерна програма (монітор подій), яка генерує експертну підтримку. МЛМ може бути пов'язаний з іншими МЛМ та утворювати мережу. Основний виразний засіб HL7 - мова графічної концептуалізації систем UML.

# 1.3.2 Огляд медичного стандарту DICOM

DICOM (англ. Digital Imaging and Communications in Medicine) – Галузевий стандарт створення, зберігання, передачі та візуалізації медичних зображень і документів обстежених пацієнтів.

DICOM спирається на ISO-стандарт OSI, підтримується основними виробниками медичного обладнання та медичного програмного забезпечення.

Стандарт DICOM, розроблений Національною асоціацією виробників електронного устаткування (National Electrical Manufacturers Association), дозволяє створювати, зберігати, передавати і друкувати окремі кадри зображення, серії кадрів, інформацію про пацієнта, дослідження, обладнання, установах, медичному персоналі, що виробляє обстеження, і т. п.

Стандартом DICOM визначено два інформаційних рівня:

* файловий рівень - DICOM File (DICOM-файл) - об'єктний файл з теговою організацією для подання кадру зображення (або серії кадрів) та супроводжуючої/керуючої інформації (у вигляді DICOM тегів);
* мережевий (Комунікаційний) - DICOM Network Protocol (мережевий DICOM-протокол) - для передачі DICOM файлів і керуючих DICOM команд по мережах з підтримкою TCP/IP.

DICOM File являє собою об'єктно-орієнтований файл з теговвою організацією. Інформаційна модель стандарту DICOM для DICOM файла чотириступінчаста (рис 1.3):

пацієнт (patient) → дослідження (study) → серія (series) → зображення (кадр або серія кадрів) (image).

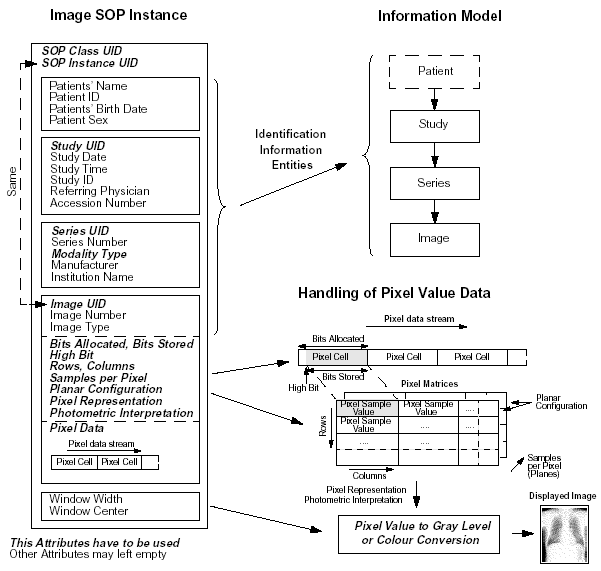


Рис. 1.3 Інформаційна модель DICOM

Файловий рівень стандарту DICOM 3.0 редакції 2008 описує:

1. Атрибути і демографічні дані пацієнта.
2. Модель і фірму виробника апарату, на якому проводилося обстеження.
3. Атрибути медичного закладу, де було проведено обстеження.
4. Атрибути персоналу, що проводив обстеження пацієнта.
5. Вид обстеження і дата/час його проведення.
6. Умови і параметри проведення дослідження пацієнта.
7. Параметри зображення або серії зображень, записаних в DICOM-файлі.
8. Унікальні ключі ідентифікації унікального ідентифікатора (UID) груп даних, описаних в DICOM-файлі.
9. Зображення, серію або набір серій, отриманих при обстеженні пацієнта.
10. Представлення, в першу чергу, PDF-документів в DICOM-файлі.
11. Представлення DICOM-записи на оптичні носії, включаючи DVD формат.
12. DICOM-протокол для передачі/прийому по мережах TCP/IP комп’ютерним.

DICOM Network Protocol використовує TCP/IP для передачі медичної інформації від медичного обладнання в PACS-систему (Picture Archiving and Communication System) і для зв'язку між PACS-системами. Протокол трирівневий (рис. 1.4) - нижній, відразу над TCP - DUL (DICOM upper level); над ним - сервіси: DIMSE (DICOM message protocol) і ACSE (Association Control protocol — standard OSI protocol); і вище DICOM Application Interface. Над ними розташоване додаток - Medical Imaging Application.

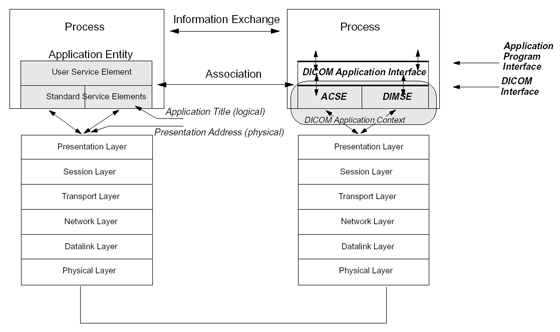


Рис. 1.4 Представлення рівнів DICOM- і TCP/IP-протоколів

Стандарт DICOM дозволяє виробляти інтеграцію медичного обладнання різних виробників, включаючи DICOM-сканери, DICOM-сервери, автоматизовані робочі місця і DICOM принтери в єдину радіологічну або клінічну інформаційну систему (англ. Hospital information system).

Стандарт DICOM включає в себе ряд мережевих (основних) сервісів:

1. DICOM Store (Storage Service Class) - запам'ятовування (збереження) зображень та іншої інформації.
2. DICOM Query/Retrieve (Query/Retrieve Service Class) - запит/отримання списку пацієнтів та/або досліджень з іншого DICOM пристрої.
3. DICOM Media Store (Media Storage Service Class) - збереження даних на носіях інформації для обміну даними.
4. DICOM SCP (Service Class Provider) - реалізує роль сервера в DICOM-мережі.
5. DICOM SCU (Service Class User) - реалізує роль клієнта в DICOM-мережі.
6. DICOM Modality Worklist (Basic Worklist Management - єдиний не нормалізований Service Class) - «Робочий Лист Досліджень» - список необхідних для пацієнтів досліджень, який може бути отриманий запитом користувача до RIS-системі.
7. DICOM Print (Print Management Service Class) - DICOM-друк, на спеціалізованих DICOM-принтерах (плівкових високого дозволу або повнокольорових), що працюють по DICOM-протоколу.

Стандарт DICOM включає в себе основні мережеві команди, кожна з яких здійснює як запит (запит) - в основному відправляє «клієнт» (Service Class User, SCU), так і відповідь (реакція) - в основному відповідає «сервер» (Service Class Provider, SCP):

Echo – перевіряє наявність DICOM-з'єднання між двома DICOM-пристроями;

Find – здійснює пошук DICOM-елементів та/або DICOM-файлів пацієнтів на вибраному DICOM-влаштуванні;

Get – зчитує DICOM-елементи пацієнтів з обраного DICOM-пристрої;

Set – встановлює DICOM-елементи на вибраному DICOM-влаштуванні;

Store – зберігає DICOM-елементи та/або DICOM-файли на вибраному DICOM-влаштуванні;

Move – копіює (переносить) DICOM-елементи та/або DICOM-файли пацієнтів з одного DICOM-пристрою на інший.

Мережевий DICOM Протокол використовує TCP/IP для передачі медичної інформації від медичного обладнання в PACS (Picture Archiving and Communication System) систему.

PACS являє собою клієнт/серверну медичну інформаційну систему, що складається з взаємозв’язаних компонентів (рис. 1.5):

* Медичного DICOM обладнання, які є DICOM клієнтами,
* Одного (обов'язково) або декількох серверів DICOM,
* Однією або кілька (бажано) діагностичних (Робочих) DICOM Станцій,
* Одного або декількох DICOM принтерів (не обов'язково).



Рис. 1.5 Структура PACS системи

Стандарт DICOM (DICOM Протокол) забезпечує цифровий зв'язок між DICOM Серверами і діагностичним обладнанням різних виробників, включаючи:

* Рентгенологічну апаратуру,
* Ангіографічні комплекси,
* Комп'ютерна томографія,
* Магнітно-резонансна томографія,
* Ізотопні сканери,
* Ультразвукові сканери,
* Мікроскопи, ендоскопи і т.п.
* Робочі DICOM станції.

# 1.4 Огляд законодавства у сфері медичної інформації

Згідно до статті 32 Конституції України, яка гарантує конфіденційність особистої інформації про людину, статтею 286 Цивільного Кодексу України, яка гарантує кожному право на право на таємницю про стан свого здоров’я, факт звернення за медичною допомогою, діагноз, а також відомості, одержані при медичному обстеженні. Згідно зі ст. 39-1 Основ законодавства України про охорону здоров’я пацієнт має право на таємницю про стан свого здоров’я, факт звернення за медичною допомогою, діагноз, а також про відомості, одержані при її медичному обстеженні. Крім того, згідно зі ст. 7 Закону України «Про захист персональних даних» до обробки персональних даних про здоров’я людини пред’являються особливі вимоги.

Згідно вищезгаданого законодавства предмет лікарської таємниці складають:

* стан здоров’я пацієнта;
* хвороби і діагноз;
* огляд і його результати;
* методи лікування;
* відомості, отримані при медичному обстеженні.

# 1.5 Огляд технологій для забезпечення захисту медичної інформації

# 1.5.1 Огляд технології VPN та протолу IPSec

Згідно законодавства України результати такі результати медичного огляду пацієнта, як рентгенографія, томографія, УЗД, тощо є предметом лікарської таємниці, тому повинні бути належним чином захищені. В умовах відкритих мереж такий захист можна забезпечити використовуючи технологію віртуальних приватних мереж.

VPN (Віртуальна приватна мережа, англ. Virtual Private Network) — це логічна мережа, створена поверх інших мереж, на базі загальнодоступних або віртуальних каналів інших мереж (Інтернет). Безпека передавання пакетів через загальнодоступні мережі може реалізуватися за допомогою шифрування, внаслідок чого створюється закритий для сторонніх канал обміну інформацією. VPN дозволяє об'єднати, наприклад, декілька географічно віддалених мереж організації в єдину мережу з використанням для зв'язку між ними непідконтрольних каналів.

VPN складається з двох частин: «внутрішня» (підконтрольна) мережа, яких може бути декілька, і «зовнішня» мережа, через яку проходять інкапсульовані з'єднання (зазвичай використовується Інтернет).

Підключення до VPN віддаленого користувача робиться за допомогою сервера доступу, який підключений як до внутрішньої, так і до зовнішньої (загальнодоступною) мережі. При підключенні віддаленого користувача (або при установці з'єднання з іншою захищеною мережею) сервер доступу вимагає проходження процесу ідентифікації, а потім процесу аутентифікації. Після успішного проходження обох процесів, віддалений користувач (віддалена мережа) наділяється повноваженнями для роботи в мережі, тобто відбувається процес авторизації.

IPsec (скорочення від IP Security) — протоколи для забезпечення захисту даних, що передаються за допомогою протоколу IP, дозволяє здійснювати підтвердження справжньості та/або шифрування IP-пакетів. IPsec також містить в собі протоколи для захищеного обміну ключами в мережі Інтернет.

Протоколи IPsec, на відміну від інших добре відомих протоколів SSL та TLS, працюють на мережевому рівні (рівень 3 моделі OSI). Це робить IPsec гнучкішим, так що він може використовуватися для захисту будь-яких протоколів, що базуються на TCP та UDP. IPsec може використовуватися для забезпечення безпеки між двома IP-вузлами, між двома шлюзами безпеки або між IP-вузлом і шлюзом безпеки. Протокол є "надбудовою" над IP-протоколом, і обробляє сформовані IP-пакети описаним нижче способом. IPsec може забезпечувати цілісність та / або конфіденційність даних переданих по мережі.

