ДВНЗ «Донецький національний технічний університет»
Факультет комп'ютерно-інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету)
Кафедра прикладної математики та інформатики
(повна назва кафедри)

«До захисту допущено»
завідувачка кафедри
_____ Наталія МАСЛОВА
(підпис) (ім'я та прізвище)
"___"_____20__ р.

Випускна кваліфікаційна робота

магістра
(освітньо-кваліфікаційний рівень)

на тему «Дослідження особливостей реалізації принципів захисту інформації в електронних медичних системах»

Виконав: студент 2 курсу, групи КНм-22
(шифр групи)
Спеціальності 122 Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)

Дудкін Андрій Олексійович
(прізвище та ініціали) (підпис)

Керівник зав. кафедри ПМІ, к.т.н., доцент Маслова Н.О.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент в.о.зав.каф. ЕТКІ, к.т.н., доц., Ковальов С.О.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доц., к.т.н., доц. кафедри АТ, Воропаєва А.О.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Луцьк-2023

ДВНЗ «Донецький національний технічний університет»
Факультет Комп'ютерних наук і технологій
Кафедра Прикладної математики та інформатики
Освітній ступінь Магістр
Спеціальність 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ:
Завідувачка кафедри
Н.О. Маслова
"_____"_____20____року

З А В Д А Н Н Я

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Дудкіну Андрію Олексійовичу
(прізвище, ім'я, по батькові)

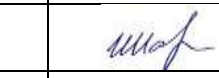
1. Тема роботи Дослідження особливостей реалізації принципів захисту інформації в електронних медичних системах

2. Керівник роботи завідувач кафедри ПМІ - к.т.н., доцент Маслова Н.О.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом від "22" вересня 2023 року № 478

3. Строк подання студентом роботи 7 грудня 2023 року
4. Вихідні дані до роботи результати переддипломної практики; система моніторингу та виявлення аномальних активностей в електронних медичних системах (ЕМС) для підвищення кібербезпеки; язик програмування Java
5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1) Аналіз поточного стану системи захисту даних.
   2) Оцінка рівня захищеності медичної інформації.
   3) Вивчення стандартів і регуляторних вимог.
   4) Розробка та впровадження стратегій і заходів безпеки.
   5) Моніторинг та аудит безпеки.
   6) Підготовка рекомендацій та звіту.
   7) Інформаційна освіта і навчання персоналу

6. Перелік графічного матеріалу: робота містить 2 схеми, 25 рисунків, загалом, в кількості, достатній для розкриття сутності та пояснення результатів випускної кваліфікаційної роботи

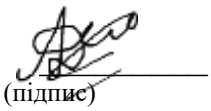
Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтролер	Доц. Назарова І.А.		

7. Дата видачі завдання 29.10.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літератури з предметної області, упорядкування	29.09.2023-06.10.2023	
2	Збір інформації і методичних відомостей щодо особливостей реалізації принципів захисту інформації в електронних медичних системах	07.10.2023-14.10.2023	
3	Розробка структури програмного забезпечення, вибір даних для обробки та експериментів.	15.10.2023-30.10.2023	
4	Тестування системи	31.10.2023-14.11.2023	
5	Оформлення пояснювальної записки	15.11.2023-30.11.2023	