IPsec використовує наступні протоколи для виконання різних функцій:

* Authentication Header (АН) забезпечує цілісність віртуального з'єднання (переданих даних), аутентифікацію джерела інформації та додаткову функцію із запобігання повторної передачі пакетів
* Encapsulating Security Payload (ESP) може забезпечити конфіденційність (шифрування) переданої інформації, обмеження потоку конфіденційного трафіку. Крім цього, він може забезпечити цілісність віртуального з'єднання (переданих даних), аутентифікацію джерела інформації та додаткову функцію із запобігання повторної передачі пакетів (Всякий раз, коли застосовується ESP, в обов'язковому порядку повинен використовуватися той чи інший набір даних послуг із забезпечення безпеки)
* Security Association (SA) забезпечують зв'язку алгоритмів і даних, які надають параметри, необхідні для роботи AH і / або ESP. Internet security association and key management protocol (ISAKMP) забезпечує основу для аутентифікації і обміну ключами, перевірки автентичності ключів.

# 1.5.2 Огляд протоколу SSL

SSL (англ. Secure Sockets Layer — рівень захищених сокетів) — криптографічний протокол, який забезпечує встановлення безпечного з'єднання між клієнтом і сервером. SSL спочатку розроблений компанією Netscape Communications . Згодом на підставі протоколу SSL 3.0 був розроблений і прийнятий стандарт RFC, що отримав ім'я TLS.



Рис. 1.6 Схема роботи SSL.

На рис. 1.6 наведено схему роботи протоколу SSL. Перед обміном повідомленнями необхідно провести обмін сертифікатами, які містять відкриті ключі клієнта і сервера. Протокол забезпечує конфіденційність обміну даними між клієнтом і сервером, що використовують TCP/IP, причому для шифрування використовується асиметричний алгоритм з відкритим ключем. При шифруванні з відкритим ключем використовується два ключі, причому будь-який з них може використовуватися для шифрування повідомлення. Тим самим, якщо використовується один ключ для шифрування, то відповідно для розшифровки потрібно використовувати інший ключ. У такій ситуації можна отримувати захищені повідомлення, публікуючи відкритий ключ, і зберігаючи в таємниці секретний ключ.

В протоколі SSL визначено два терміни: з’єднання (connection) –це пристосування яке забезпечує сервіс деякого типу. Зєднання – є тимчасовими і існують в межах сеансу.

Сеанс (session) – зв'язок між клієнтом і сервером. Створюється за допомогою протоколу рукостискання і визначення набору параметрів криптографічного захисту.

Типовий час зєднання 10 хв., а час сеансу 30 хв. Після чого вони скидуються і встановлюються знову.

Протокол SSL складається з двох підпротоколів: протокол SSL запису і рукостискання. Протокол SSL запису визначає формат, який використовується для передачі даних. Протокол SSL включає рукостискання з використанням протоколу SSL запису для обміну серіями повідомлень між сервером і клієнтом, під час встановлення першого з'єднання. Для роботи SSL потрібно, щоб на сервері був SSL- сертифікат.

SSL надає канал, що має три основні властивості:

* Аутентифікація. Сервер завжди автентифікований, в той час як клієнт автентифікований в залежності від алгоритму.
* Цілісність. Обмін повідомленнями включає в себе перевірку цілісності.
* Конфіденційність каналу. Шифрування використовується після встановлення з'єднання і використовується для всіх наступних повідомлень.

Взаємна автентифікація двох сторін в SSL відбувається шляхом обміну цифровими сертифікатами відкритих ключів користувачів (клієнта і сервера) завірених цифровим підписом спеціальних сертифікаційних центрів. SSL підтримує сертифікати стандарту Х.509, а також стандарти інфраструктури відкритих ключів PKI, з допомогою якої організована видіча і перевірка сертифікатів. Автентифікація відбувається за допомогою протоколу рукостискання (рис. 1.7)

SSL підтримує 3 типи аутентифікації:

* Аутентифікація обох сторін (клієнт — сервер),
* Аутентифікація сервера з нерозпізнаних клієнтом
* Повна анонімність.



Рис 1.7. Взаємна автентифікація

Кожного разу, коли сервер автентифіковані, канал безпечний проти спроби перехоплення даних між веб-сервером і браузером, але повністю анонімна сесія за своєю суттю вразлива до такої атаки. Анонімний сервер не може автентифікувати клієнта. Якщо сервер аутентифікований, то його повідомлення сертифікації має забезпечити вірну сертифікаційну ланцюжок, що веде до прийнятного центру сертифікації. Простіше кажучи, аутентіфіцированний клієнт повинен надати допустимий сертифікат сервера. Кожна сторона відповідає за перевірку того, що сертифікат іншого боку ще не закінчився і не був скасований.

SSL клієнт і сервер домовляються про встановлення зв'язку за допомогою процедури рукостискання (рис. 1.8).

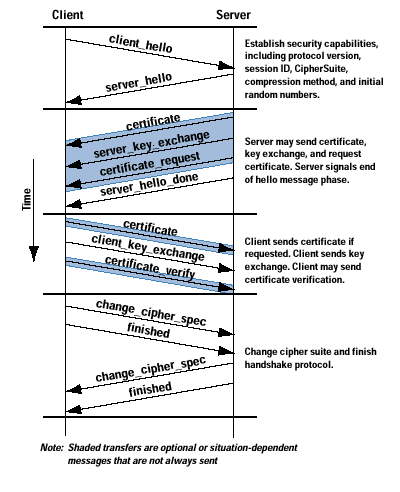


Рис. 1.8 Протокол рукостискання

Під час рукостискання клієнт і сервер домовляються про різні параметри, які будуть використані, щоб забезпечити безпеку з'єднання.

* Рукостискання починається тоді, коли клієнт підключається до SSL сервера. Запит безпечного з'єднання являє собою список підтримуваних шифрів та хеш-функцій.
* З цього списку сервер вибирає найсильніший шифр та хеш-функцію, яку він також підтримує, і повідомляє клієнтів про прийняте рішення.
* Сервер відсилає це рішення у вигляді цифрового сертифікату. Сертифікат, звичайно, містить ім'я сервера, довірений Центр Сертифікації, і відкритий ключ шифрування сервера. Клієнт може зв'язатися з сервером, який видав сертифікат і переконатися, що сертифікат є справжнім, перш ніж продовжити.
* Для того, щоб згенерувати ключі сеансу, використовується безпечне з'єднання. Клієнт шифрує випадкове число за допомогою відкритого ключа (ВК) сервера і відправляє результат на сервер. Тільки сервер в змозі розшифрувати його (з його закритим ключем (ЗК)), і тільки цей факт робить ключі прихованими від третьої сторони, так як тільки сервер і клієнт мали доступ до цих даних. Клієнт знає відкритий ключ і випадкове число, а сервер знає закритий ключ і (після розшифровки повідомлення клієнта) випадкове число. Третя сторона, можливо, знає тільки відкритий ключ, якщо закритий ключ не був зламаний.
* З випадкового числа обидві сторони створюють ключові дані для шифрування та розшифрування.

На цьому рукостискання завершується, і починається захищене з'єднання, яке зашифровується і розшифровується за допомогою ключових даних. Якщо будь-що з перерахованих вище дій не вдається, то рукостискання SSL не вдалося, і з'єднання не створюється.

Головна мета процесу обміну ключами — це створення секрету клієнта (pre\_master\_secret), відомого тільки клієнту і серверу. Секрет (pre\_master\_secret) використовується для створення спільної таємниці (master\_secret). Загальний секрет необхідний для того щоб створити повідомлення для перевірки сертифіката, ключів шифрування, секрету MAC (message authentication code) і повідомлення «finished». При посилці вірного повідомлення «finished», тим самим сторони доведуть що вони знають вірний секрет (pre\_master\_secret).

Конфіденційність забезпечується шифруванням передаваних повідомлень з використанням симетричних сесійних ключів якимим сторони обмінюються при встановленні зєднання. Сесійні ключі передаються також в зашифрованому вигляді, при цьому вони шифрують ся за допомогою відкритих ключів, які витягуються з сертифікату абонента. Використання симетричних ключів пов’язано з більшою швидкість роботи симетричних алгоритмів шифрування ніж асиметричних.

Достовірність інформації забезпечується за допомогою формування і перевірки цифрового підпису. Для цифрового підпису і обміну ключами використовується алгоритми з відкритим ключем.

В рамках протоколу SSL визначені чотири види шифрування:

* digitally-signed (шифрування електронного підпису)
* stream-ciphered (потокове шифрування)
* block-ciphered (блочне шифрування)
* publick-key-encrypted (шифрування за допомогою відкритого ключа)

Сесія може бути встановлена при використанні алгоритму RSA або Діффі-Хеллмана для створення ключів обміну. У разі використання RSA клієнт шифрує секрет (pre\_master\_secret) за допомогою відкритого ключа несертифікованого сервера. Відкритий ключ клієнт дізнається з повідомлення обміну ключами від сервера. Результат надсилається в повідомленні обміну ключами від клієнта. При використанні алгоритму Діффі-Хеллмана відкриті параметри сервера містяться в повідомленні обміну ключами від сервера, і клієнтові посилають в повідомленні обміну ключами. Перехоплювач, який не знає приватних значень, не зможе знайти секрет (pre\_master\_secret).

* Для обміну ключами та перевірки їх достовірності застосовуються: RSA, Diffie-Hellman, ECDH, SRP, PSK.
* Для аутентифікації: RSA, DSA, ECDSA.
* Для симетричного шифрування: RC2, RC4, IDEA, DES, Triple DES або AES, Camellia.
* Для хеш-функцій: SHA, MD5, MD4 і MD2.

Для роботи SSL потрібно установлення SSL сертифікату. SSL сертифікат – це цифровий підпис доменного імені.

Цифрові сертифікати в основному служать двом цілям:

* встановити особу власника;
* зробити доступним первинний ключ власника.

Цифровий сертифікат випускається перевіреної повноважною організацією - джерелом сертифікатів (certificate authority - CA) і видається тільки на обмежений час. Після закінчення терміну дії сертифіката його необхідно замінити. Протокол SSL використовує цифрові сертифікати для обміну ключами, аутентифікації серверів і, при необхідності, аутентифікації клієнтів.

Цифровий сертифікат містить наступні фрагменти інформації про особу власника сертифіката та джерелі сертифікатів:

* повне (унікальне) ім'я власника сертифіката;
* публічний ключ власника;
* дата видачі цифрового сертифіката;
* дата закінчення дії сертифіката;
* повне (унікальне) ім'я видавця (джерела сертифіката);
* цифровий підпис видавця.

Протокол SSL використовує сертифікати для перевірки з'єднання. Ці SSL-сертифікати розташовані на безпечній сервері і служать для шифрування даних та ідентифікації Web-сайту. Сертифікат SSL допомагає підтвердити, що сайт дійсно належить тому, хто це заявляє, і містить інформацію про держателя сертифіката, домену, для якого був виданий сертифікат, і назва джерела (CA), що видала сертифікат.

Є три способи отримати SSL-сертифікат:

1. використовувати сертифікат від джерела сертифікатів;
2. використовувати самоподпісанний сертифікат;
3. використовувати "порожній" сертифікат

# Висновок

Розробка та дослідження медичних інформаційних систем є дуже важливою. Зараз жодна з них не впроваджена на рівні держави, тому дослідження в даній області є актуальними. Через це існує великий інтерес на ринку програмного забезпечення до таких систем. Тобто зараз є актуальним пошук рішень для покращення роботи таких систем.

До складу мідичних інформаційх систем входять різні компоненти : мережна інфраструктура, системи передачі та зберігання медичних зображень, .медичне обладнання лікарні, медичні станції для діагностики, бази даних тощо.

Діагностичними станціями є найчастіше спеціальне програмне забезпечення для персональних комп'ютерів і ноутбуків, що забезпечує зв'язок їх з мережною та програмною інфраструктурою лікарні.

При аналізі існуючих медичних інформаційних систем було знайдено ряд недоліків:

* пацієнт повинен багато часу витрачати в черзі на друк нових знімків, в цей час лікар не може приймати його для встановлення діагнозу;
* лікар мусить постійно перебувати в кабінеті де є встановлено комп’ютер з ПЗ для перегляду медичних зображень;
* існуючим МІС не вистачає мобільного терміналу для перегляду зображень та встановлення діагнозу.

В наступних розділах буде запропоновано методи вирішення цих недоліків.

# 2. ОБҐРУНТУВАННЯ ВИБРАНОГО НАПРЯМУ РОБОТИ

ОПИСАТИ ЧОМУ ПОТРІБЕН ТЕРМІНАЛ ЙОГО Х-КИ

До основних функціональних можливостей повинно входити можливість додаткового аналізу та обробки графічних даних. Для цього мобільний термінал повинен мати наступні функції:

* зміна масштабу зображень;
* зміна контрасту;
* зміна яскравості;
* фарбування зображення в псевдокольори для виділення окремих деталей;
* отримання негативу зображення;
* перегляд метаінформації про пацієнта, дату дослідження, тип процедури, медичний прилад і т. д.

Такі функціональні можливості необхідні для лікаря при встановленню діагнозу і перегляду зображень оскільки дозволяють змінювати зображення на льоту, при цьому можливо побачити різні деталі зображень котрі непомітні при простому виводі зображення на екран чи друку зображення.

# 2.1 Дослідження алгоритмів обробки первинного зображення

Класичні реалізації алгоритмів обробки зображень передбачають прохід по елементах зображення пошук певних величин, які характеризують зображення, наприклад гістограм, та зміну елементів зображення залежно для подальшої обробки чи відображення. По такому самому принципу будуть працювати алгоритми зміни контрасту, яскравості, та фарбування зображенн в псевдокольори. Тобто базовий алгоритм для всіх вихідних алгоритмів буде складатись з проходження елементів зображення та відповідної обробки кожного елементу (рис. 2.1)



Рис 2.1 Базовий алгоритм обробки зображення

Далі необхідно вибрати спосіб обробки елементів зображення для різних цілей. Наприклад для отрамання негативу просто необхідно просто замінити значення елементу на його доповнення до найбільш можливого значення

Yn = Ymax – Y, (2.1)

Для пришвидшення роботи з зображенням потрібно розглянути як зберігається елементи зображення в памяті. Для ОС Android зображення представляється у вигляді пікселів розміром по 3 байта, по одному байту для кожної компоненти кольору. Так можна використати операцію інверсії замість віднімання для пришвидшення роботи програми, оскільки однією операцією буде оброблятися один елемент зображення, без розбиття на окремі компоненти для кожного кольору. Тому отримання негативу зображення буде виглядати наступним чином:

Yn = !Y, (2.2)

Згідно цієї формули можна побудувати алгоритм отримання негативу зображення (рис. 2.3)



Рис 2.2 Алгоритм отримання негативу зображення

Контраст — міра виявлення об'єкта на якому-небудь тлі. Розрахувати контраст можна, якщо відома яскравість об'єкта і фону, на якому ми спостерігаємо об'єкт. Правильний розрахунок контрасту необхідний для отримання гармонійного розподілу яскравості. Це дуже важливо, оскільки баланс яскравостей справляє сильний вплив на сприйняття освітлюваного об'єкта. Різкий контраст справляє неприємне враження і приводить до зорової втоми, оскільки оку доводиться постійно пристосовуватися до різних рівнів яскравості. Недостатній контраст також небажаний, оскільки в цьому випадку об'єкти виглядають плоскими і погано розрізняються на фоні, що також приводить до зорового стомлення, тому необхідно розробити алгоритм зміни контрасту. Для обчислення елементу зображення по заданому значенню контрасту використовується наступна формула:

Yn = ((Yn / Ymax) - 0.5) \* C) + 0.5) \* Ymax), (2.3)

Згідно цієї формули можна побудувати алгоритм зміни контрасту зображення (рис. 2.3)



Рис 2.3 Алгоритм зміни контрасту

При обчисленні нового значення, слід також нормалізувати його, тобто перевірити чи нове значення елементу зображення не виходить за межі максимальних.

Яскравість є атрибутом зорового сприйняття, який з'являється, коли є джерело світла та об’єкт, що відбиває світло. Іншими словами, яскравість сприйняття визначається яскравістю об’єкта що відбиває світло, і є світловою характеристикою об’єктів. В колірній системі RGB яскравість визначається середнім арифметичним з червоного, зеленого і синього кольору елемента зображення. Для зміни яскравості необхідно змінити всі три компоненти кольору. Відповідно зміна яскравості буде визначатися за формулою 4. При чорнобілому зображенні значення всіх трьох компонент RGB буде однакове, тому можна взяти одну з них як початкове значення яскравості.

Yn = Y + B, (2.4)

Згідно цієї формули можна побудувати алгоритм зміни яскравості зображення (рис. 2.4)



Рис 2.3 Алгоритм зміни яскавості

Розфарбовування псевдокольорами потрібне для того, щоб виділити різні елементи зображення залежно від їхньої яскравості, це дозволить краще розрізняти різні деталі зображення, оскільки в людини краще розвинений кольоровий зір. Для розфарбовування зображення псевдокольорами необхідно виконати інший підхід, ніж для зміни контрасту чи яскравості. Для цього слід обрати метод розфарбовування і кольори для яких буде відбуватись розфарбовування. В мобільному терміналі реалізовано триколірне розфарбовування, згідно до значення яскравості компонента зображення. Для цього використовується колірна модель HSB, яка описує кольоровий простір, заснований на трьох характеристиках кольору: кольоровому тоні (Hue), насиченості (Saturation) і яскравості (Brightness). Шкала відтінків зображена на рис. 2.5

220px-HueScale

Рис 2.5 Шкала відтінків Hue

Значення насиченості кольору слід взяти рівним одиниці оскільки тоді кольори будуть найбільш насиченими. Яскравість та кольоровий тон необхідно взяти з значення яскравості для вихідного зображення. Оскільки характеристики моделі HSB приймають інші максимальні значення ніж в моделі RGB, необхідно обчислювати їх пропорційно. Для переведення характеристик HSB в RGB слід використати формули перетворення (Для будь-яких відтінків *H* ∈ [0°, 360°), насиченість *S* ∈ [0, 100], і яскравість *B* ∈ [0, 100]:):

, (2.5)

, (2.6)

, (2.7)

, (2.8)

, (2.9)

Після обчислення слід скористатись таблицею 2.1 для переведення обчислених величин в RGB значення. Також необхідно домножити отримані RGB значення на 2.55 оскільки вони обчисленні в процентному відношені.

Таблиця 2.1 Переведення HSB в RGB

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***R*** | ***G*** | ***B*** |
| *0* |  |  |  |
| *1* |  |  |  |
| *2* |  |  |  |
| *3* |  |  |  |
| *4* |  |  |  |
| *5* |  |  |  |

Як видно з формул їх можна спростити, так як значення насиченості кольору рівне одиниці, тобто 100%.

, (2.10)

, (2.11)

, (2.12)

, (2.13)

, (2.14)

Таблиця для переведення HSB в RGB тоді прийме інший вигляд (таблиця 2.2)

Таблиця 2.2 Спрощене переведення HSB в RGB

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***R*** | ***G*** | ***B*** |
| *0* |  |  | *0* |
| *1* |  |  | *0* |
| *2* | *0* |  |  |
| *3* | *0* |  |  |
| *4* |  | *0* |  |
| *5* |  | *0* |  |

Згідно спрощених формул можна побудувати алгоритм розфарбовування компонентів вихідного зображення (рис 2.6).



Рис 2.6 Агроритм розфарбовки зображення в псевдокольори

# Висновок

Написати висновок

# 3. ПРОЕКТНИЙ РОЗДІЛ

**3.1. Вибір технології програмування**

Об’єктно-орiєнтоване програмування (ООП) визначається як технологія створення складного програмного забезпечення, заснована на поданні програми у вигляді сукупності об'єктів, кожний з яких є екземпляром певного типу (класу), а класи утворять ієрархію зі спадкуванням властивостей. Взаємодія програмних об'єктів у такій системі здійснюється шляхом передачі повідомлень.

Основною перевагою об’єктно-орiєнтоване програмування в порівнянні з модульним програмуванням є "більше природна" декомпозиція програмного забезпечення, що істотно полегшує його розробку. Це приводить до більше повної локалізації даних й інтегруванню їх з підпрограмами обробки, що дозволяє вести практично незалежну розробку окремих частин (об'єктів) програми. Крім цього, об'єктний підхід пропонує нові способи організації програм, засновані на механізмах спадкування, поліморфізму, композиції, наповнення. Ці механізми дозволяють конструювати складні об'єкти з порівняно простих. У результаті істотно збільшується показник повторного використання кодів і з'являється можливість створення бібліотек класів для різних застосувань.

Бурхливий розвиток технологій програмування, заснованих на об'єктному підході, дозволило вирішити багато роблем. Так були створені середовища, що підтримують візуальне програмування, наприклад, Visual Stdudio, NetBeans і т.д. При використанні візуального середовища в програміста з'являється можливість проектувати деяку частину, наприклад, інтерфейси майбутнього продукту, із застосуванням візуальних засобів додавання й настроювання спеціальних бібліотечних компонентів. Результатом візуального проектування є заготівля майбутньої програми, у яку вже внесені відповідні коди.

Вся програма умовно може бути розділена на дві частини: основну й допоміжну. В основній частині виробляється найпростіша обробка інформації, організується звертання до різних підпрограм. Допоміжний алгоритм теж може викликати інші допоміжні, довжина такого ланцюжка викликів теоретично не обмежена. Допоміжними й основними алгоритми є не самі по собі, а по відношенню один до одного.

При використанні допоміжних алгоритмів необхідно враховувати спосіб передачі значень вихідних даних для них й одержання результату від них. Аргументи допоміжного алгоритму - це змінні, у яких повинні бути поміщені вихідні дані для рішення відповідної підзадачі. Результати допоміжного алгоритму - це також змінні, де втримуються результати рішення цих підзадач, а також результатом може бути конкретна дія, що робить комп'ютер під дією підпрограми.

Підпрограми можуть бути двох видів: підпрограма без параметрів і підпрограма з параметрами. Звертання до підпрограми може бути організоване з будь-якого місця основної програми або іншої підпрограми скільки завгодно раз.

# 3.2 Вибір медичного стандарту для медичних зображень

Неважко помітити подібність структури стандартів DICOM і HL7. Однак між ними є істотна відмінність: DICOM є повним стандартом обміну інформацією, пов'язаною з зображеннями, а HL7 – хоча і дозволяє працювати з зображеннями, є стандартом електронної передачі медичних документів між віддаленими медичними установами. Більшість сучасного обладнання підтримує стандарт DICOM, як основний стандарт для роботи з медичними зображеннями, тому стандарт DICOM краще підходить для використання в мобільному терміналі системи передачі медичних зображень.

# 3.3 Вибір технології для забезпечення безпеки даних

З оглянутих технологій забезпечення безпеки було виділено технології VPN та SSL. Обидві вони призначені для забезпечення безпечного з’єднання проте між ними є відмінності. В SSL, код який керує з’єднаннями повинен сам забезпечувати безпеку з’єднання; з точки зору програмування, це не те саме що відкрити SSL так ніби це просто сокет. З допомогою деяких бібліотек зробити це відносно просто, але все-таки, управління безпекою повинне виконуватись на рівні додатків. VPN, з іншого боку, налаштований на рівні операційної системи, так безпека забезпечується не між додатком на стороні клієнта і додатком на стороні сервера, а між клієнтською операційною системою та серверною операційною системою: це не така ж сама модуль безпеки, хоча в багатьох ситуаціях різниця в використанні не значна.

# 3.4 Вибір платформи та інструментальних засобів

В якості середовища розробки було вибрано Android Studio версії 0.8.6. Насамперед, коли розробник перший раз запускає середовище розробки, він зустрічається з її користувальницьким інтерфейсом, новим й, іноді, інтуїтивно незрозумілим. Недосвідчений, початківець розробник часто губиться в різноманітті вікон, іконок, кнопок і панелей інструментів, що помітно загальмовує освоєння середовища.

Android Studio побудоване на основі середовища IntelliJ Idea, тому інтерфейс є доволі подібним. IntelliJ Idea Community Edition є доволі потужним середовищем розробки для програм на мові Java. В Android Studio додано вбудовану підтримку розробки програм для ОС Android.

Крім цього є велика кількість іконок і картинок для того, щоб вони відповідали новим вимогам нових стилів IDE.

Інструменти Android Studio

Завдяки безлічі корисних інструментів Android Studio є самим зручним і потужним середовищем розробки програм. З кожною новою версією інструменти вдосконалюються, а рутинні завдання спрощуються, що значно прискорює процес розробки проектів. Але знову ж, на превеликий жаль, багато розробників не знають всіх можливостей наявних інструментів, намагаючись обходиться самотужки.

Вбудована підримка систем контролю версій

Системи контролю версій – це потужна система, яка має безліч функцій, головною з яких є збереження цілісності проекту та його контрольних точок, забезпечення збереження всіх змін, можливість відкату файлів до попередніх версій, перегляд історії, перегляд різниці між файлами тощо. Це дуже потужний засіб розробника програмного забезпечення. Також система контролю версій може бути зв’язана з зовнішнім репозиторієм і може робити резервні копії проектів на зовнішньому сервері. Є підтримка таких систем контролю версій, як Mercurial, Git, SVN.

Настроювання IDE

По ходу роботи розробник пристосовує IDE під свої власні потреби. І цей процес може бути досить тривалий і стомлюючим, якщо Ваші вимоги до середовища досить строгі, а Ваші звички не дають Вам довго "плазувати" по менюшкам у пошуках потрібних функцій. І як не хочеться повторювати цей процес знову й знову! Розробники Android Studio подбали про простий перенос всіх настроювань середовища між різними машинами. Тут є вбудована підтримка між переключенням настройок клавіатури з різних ОС чи інших середовищ розрбки

Налагодження

Налагодження істотно спростилося завдяки використанню вбудованого відлагоджувача прогам

Установивши точку зупинки або в ручну перервавши виконання додатка, можна насолодитися новими можливостями вікна Watch. Наприклад, можна писати багаторядкові частини коду для перевірки правильні певної умови.

Ще одна чудова можливість налагодження Android Studio - можливість установки точки зупинки (breakpoint) на виклик певної функції:

Редагування коду

Для того щоб вставити інформацію про параметри функції, що перебуває на спливаючій підказці, потрібно нажати комбінацію Alt+Sapce. У цьому випадку будуть вставлені параметри, які ще не задані для заданої функції. Якщо буде натиснута комбінація Alt+Enter, буде запропоновано можливий спосіб виправлення помилок. Також є різні способи виділення тексту, наприклад вертикальний коли потрібно видалити багато повторень в різних рядках. Комбінація Shift+F6 робить можливим переіменування змінної методу чи класу в усіх областях видимості де вони прсутні. Також тут є багато автоматичних методів рефакторингу та генерації коду.

# 3.5 Розробка мобільного терміналу

Мобільний термінал системи передачі медичних зображень повинен реалізовувати дві основні функції:

* мережна – включає в себе зв'язок з існуючою системою та можливість запиту списку файлів медичних зображень, завантаження та зберігання файлів;
* функції перегляду – влючає в себе можливість переглядати медичні зображення та метаінформацію про пацієнта;
* розширені функції перегляду – включає зміну контрасту, яскравості, мастабу, отримання негативу зображення, розфарбовування зображення псевдокольорами.

Для реалізації мобільного терміналу була вибрана платформа Android, так як вона є впровадженою на більшості мобільних пристроїв і містить широкий набір засобів для забезпечення безпеки передаваної інформації.

# 3.2.1 Розробка діаграми варіантів використання

Діаграма варіантів використання моделює взаємодію між зовнішніми клієнтами системи та варіантами використання системи. Кожен варіант використання представляє іншу можливість, що надає клієнту система.

Актори - це будь-які зовнішні одиниці, наприклад, люди, машини, чи інші системи, що використовують дану систему. Єдиним актором в нашій системі є особа, яка хоче скористатися терміналом.

Кожен варіант використання позначається овалом з надписом всередині. В нашій системі актор використовує ліфт лише з однією метою - переміститися на інший поверх.

Згідно визначеного функціоналу можна побувати наступні діаграми послідовностей (рис 3.1 – 3.3)



Рис 3.1 Діаграма використання функцій перегляду



Рис 3.2 Діаграма використання розширених функцій переглядача



Рис 3.3 Діаграма використання мережних функцій

# 3.2.2 Розробка діаграми класів

Діаграмою класів (Class diagram) називають діаграму, на якій показана множина класів, інтерфейсів, кооперацій і відносин між ними. Її зображають у вигляді множини вершин і дуг.

Звичайно діаграми класів використовуються в наступних цілях:

* для моделювання словника системи. Моделювання словника системи передбачає ухвалення розв'язання про те, які абстракції є частиною системи, а які - ні. За допомогою діаграм класів ви можете визначити ці абстракції і їхні обов'язки;
* для моделювання простих кооперацій. Кооперація - це сукупність класів, інтерфейсів і інших елементів, працюючих спільно для забезпечення деякої кооперативної поведінки, більш значущої, ніж сума становлячих його елементів. Наприклад, моделюючи семантику транзакцій в розподіленій системі, ви не зможете зрозуміти процеси, що відбуваються, дивлячись на один-єдиний клас, оскільки відповідна семантика забезпечується декількома спільно працюючими класами. За допомогою діаграм класів вдається візуалізувати і специфікувати ці класи і відносини між ними;
* для моделювання логічної схеми бази даних. Логічну схему можна уявляти собі як креслення концептуального проекту бази даних. В багатьох сферах діяльності вимагається зберігати стійку (persistent) інформацію в реляційній або об'єктно-орієнтованій базі даних. Моделювати схеми також можна за допомогою діаграм класів.

Діаграма класів дозволяє моделювати класи та зв’язки між ними. Кожен клас позначається прямокутником, що складається з трьох частин. В верхній частині вказують назву класу, в середній—атрибути класу, а в нижній—операції класу. UML дозволяє зображати лише верхню частину з назвою класу для покращення читабельності.

# 

Рис 3.4 Діаграма класів підсистеми зчитування та обробки зображень

# 

Рис 3.5 Діаграма класів підсистеми взаємодії з користувачем

Рис 3.6 Діаграма класів підсистеми взаємодії з МІС

# 3.2.2 Підсистема комунікації з МІС

ОПИСАТИ ВЗАЄМОДІЮ з МІС

# 3.2.3 Підсистема зчитування та обробки медичних зображень

Підсистема зчитування та обробки зображень є одним з основних компонентів ситеми . Вона відповідає за відкриття DICOM файлів, зчитування інформації про пацієнта та зчитування зображень. Також тут присутні методи для обробки зображень, зокрема для зміни контрасту, яскравості та розфарбовування зображення в псевдокольори. Для уникнення роботи з низькорівневими даними DICOM файлу було використано бібліотеку Imebra. Вона має методи для зчитування тегів DICOM файлу, а також методи для зчитування кадрів зображення. За зчитування відповідає клас DCMData, в якому є екземляри класу MetaData та ImageData, які відповідно містять дані зображень та метадані пацієнтів, а також методи для їх обробки.

Після зчитування кадри зписуються в список для подальшої вибірки та обробки в класі ImageData. Також підсистема містить буфер для зберігання та обробки поточного кадру. Після завантаження проводиться копіювання поточного кадру в буфер, далі проводиться обробка зображення з буфера. Зокрема обчислюється контраст, яскравість, при потребі може виконуватись взяття негативу зображення чи розфарбовування в псевкольори. Клас ImageData містить методи необхідні для цих функцій. Після цього відбувається перетворення даних зображення з буфера в клас Bitmap для подальшого відображення на інтерфейс користувача. Також клас MetaData містить метод для відображення метаданих зручному для перегляду форматі. Алгоритм роботи підсистеми зчитування та обробки медичних зображень зображено на рисунку 3.7



Рис 3.7 Алгоритм роботи підсистеми зчитування та обробки медичних зображень

# 3.2.3 Підсистема взаємодії з користувачем

Взаємодія з користувачем відбувається за допомогою стандартних компонентів Android додатку.

# 3.3 Художньо конструкторське оформлення мобільного терміналу

Перед будь-якою проектованою системою ставляться визначені вимоги в її художнім оформленні. Багато систем не набули широкого поширення саме через своє невдале художнє оформлення. В другу чергу рекомендується вибирати насичений колір для відображення написів, з іншої сторони дуже яскраві кольори досить швидко дратують очі, що унеможливлює довгий контакт користувача з програмою. Варто обмежитися постільними тонами в палітрі кольорів. Також слід звернути увагу на обмеження розмірів екрану мобільних пристроїв і те, що потрібнов ивводити інформацію про пацієнта, медичне зображення, а також контрольні кнопки для різноманітних функцій перегляду.

Іншою найбільш доладною проблемою є правильність вибору розміру і стилю шрифту. Людина дуже швидко утомлюється, якщо розмір шрифту в тексті програми дуже малий. Тим більше, якщо робота з даним додатком проводитися щодня. Однак занадто великий шрифт теж доставляє масу неприємностей. Він впадає в око і може привести до помилок у введенні інформації користувачем через свою здатність відволікати увагу. Варто використовувати розмір шрифту від 10 до 20.

Також варто врахувати і ту проблему, що на користувачів впливають характеристики пристрою відображення, тобто дисплей. Головною проблемою тут для багатьох проектів є вибір розширення дисплею при проектуванні системи. Адже, повне медичне зобрадження при великому розширення ніколи не поміститься в доступну для огляду область дисплею з меншою роздільною здатністю. На сьогоднішній день існує багато різних розширень дисплеїв для мобільних пристроїв. Для зручності користування додатком слід вибрати прстрій з великим розміром екрану та роздільною здатністю, оскільки на малому пристрої може бути непомітно деякі деталі зображень.

Для зручності користування додатком варто уникати великої вкладеності списків підменю. Оптимальне використання іконок із зображенням на них дії, яка відбудеться при натисканні на них. Даний прийом не тільки зручний практично, але і значно збагачує загальний інтерфейс програми.

Під час впровадження в системі інтерфейсної частини я використовував ще можливість Android для побулови власних графічних елементів. Таким чином побудовий діалог вибору файлів.

Як рішення для графічного оформлення системи було створено xml-файл в якому описано розміщення та властивості елементів графічної оболонки. При ініціалізації головної activity цей файл зчитується, після чого графічні елементи малюються на екрані. При цьому, також до елементів підключаються так звані прослуховувачі (Listeners) – класи, що реагують на певні зміни графічних елементів, наприклад на натискання. Це дозволяє будувати власні елементи керування чи змінювати їх поведінку. Наприклад для збільшення інформативності було реалізовано збільшення прозорості елемента з інформацією про пацієнта та зображення при натисканні на нього, для того щоб користувач краще бачив медичне зображення. З цією метою також було додано напрозорість бокових панелей.

# Висновок

В розділі було реалізовано мобільний термінал системи передачі медичних зображень. Для реалізації цієї системи вибрано платформу Android, мову програмування – Java. Інструментами розробки було вибрано Android Studio 0.8.6.

Розроблено UML діаграми використання, UML діаграми класів, UML діаграми послідовностей для всіх модулів.

В наступному розділі буде досліджено запропоновані методи та засоби моніторингу на їх актуальність і доцільність у використанні.

# 4. ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

# 

# 5. ЕКОНОМІЧНИЙ РОЗДІЛ

У даному розділі бакалаврської кваліфікаційної роботи я здійснив економічне обґрунтування доцільності розробки програмного забезпечення.

Розроблене програмне забезпечення призначене для пришвидшення детектування поверхні деталей та дефектів з комп’ютерних томограм металевих деталей і характеризується підвищеною швидкістю виконання алгоритму детектування поверхонь деталей та дефектів, що призводить до зменшення часу визначення характеристик деталей, а також до економії енергоспоживання обладнання.

Прогнозованим місцем використання є промислові підприємства важкої промисловості. Метою використання є пошук дефектів деталей, для відбору деталей з заданою міцністю.

# 4.1. Розрахунок витрат на розробку програмного забезпечення

Витрати на розробку і впровадження програмних засобів (*К*) включають:

*К* = *К1* + *К2*(4.1)

де *K1* - витрати на розробку програмних засобів, грн.

*К2* - витрати на відлагодження і дослідну експлуатацію програми рішення задачі на ЕОМ, грн.

Витрати на розробку програмних засобів включають:

* витрати на оплату праці розробників;
* витрати на відрахування у спеціальні державні фонди (*Вф*);
* витрати на куповані вироби (*Кв*);
* витрати на придбання спецобладнання для експериментальних робіт (*Об*);
* накладні витрати (*Н*);
* інші витрати (*Ів*).

Витрати на оплату праці розробників проекту визначаються за формулою:

(4.2)



де *nij* – чисельність розробників і-ої спеціальності j-ro тарифного

розряду, які приймають участь в проектуванні, чол.;

*tij* - час, який затрачений на розробку проекту співробітника і-ої

спеціальності j-ro тарифного розряду, днів;

*Cij* - денна заробітна плата і-ої спеціальності j-ro тарифного розряду,

грн.;

(4.3)



де Сij - основна місячна заробітна плата розробника *і*-ої спеціальності

*j*-гo тарифного розряду, грн.;

*h* - коефіцієнт, що визначає розмір додаткової заробітної плати;

*р* - середня кількість робочих днів у місяці.

Таблиця 4.1. Вихідні дані для розрахунку витрат на оплату праці

|  |  |  |  |
| --- | --- | --- | --- |
| № | Посада виконавців | Місячний оклад, грн. | Середньоденна ставка, грн./дні |
| 1 | Керівник дипломної роботи | 2600 | 123,81 |
| 2 | Консультант з економіки | 3200 | 152,38 |
| 4 | Студент | 720 | 34,29 |

*Середня кількість робочих днів у місяці: 21*

Таблиця 4.2. Розрахунок витрат на оплату праці

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Посада виконавців | Час розробки, ДНІ | Денна заробітна плата, грн. | Витрати на розробку, грн. |
| 1 | Керівник дипломної роботи | 4 | 123,81 | 495,24 |
| 2 | Консультант з економіки | 1/8 | 152,38 | 19,05 |
| 4 | Студент | 65 | 34,29 | 2228,85 |
| Разом |  |  |  | 2743,14 |

Величину відрахувань у спеціальні державні фонди визначають у процентному співвідношенні від суми основної та додаткової заробітної плати. Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 36,3%від суми заробітної плати:

*Вф* = 36,3 / 100 ⋅ *З*  (4.4)

*Вф* = 0,363 ⋅ 2743,14 = 995,76 грн.

Витрати на куповані вироби визначаються за фактичними цінами з врахуванням номенклатури та необхідної їх кількості на проект.

Таблиця 4.3. Розрахунок витрат на куповані вироби

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  п\п | Найменування Виробів | Марка, тип | Кількість, Штук | Ціна за одиницю, Грн. | Сума витрат, грн. |
| 1 | Компакт – диск  CD-R | Verbatim 700 MB | 1 | 1,50 | 1,50 |
| 2 | Папір | А4, 210x297 | 200 | 0,10 | 20 |
| Всього | | | 21,50 | | |

При розробці даного програмного забезпечення спеціальне обладнання не використовувалось, тому витрати на спеціальне обладнання відсутні.

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими процентами до витрат на оплату праці:

(4.5)



*Н* = 0,3 ⋅ 2743,14 = 822,94 грн.

Інші витрати відображають видатки, які не враховані в інших статтях витрат. Вони розраховуються за встановленими процентами до витрат на оплату праці:

*Ів* =10 / 100 ⋅ *З* (4.6)

*Ів* = 0,1 ⋅ 2743,14 = 274,31 грн.

Витрати на розробку програмного забезпечення розраховуються за формулою:

*К1* = *3* + *Вф* + *Кв* + *Об* + *Н* + *Ів* (4.7)

*К1* = 2743,14 + 995,76 + 21,50 + 0 + 822,94 + 274,31 = 4857,65 грн.

Витрати на відлагодження і дослідну експлуатацію програмного забезпечення визначаються за формулою:

*K2*= *SMг* ⋅ *tBiд*(4.8)

де *SMг* - вартість однієї машино-години роботи ЕОМ, грн./год.;

*tBід* - машинний час, витрачений на відлагодження і дослідну експлуатацію програмних засобів, год.

Загальна кількість днів роботи на ЕОМ рівна 65днів. Середній щоденний час роботи на ЕОМ - 2 год., тому:

*tвiд*= 65 ⋅ 2 = 130год.

За даними обчислювального центру НУ "Львівська Політехніка" для ЕОМ типу IBM PC/AT *SMг* = 2,50 грн. Отже:

*К2* = 2,50 ⋅ 130 = 325,00 грн.

Таблиця 4.4. Кошторис витрат на розробку програмного забезпечення

|  |  |  |
| --- | --- | --- |
| № | Найменування елементів витрат | Сума витрат, грн. |
| 1 | Витрати на оплату праці | 2743,14 |
| 2 | Відрахування у спеціальні державні фонди | 995,76 |
| 3 | Витрати на куповані вироби | 21,50 |
| 4 | Накладні витрати | 822,94 |
| 5 | Інші витрати | 274,31 |
| 6 | Витрати на відлагодження і дослідну експлуатацію програмного забезпечення | 325,00 |
|  | Всього | 5182,65 |

# 4.2. Вибір та обґрунтування аналогу

В якості аналогу виступає система TomoCAD. Система TomoCAD була розроблена в 2006 році компанією Intron Ltd. Основною задачею системи TomoCAD є автоматичний аналіз і обробка комп’ютерних томограм для детектування поверхонь металевих деталей і їх внутрішніх дефектів. Також дана система представляє можливості зберігання даних в форматах що можуть використовуватись іншим спеціалізованим програмним забезпеченням.

Дана система була вибрана в якості аналогу, оскільки в ній використовуються алгоритми, аналогічні до розглянутих в даній бакалаврській роботі. Оскільки алгоритми даної системи використовують тільки можливості центрального процесора, особливо актуально постає проблема прискорення даних алгоритмів використовуючи можливості графічних процесорів.

# 4.3. Визначення експлуатаційних витрат

Для оцінки економічної ефективності розроблюваного програмного продукту слід порівняти його з аналогом, тобто існуючим програмним забезпеченням ідентичного функціонального призначення.

Експлуатаційні одноразові витрати по програмному забезпеченню і аналогу включають вартість підготовки даних і вартість машино-годин

роботи ЕОМ (за час дії програми):

(4.9)



де *Еп* - одноразові експлуатаційні витрати на проектне рішення (аналог), грн.;

*Е1п* - вартість підготовки даних для експлуатації проектного рішення (аналогу), грн.;

*Е2п* - вартість машино-годин роботи ЕОМ для виконання проектного рішення (аналогу), грн.

Річні експлуатаційні витрати *Веп* визначаються за формулою:

(4.10)



де *Nп* - періодичність експлуатації проектного рішення (аналогу), раз/рік.

Вартість підготовки даних для роботи на ЕОМ визначається за формулою:

(4.11)



де 1 - номери категорій персоналу, який приймає участь у підготовці даних (1=1,2,...L);

*n1*, - чисельність співробітників 1-ої категорії, чол.;

*t1*, - трудоємність роботи співробітників 1-ої категорії по підготовці

даних, год.;

*с1* — середнього динна ставка співробітника 1-ої категорії з врахуванням додаткової заробітної плати та відрахувань у спеціальні державні фонди, грн./год.

(4.12)



де - основна місячна заробітна плата працівника 1-ої категорії, грн.; b - коефіцієнт, який враховує додаткову заробітну плату і відрахування у спеціальні державні фонди;



Для роботи з даними як для проектного рішення так і аналогу потрібен один працівник, основна місячна заробітна плата якого складає: с° = 1600 грн. Тоді:

грн./год



Трудоємність працівника по підготовці даних для проектного рішення складає 2 год., для аналога 3 год.

Таблиця 4.5. Розрахунок витрат на підготовку даних та реалізацію проектного рішення на ЕОМ

|  |  |  |  |
| --- | --- | --- | --- |
| № | Час роботи співробітників, год. | Середньогодинна заробітна плата, грн./год. | Витрати , грн. |
|  | Проектне рішення | | |
| 1 | 2 | 14,95 | 29,9 |
|  | Аналог | | |
| 1 | 3 | 14,95 | 44,85 |

Витрати на експлуатацію ЕОМ визначається за формулою:

(4.13)



де *t* - витрати машинного часу для реалізації проектного рішення (аналогу), год.;

*SMГ* - вартість однієї машино-години роботи ЕОМ, грн./год.

*Е2п* = 2 ⋅ 2,5 = 5 грн.

*Е2а*= 3 ⋅ 2,5 = 7,5 грн.

*Еп* = 29,9 + 5 = 34,9 грн.

*Еа* = 44,85 + 7,5 = 52,35 грн.

*Веп* = 34,9 ⋅ 252 = 8794,8 грн.

*Веа* = 52,35 ⋅ 252 = 13192,2 грн.

# 4.4. Розрахунок ціни споживання проектного рішення

Ціна споживання - це витрати на придбання і експлуатацію проектного рішення за весь строк його служби:

*Цсп = Цп + Вепрv*(4.14)

де *Цп* - ціна придбання проектного рішення, грн.:

(4.15)



де *Пр* - норматив рентабельності;

*Ко* - витрати на прив'язку та освоєння проектного рішення на конкретному об'єкті, грн.;

*Кк* - витрати на доукомплектування технічних засобів на об'єкті, грн.;

*Цп*= 5182,65 ⋅ (1 + 0,3) = 6737,45 грн.

Benpv - теперішня вартість витрат на експлуатацію проектного рішення (за весь час його експлуатації), грн.:

(4.16)



де Веп - річні експлуатаційні витрати, грн.;

T - строк служби проектного рішення, років;

R - річна ставка проценту банківського

грн.



грн.



Цсп = 6737,45 + 29247,2 = 35984,65 грн.

Цса = 5200,0 + 43870,79 = 49070,79 грн.

# 4.5. Визначення показників економічної ефективності

Економічний ефект в сфері проектування рішення:

*Епр*= *Ца*- *Цп* (4.17)

*Епр* = 5200,0 – 6737,45 = -1537,45 грн.

Річний економічний ефект в сфері експлуатації:

*Екс* = *Веа* - *Веп* (4.18)

*Екс*= 13192,2 - 8794,8 = 4397,4 грн.

Додатковий економічний ефект у сфері експлуатації:



грн.



Сумарний ефект складає:

-1537,45 + 43665,83 = 42128,38 грн.



Таблиця 4.6. Показники економічної ефективності проектного рішення.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Найменування | Одиниці вимірювання | Значення показників | |
| Базовий  варіант | Новий варіант |
| 1 | Капітальні вкладення | Грн. | - | 5182,65 |
| 2 | Ціна придбання | Грн. | 5200 | 6737,45 |
| 3 | Річні експлуатаційні витрати | Грн. | 43870,79 | 29247,2 |
| 4 | Ціна споживання | Грн. | 49070,79 | 35984,65 |
| 5 | Економічний ефект в сфері проектування | Грн. | - | -1537,45 |
| 6 | Економічний ефект в сфері експлуатації | Грн. | - | 4397,4 |
| 7 | Додатковий ефект в сфері експлуатації | Грн. | - | 43665,83 |
| 8 | Сумарний ефект | Грн. | - | 42128,38 |

# Висновки до економічного розділу

У даному розділі бакалаврської кваліфікаційної роботи я здійснив економічне обґрунтування доцільності розробки програмного забезпечення. Також було здійснено порівняння з існуючим аналогом. Результати порівняння показали, що дане програмне забезпечення має переваги порівняно з аналогом, зокрема: зменшення часу обробки томограм, надійність, гнучкість. Розробка і впровадження даного проектного рішення є економічно доцільною, згідно здійсненого економічного обґрунтування. Економічний ефект від впровадження даного проектного рішення додатній і складає 42128,38 грн.

# ВИСНОВКИ

Покращення функціонування медичних інформаційних систем має дуже важливе значення. Особливо критично це проявляється під час надзвичайних ситуацій. В даній магістерській кваліфікаційній роботі розроблено мобільний термінал системи передачі медичних зображень для покращення роботи таких систем.

В першому розділі було проведено аналіз існуючих систем, порівняння їх функціональних особливостей. Було розглянуто найбільш поширені стандарти галузі. Також було проаналізовано питання безпеки медичних даних.

В другому розділі було

В третьому розділі було

В четвертому розділі було

В пятому розділі було роведено економічне обґрунтування доцільності розробки мобільного терміналу системи передачі медичних зображень та показано що розробка і впровадження цього рішення є економічно доцільною.

Розроблений програмний продукт може бути впроваджений у роботу лікарень та госпіталів. Програма має хороші перспективи розвитку, оскільки можна додати нові функції для роботи з зображеннями, додати підтримку старіший версій мобільних пристроїв, тощо. Можна зробити висновок, що запропонована система моніторингу покращує можливості медичних інформаційних систем, а саме:

* завдяки мобільності система може бути розгорнута на базі поширених мобільних пристроїв (телефони, планшети);
* дозволяє скоротити час на обслуговування пацієнта;
* дозволяє лікарю напряму працювати з первинним зображенням;
* дозволяє скоротити витрати на друк медичних зображень.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Основи медичної інформатики. Підручник /Л. О. Момоток, Л. В. Юшина, О. В. Рожнова. - К.: Медицина, 2008.-232с.

2. Прокопчук Ю.А. Интеллектуальные медицинские системы: формально-логический уровень.-Днепропетровск: ИТМ НАНУ и НКАУ, 2007.-259 с.

3. Лапшин М.А. Расширенная архитектура PACS в распределенной медицинской информационной системе ИПС РАН/ ЛапшинМ.А. – Переславль-Залесский, 2004. – 188c.

4. Цивільний кодекс України [Текст] : прийнятий 16 січня 2003 р. / Верховна Рада України. - К. : Парламентське видавництво, 2003. - 352 с.

5. Основи законодавства України про охорону здоров'я: Закон України від 19 лист. 1992 р. № 2801-XII (в ред. Закону України від 04 липн. 2014 р. № N 1590-VII).

6. Oleg S. Pianykh, Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide, Springer Science & Business Media, 2008. - 404 с.

7. O'Reilly Media. OReilly - Making Android Accessories with IOIO: “Simon Monk”, 2012. - 60 c.

8. Greg Milette, Adam Stroud. Professional Android Sensor Programming. “Wrox”, 2013. - 556 c.

9. Хашими С., Коматинени С., Маклин Д. Разработка приложений для Android: “Питер”, 2011. - 738 с.

10. Mark L. Murphy. Beginning Android: “Apress”, 2010 — 416 c.

11. Олифер В.Г., Олифер Н.А. Компьютерные сети: “Питер”, 2010. – 944 с.

# ДОДАТОК А

**DCMData.java**

package lp.edu.ua.sopushynskyi.dicom;

import android.graphics.Bitmap;

import com.imebra.dicom.CodecFactory;

import com.imebra.dicom.ColorTransformsFactory;

import com.imebra.dicom.DataSet;

import com.imebra.dicom.DrawBitmap;

import com.imebra.dicom.Image;

import com.imebra.dicom.ModalityVOILUT;

import com.imebra.dicom.Stream;

import com.imebra.dicom.StreamReader;

import com.imebra.dicom.TransformsChain;

import com.imebra.dicom.VOILUT;

import java.util.LinkedList;

import java.util.List;

public class DCMData {

List<ImageData> frames = new LinkedList<ImageData>();

private int brightness;

private int currentFrame;

private double contrast;

private ColorSchema schema;

private boolean isLoaded;

private String fileName;

private MetaData metaData;

public DCMData() {

setDefault();

}

public boolean isLoaded() {

return isLoaded;

}

public void setContrast(float contrast) {

this.contrast = contrast;

}

public void setBrightness(int brightness) {

this.brightness = brightness;

}

private void setLoaded(boolean isLoaded) {

this.isLoaded = isLoaded;

}

public void setDefault() {

currentFrame = 0;

brightness = 0;

contrast = 1;

schema = ColorSchema.NORMAL;

}

public void loadDCM(String fileName) {

frames.clear();

metaData = new MetaData();

setLoaded(true);

Stream stream = new Stream();

stream.openFileRead(fileName);

DataSet dataSet = CodecFactory.load(new StreamReader(stream), 256);

int framesSize = 0;

String manufacturer = dataSet.getString(0x0008, 0, 0x0070, 0);

String manufacturerModel = dataSet.getString(0x0008, 0, 0x1090, 0);

String patientName = dataSet.getString(0x0010, 0, 0x0010, 0);

String patientID = dataSet.getString(0x0010, 0, 0x0020, 0);

String patientAge = dataSet.getString(0x0010, 0, 0x1010, 0);

String patientWeight = dataSet.getString(0x0010, 0, 0x1030, 0);

String procedure = dataSet.getString(0x0032, 0, 0x1060, 0);

String date = dataSet.getString(0x0008, 0, 0x0020, 0);

String time = dataSet.getString(0x0008, 0, 0x0030, 0);

metaData.setManufacturer(manufacturer);

metaData.setManufacturerModel(manufacturerModel);

metaData.setPatientName(patientName);

metaData.setPatientID(patientID);

metaData.setPatientAge(patientAge);

metaData.setPatientWeight(patientWeight);

metaData.setProcedure(procedure);

if(time != null && !time.equals(""))

metaData.setDate(date + " " + time.substring(0, 8));

try {

for (int i = 0; ; i++) {

dataSet.getImage(i);

framesSize++;

}

} catch (Exception e) {

e.printStackTrace();

}

for (int i = 0; i < framesSize; i++) {

Image image = dataSet.getImage(i);

if (ColorTransformsFactory.isMonochrome(image.getColorSpace())) {

ModalityVOILUT modalityVOILUT = new ModalityVOILUT(dataSet);

if (!modalityVOILUT.isEmpty()) {

Image modalityImage = modalityVOILUT.allocateOutputImage(image, image.getSizeX(), image.getSizeY());

modalityVOILUT.runTransform(image, 0, 0, image.getSizeX(), image.getSizeY(), modalityImage, 0, 0);

image = modalityImage;

}

}

TransformsChain transformsChain = new TransformsChain();

if (ColorTransformsFactory.isMonochrome(image.getColorSpace())) {

VOILUT voilut = new VOILUT(dataSet);

int voilutId = voilut.getVOILUTId(0);

if (voilutId != 0) {

voilut.setVOILUT(voilutId);

} else {

voilut.applyOptimalVOI(image, 0, 0, image.getSizeX(), image.getSizeY());

}

transformsChain.addTransform(voilut);

}

DrawBitmap bitmap = new DrawBitmap(image, transformsChain);

int[] buffer = new int[1024];

int requiredSize = bitmap.getBitmap(image.getSizeX(), image.getSizeY(), 0, 0,

image.getSizeX(), image.getSizeY(), buffer, 0);

buffer = new int[requiredSize];

bitmap.getBitmap(image.getSizeX(), image.getSizeY(), 0, 0,

image.getSizeX(), image.getSizeY(), buffer, requiredSize);

frames.add(new ImageData(buffer, image.getSizeX(), image.getSizeY()));

this.fileName = fileName;

}

}

public Bitmap getFrame() {

return frames.get(currentFrame).getBitmap(isInverted(), isRainbowed(), getContrast(), getBrightness());

}

public int getBrightness() {

return brightness;

}

public String getFileName() {

return fileName;

}

public void addBrightness(int brightness) {

this.brightness += brightness;

if (this.brightness >= 260)

this.brightness = 260;

}

public void minusBrightness(int brightness) {

this.brightness -= brightness;

if (this.brightness <= -260)

this.brightness = -260;

}

public void addContrast(double contrast) {

this.contrast += contrast;

if (this.contrast > 2)

this.contrast = 2;

}

public void minusContrast(double contrast) {

this.contrast -= contrast;

if (this.contrast < 0)

this.contrast = 0;

}

public boolean isInverted() {

return schema == ColorSchema.INVERSE;

}

public void inverse() {

schema = ColorSchema.INVERSE;

}

public double getContrast() {

return contrast;

}

public void rainbow() {

schema = ColorSchema.RAINBOW;

}

public boolean isRainbowed() {

return schema == ColorSchema.RAINBOW;

}

public String getColorSchema() {

return schema.toString();

}

public String getMetaInfo() {

return metaData.toString();

}

public void normal() {

schema = ColorSchema.NORMAL;

}

public boolean isNormal() {

return schema == ColorSchema.NORMAL;

}

public int getCurrentFrame() {

return currentFrame;

}

public int getFramesNumber() {

return frames.size();

}

public boolean nextFrame() {

currentFrame = currentFrame + 1;

if(currentFrame >= frames.size()) {

currentFrame = frames.size() - 1;

return false;

}

return true;

}

public boolean prevFrame() {

currentFrame = currentFrame - 1;

if(currentFrame < 0) {

currentFrame = 0;

return false;

}

return true;

}

}

**ImageData.java**

package lp.edu.ua.sopushynskyi.dicom;

import android.graphics.Bitmap;

import android.graphics.Color;

public class ImageData {

private int[] buffer;

private int[] workBuffer;

private int sizeX;

private int sizeY;

public ImageData(int[] buffer, int sizeX, int sizeY) {

this.buffer = buffer;

this.workBuffer = new int[buffer.length];

this.sizeX = sizeX;

this.sizeY = sizeY;

}

public Bitmap getBitmap(boolean isInverted, boolean isRainbowed, double contrast, int brightness) {

copyBuffer();

contrastImage(contrast);

brightImage(brightness);

invertImage(isInverted);

rainbowImage(isRainbowed);

return Bitmap.createBitmap(workBuffer, sizeX, sizeY, Bitmap.Config.RGB\_565);

}

private void copyBuffer() {

int length = sizeX \* sizeY;

System.arraycopy(buffer, 0, workBuffer, 0, length);

}

private void invertImage(boolean isInverted) {

if (isInverted) {

int length = sizeX \* sizeY;

for (int i = 0; i < length; i++) {

workBuffer[i] = (workBuffer[i] & 0xff000000) | ~(workBuffer[i] & 0x00ffffff);

}

}

}

private void rainbowImage(boolean isRainbowed) {

float hsb[] = new float[3];

if (isRainbowed) {

int length = sizeX \* sizeY;

int value;

hsb[1] = 1f;

for (int i = 0; i < length; i++) {

value = workBuffer[i] & 0x000000ff;

hsb[0] = (255f - (float) value) / 255;

hsb[2] = (float) value / 255;

workBuffer[i] = hsbToColor(hsb[0], hsb[2]);

}

}

}

private void contrastImage(double contrast) {

int value;

int length = sizeX \* sizeY;

for (int i = 0; i < length; i++) {

value = workBuffer[i] & 0x000000ff;

value = (int) (((((value / 255.0) - 0.5) \* contrast) + 0.5) \* 255.0);

if (value < 0) {

value = 0;

} else if (value > 255) {

value = 255;

}

workBuffer[i] = Color.rgb(value, value, value);

}

}

private void brightImage(int brightness) {

if (brightness != 0) {

int length = sizeX \* sizeY;

int value;

for (int i = 0; i < length; i++) {

value = workBuffer[i] & 0x000000ff;

value += brightness;

if (value > 255) {

value = 255;

} else if (value < 0) {

value = 0;

}

workBuffer[i] = Color.rgb(value, value, value);

}

}

}

private int hsbToColor(float hue, float brightness) {

int r = 0, g = 0, b = 0;

double h = (hue - StrictMath.floor(hue)) \* 6.0f;

double f = h - StrictMath.floor(h);

double q = brightness \* (1.0f - f) \* 255.0f + 0.5f;

double t = brightness \* f \* 255.0f + 0.5f;

switch ((int) h) {

case 0:

r = (int) (brightness \* 255.0f + 0.5f);

g = (int) t;

b = 0;

break;

case 1:

r = (int) q;

g = (int) (brightness \* 255.0f + 0.5f);

b = 0;

break;

case 2:

r = 0;

g = (int) (brightness \* 255.0f + 0.5f);

b = (int) t;

break;

case 3:

r = 0;

g = (int) q;

b = (int) (brightness \* 255.0f + 0.5f);

break;

case 4:

r = (int) t;

g = 0;

b = (int) (brightness \* 255.0f + 0.5f);

break;

case 5:

r = (int) (brightness \* 255.0f + 0.5f);

g = 0;

b = (int) q;

break;

}

return 0xff000000 | (r << 16) | (g << 8) | (b << 0);

}

}

**ColorSchema.java**

package lp.edu.ua.sopushynskyi.dicom;

public enum ColorSchema {

NORMAL("норма"),

INVERSE("негатив"),

RAINBOW("псевдо");

public String name;

ColorSchema(String name) {

this.name = name;

}

public String toString() {

return name;

}

}

**MetaData.java**

package lp.edu.ua.sopushynskyi.dicom;

public class MetaData {

private String manufacturer;

private String manufacturerModel;

private String patientName;

private String patientID;

private String procedure;

private String date;

private String patientAge;

private String patientWeight;

public String getManufacturer() {

return manufacturer;

}

public void setManufacturer(String manufacturer) {

this.manufacturer = manufacturer;

}

public String getDate() {

return date;

}

public void setDate(String date) {

this.date = date;

}

public String getProcedure() {

return procedure;

}

public void setProcedure(String procedure) {

this.procedure = procedure;

}

public String getManufacturerModel() {

return manufacturerModel;

}

public void setManufacturerModel(String manufacturerModel) {

this.manufacturerModel = manufacturerModel;

}

public String getPatientName() {

return patientName;

}

public void setPatientName(String patientName) {

this.patientName = patientName;

}

public String getPatientID() {

return patientID;

}

public void setPatientID(String patientID) {

this.patientID = patientID;

}

public String getPatientAge() {

return patientAge;

}

public void setPatientAge(String patientAge) {

this.patientAge = patientAge;

}

public String getPatientWeight() {

return patientWeight;

}

public void setPatientWeight(String patientWeight) {

this.patientWeight = patientWeight;

}

@Override

public String toString() {

StringBuilder bld = new StringBuilder();

bld.append(String.format("Пацієнт : [%s] %s\n", patientID, patientName));

bld.append(String.format("Дата : %s\n", date));

bld.append(String.format("Процедура: %s\n", procedure));

bld.append(String.format("Прилад : %s %s", manufacturer, manufacturerModel));

return bld.toString();

}

}

**OpenFileDialog.java**

package lp.edu.ua.sopushynskyi.dialogs;

import android.app.AlertDialog;

import android.content.Context;

import android.content.DialogInterface;

import android.graphics.Paint;

import android.graphics.Point;

import android.graphics.Rect;

import android.os.Environment;

import android.util.DisplayMetrics;

import android.util.TypedValue;

import android.view.Display;

import android.view.Gravity;

import android.view.View;

import android.view.ViewGroup;

import android.view.WindowManager;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.LinearLayout;

import android.widget.ListView;

import android.widget.TextView;

import android.widget.Toast;

import java.io.File;

import java.io.FilenameFilter;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.Comparator;

import java.util.LinkedList;

import java.util.List;

public class OpenFileDialog extends AlertDialog.Builder {

private String currentPath = Environment.getExternalStorageDirectory().getPath();

private List<File> files = new ArrayList<File>();

private TextView title;

private ListView listView;

private FilenameFilter filenameFilter;

private int selectedIndex = -1;

private OpenDialogListener listener;

public OpenFileDialog(Context context) {

super(context);

title = createTitle(context);

changeTitle();

LinearLayout linearLayout = createMainLayout(context);

files.addAll(getFiles(currentPath));

listView = createListView(context);

listView.setAdapter(new FileAdapter(context, files));

linearLayout.addView(listView);

setCustomTitle(title)

.setView(linearLayout)

.setPositiveButton(android.R.string.ok, null)

.setNegativeButton(android.R.string.cancel, null)

.setPositiveButton(android.R.string.ok, new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

if (selectedIndex > -1 && listener != null) {

listener.OnSelectedFile(listView.getItemAtPosition(selectedIndex).toString());

}

}

});

}

private static Display getDefaultDisplay(Context context) {

return ((WindowManager) context.getSystemService(Context.WINDOW\_SERVICE)).getDefaultDisplay();

}

private static Point getScreenSize(Context context) {

Point screeSize = new Point();

getDefaultDisplay(context).getSize(screeSize);

return screeSize;

}

private static int getLinearLayoutMinHeight(Context context) {

return getScreenSize(context).y;

}

@Override

public AlertDialog show() {

files.addAll(getFiles(currentPath));

listView.setAdapter(new FileAdapter(getContext(), files));

return super.show();

}

public OpenFileDialog setFilter(final String filter) {

filenameFilter = new FilenameFilter() {

@Override

public boolean accept(File file, String fileName) {

File tempFile = new File(String.format("%s/%s", file.getPath(), fileName));

if (tempFile.isFile())

return tempFile.getName().endsWith(filter);

return true;

}

};

return this;

}

private TextView createTextView(Context context, int style) {

TextView textView = new TextView(context);

textView.setTextAppearance(context, style);

int itemHeight = getItemHeight(context);

textView.setLayoutParams(new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH\_PARENT, itemHeight));

textView.setMinHeight(itemHeight);

textView.setGravity(Gravity.CENTER\_VERTICAL);

textView.setPadding(15, 0, 0, 0);

return textView;

}

private TextView createTitle(Context context) {

TextView textView = createTextView(context, android.R.style.TextAppearance\_DeviceDefault\_DialogWindowTitle);

return textView;

}

public int getTextWidth(String text, Paint paint) {

Rect bounds = new Rect();

paint.getTextBounds(text, 0, text.length(), bounds);

return bounds.left + bounds.width() + 80;

}

private void changeTitle() {

String titleText = currentPath;

int screenWidth = getScreenSize(getContext()).x;

int maxWidth = (int) (screenWidth \* 0.99);

if (getTextWidth(titleText, title.getPaint()) > maxWidth) {

while (getTextWidth("..." + titleText, title.getPaint()) > maxWidth) {

int start = titleText.indexOf("/", 2);

if (start > 0)

titleText = titleText.substring(start);

else

titleText = titleText.substring(2);

}

title.setText("..." + titleText);

} else {

title.setText(titleText);

}

}

private int getItemHeight(Context context) {

TypedValue value = new TypedValue();

DisplayMetrics metrics = new DisplayMetrics();

context.getTheme().resolveAttribute(android.R.attr.rowHeight, value, true);

getDefaultDisplay(context).getMetrics(metrics);

return (int) TypedValue.complexToDimension(value.data, metrics);

}

private void RebuildFiles(ArrayAdapter<File> adapter) {

try {

selectedIndex = -1;

List<File> fileList = getFiles(currentPath);

files.clear();

files.addAll(fileList);

adapter.notifyDataSetChanged();

changeTitle();

} catch (NullPointerException e) {

Toast.makeText(getContext(), android.R.string.unknownName, Toast.LENGTH\_SHORT).show();

}

}

private LinearLayout createMainLayout(Context context) {

LinearLayout linearLayout = new LinearLayout(context);

linearLayout.setOrientation(LinearLayout.VERTICAL);

linearLayout.setMinimumHeight(getLinearLayoutMinHeight(context));

return linearLayout;

}

private List<File> getFiles(String directoryPath) {

File directory = new File(directoryPath);

List<File> fileList = new LinkedList<File>(Arrays.asList(directory.listFiles(filenameFilter)));

Collections.sort(fileList, new Comparator<File>() {

@Override

public int compare(File file, File file2) {

if (file.isDirectory() && file2.isFile())

return -1;

else if (file.isFile() && file2.isDirectory())

return 1;

else

return file.getPath().compareTo(file2.getPath());

}

});

if (!currentPath.equals(Environment.getExternalStorageDirectory().getPath())) {

fileList.add(0, new File(currentPath).getParentFile());

}

return fileList;

}

private ListView createListView(Context context) {

ListView listView = new ListView(context);

listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> adapterView, View view, int index, long l) {

final ArrayAdapter<File> adapter = (FileAdapter) adapterView.getAdapter();

File file = adapter.getItem(index);

selectedIndex = -1;

if (file.isDirectory()) {

currentPath = file.getPath();

RebuildFiles(adapter);

} else {

if (index != selectedIndex)

selectedIndex = index;

}

adapter.notifyDataSetChanged();

}

});

return listView;

}

public OpenFileDialog setOpenDialogListener(OpenDialogListener listener) {

this.listener = listener;

return this;

}

public interface OpenDialogListener {

public void OnSelectedFile(String fileName);

}

private class FileAdapter extends ArrayAdapter<File> {

public FileAdapter(Context context, List<File> files) {

super(context, android.R.layout.simple\_list\_item\_1, files);

}

@Override

public View getView(int position, View convertView, ViewGroup parent) {

TextView view = (TextView) super.getView(position, convertView, parent);

File file = getItem(position);

if (file.getAbsolutePath().equals(new File(currentPath).getParentFile().getAbsolutePath())) {

view.setText("..");

} else {

view.setText(file.getName());

}

if (selectedIndex == position) {

view.setBackgroundColor(getContext().getResources().getColor(android.R.color.holo\_blue\_light));

} else {

view.setBackgroundColor(getContext().getResources().getColor(android.R.color.transparent));

}

return view;

}

}

}

**VerticalSeekBar.java**

package lp.edu.ua.sopushynskyi.components;

import android.content.Context;

import android.graphics.Canvas;

import android.util.AttributeSet;

import android.view.MotionEvent;

import android.widget.SeekBar;

public class VerticalSeekBar extends SeekBar {

public VerticalSeekBar(Context context) {

super(context);

}

public VerticalSeekBar(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

}

public VerticalSeekBar(Context context, AttributeSet attrs) {

super(context, attrs);

}

protected void onSizeChanged(int w, int h, int oldw, int oldh) {

super.onSizeChanged(h, w, oldh, oldw);

}

@Override

protected synchronized void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {

super.onMeasure(heightMeasureSpec, widthMeasureSpec);

setMeasuredDimension(getMeasuredHeight(), getMeasuredWidth());

}

protected void onDraw(Canvas c) {

c.rotate(-90);

c.translate(-getHeight(), 0);

super.onDraw(c);

}

@Override

public boolean onTouchEvent(MotionEvent event) {

if (!isEnabled()) {

return false;

}

switch (event.getAction()) {

case MotionEvent.ACTION\_DOWN:

case MotionEvent.ACTION\_MOVE:

case MotionEvent.ACTION\_UP:

setProgress(getMax() - (int) (getMax() \* event.getY() / getHeight()));

onSizeChanged(getWidth(), getHeight(), 0, 0);

break;

case MotionEvent.ACTION\_CANCEL:

break;

}

return true;

}

}

**ColorSchema.java**

package lp.edu.ua.sopushynskyi.activities;

import android.app.Activity;

import android.graphics.RectF;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuInflater;

import android.view.MenuItem;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.ImageView;

import android.widget.LinearLayout;

import android.widget.RelativeLayout;

import android.widget.SeekBar;

import android.widget.TextView;

import android.widget.Toast;

import com.example.mykola.mydicom.R;

import lp.edu.ua.sopushynskyi.components.VerticalSeekBar;

import lp.edu.ua.sopushynskyi.dialogs.OpenFileDialog;

import lp.edu.ua.sopushynskyi.dicom.DCMData;

import lp.edu.ua.sopushynskyi.dicom.Echo;

import uk.co.senab.photoview.PhotoViewAttacher;

public class MyActivity extends Activity {

private static final String FTYPE = ".dcm";

ImageView img;

TextView imgInfo;

TextView dcmInfo;

PhotoViewAttacher mAttacher;

private String mChosenFile;

private DCMData dcmData = new DCMData();

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_my);

Button normal = (Button) findViewById(R.id.normal);

Button inverse = (Button) findViewById(R.id.inverse);

Button rainbow = (Button) findViewById(R.id.rainbow);

Button next = (Button) findViewById(R.id.next);

Button prev = (Button) findViewById(R.id.previous);

next.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if(dcmData.nextFrame())

redrawImage();

}

});

prev.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if(dcmData.prevFrame())

redrawImage();

}

});

img = (ImageView) findViewById(R.id.image);

imgInfo = (TextView) findViewById(R.id.info);

dcmInfo = (TextView) findViewById(R.id.metaInfo);

System.loadLibrary("imebra\_lib");

View.OnClickListener infoPanelListener = new View.OnClickListener() {

@Override

public void onClick(View v) {

TextView textPanel = (TextView)v;

if(textPanel.getCurrentTextColor() == 0xff00ff00) {

textPanel.setTextColor(0x55000000);

textPanel.setBackgroundColor(0x00ffffff);

} else {

textPanel.setTextColor(0xff00ff00);

textPanel.setBackgroundColor(0x55ffffff);

}

}

};

dcmInfo.setOnClickListener(infoPanelListener);

imgInfo.setOnClickListener(infoPanelListener);

inverse.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

dcmData.inverse();

redrawImage();

}

});

rainbow.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

dcmData.rainbow();

redrawImage();

}

});

normal.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

dcmData.normal();

redrawImage();

}

});

mAttacher = new PhotoViewAttacher(img);

mAttacher.setOnMatrixChangeListener(new PhotoViewAttacher.OnMatrixChangedListener() {

@Override

public void onMatrixChanged(RectF rectF) {

printInfo();

}

});

LinearLayout.LayoutParams layoutParams =

new LinearLayout.LayoutParams(ViewGroup.LayoutParams.WRAP\_CONTENT, 500);

layoutParams.setMargins(0, 0, 0, 0);

VerticalSeekBar contrastBar = new VerticalSeekBar(this);

contrastBar.setLayoutParams(layoutParams);

contrastBar.setMax(100);

contrastBar.setProgress(25);

contrastBar.setPadding(30, 20, 30, 20);

contrastBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

dcmData.setContrast(progress / 100f \* 2 + 0.5f);

redrawImage();

}

public void onStartTrackingTouch(SeekBar seekBar) {

}

public void onStopTrackingTouch(SeekBar seekBar) {

}

});

RelativeLayout rightPanel = (RelativeLayout)findViewById(R.id.rightPanel);

RelativeLayout.LayoutParams contrastParams =

new RelativeLayout.LayoutParams(ViewGroup.LayoutParams.WRAP\_CONTENT, 500);

contrastParams.addRule(RelativeLayout.BELOW, R.id.icon\_contrast);

rightPanel.addView(contrastBar, contrastParams);

VerticalSeekBar brightnessBar = new VerticalSeekBar(this);

brightnessBar.setLayoutParams(layoutParams);

brightnessBar.setMax(100);

brightnessBar.setProgress(50);

brightnessBar.setPadding(30, 20, 30, 20);

brightnessBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

dcmData.setBrightness((int)(progress / 100f \* 500) - 250);

redrawImage();

}

public void onStartTrackingTouch(SeekBar seekBar) {

}

public void onStopTrackingTouch(SeekBar seekBar) {

}

});

RelativeLayout.LayoutParams brightnessParams =

new RelativeLayout.LayoutParams(ViewGroup.LayoutParams.WRAP\_CONTENT, 500);

brightnessParams.addRule(RelativeLayout.BELOW, R.id.icon\_brightness);

RelativeLayout leftPanel = (RelativeLayout)findViewById(R.id.leftPanel);

leftPanel.addView(brightnessBar, brightnessParams);

}

private void redrawImage() {

if (dcmData.isLoaded()) {

img.setImageBitmap(dcmData.getFrame());

printInfo();

}

}

private void printInfo() {

String info = String.format(

"Схема : %s\n" +

"Контраст: %.2f\n" +

"Яскрав. : %d\n" +

"Масштаб : %.2f%%\n" +

"Кадр : %d/%d",

dcmData.getColorSchema(),

dcmData.getContrast(),

dcmData.getBrightness(),

mAttacher.getScale() \* 100,

dcmData.getCurrentFrame() + 1,

dcmData.getFramesNumber());

imgInfo.setText(info);

String metaInfo = dcmData.getMetaInfo();

dcmInfo.setText(metaInfo);

}

private void loadDCM(String fileName) {

dcmData.loadDCM(fileName);

redrawImage();

mAttacher.update();

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.my, menu);

return super.onCreateOptionsMenu(menu);

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.action\_search:

OpenFileDialog fileDialog = new OpenFileDialog(this).setFilter(FTYPE).setOpenDialogListener(new OpenFileDialog.OpenDialogListener() {

@Override

public void OnSelectedFile(String fileName) {

mChosenFile = fileName;

setTitle(mChosenFile);

Toast.makeText(getApplicationContext(), fileName, Toast.LENGTH\_LONG).show();

loadDCM(fileName);

}

});

fileDialog.show();

return true;

case R.id.action\_network:

Echo echo = new Echo();

echo.makeEcho();

return true;

default:

return super.onOptionsItemSelected(item);

}

}

}

**AndroidManifest.xml**

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.mykola.mydicom" >

<application

android:allowBackup="true"

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<activity

android:name="lp.edu.ua.sopushynskyi.activities.MyActivity"

android:label="@string/app\_name"

android:screenOrientation="landscape">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.INTERNET" />

</manifest>

**myMenu.xml**

<menu xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

tools:context=".MyActivity" >

<item android:id="@+id/action\_search"

android:title="@string/action\_find"

android:icon="@android:drawable/ic\_menu\_search"

android:orderInCategory="100"

android:showAsAction="ifRoom" />

<item android:id="@+id/action\_network"

android:title="@string/action\_find"

android:icon="@android:drawable/stat\_notify\_sync"

android:orderInCategory="100"

android:showAsAction="ifRoom" />

</menu>

**activity\_my.xml**

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/mainLayout"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="horizontal">

<ImageView

android:id="@+id/image"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@android:color/black" />

<RelativeLayout

android:id="@+id/leftPanel"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_gravity="left"

android:background="#11ffffff"

android:orientation="vertical">

<ImageView

android:id="@+id/icon\_brightness"

android:layout\_width="100px"

android:layout\_height="wrap\_content"

android:src="@drawable/ic\_br" />

<Button

android:id="@+id/rainbow"

android:layout\_width="100px"

android:layout\_height="100px"

android:layout\_alignParentBottom="true"

android:text="R" />

<Button

android:id="@+id/inverse"

android:layout\_width="100px"

android:layout\_height="100px"

android:layout\_above="@+id/rainbow"

android:text="I" />

<Button

android:id="@+id/normal"

android:layout\_width="100px"

android:layout\_height="100px"

android:layout\_above="@+id/inverse"

android:text="N" />

</RelativeLayout>

<TextView

android:id="@+id/metaInfo"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentTop="true"

android:layout\_toRightOf="@+id/leftPanel"

android:background="#55ffffff"

android:fontFamily="Courier New"

android:gravity="bottom"

android:minWidth="300px"

android:text="@string/image\_info"

android:textColor="#ff00ff00" />

<TextView

android:id="@+id/info"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:layout\_toRightOf="@+id/leftPanel"

android:background="#55ffffff"

android:fontFamily="Courier New"

android:gravity="bottom"

android:minWidth="300px"

android:text="@string/image\_info"

android:textColor="#ff00ff00" />

<RelativeLayout

android:id="@+id/rightPanel"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_alignParentRight="true"

android:layout\_gravity="right"

android:background="#11ffffff"

android:orientation="vertical">

<ImageView

android:id="@+id/icon\_contrast"

android:layout\_width="100px"

android:layout\_height="wrap\_content"

android:src="@drawable/ic\_cn"/>

<Button

android:id="@+id/next"

android:layout\_width="100px"

android:layout\_height="100px"

android:layout\_alignParentBottom="true"

android:text="@string/next" />

<Button

android:id="@+id/previous"

android:layout\_width="100px"

android:layout\_height="100px"

android:layout\_above="@+id/next"

android:text="@string/prev" />

</RelativeLayout>

</RelativeLayout>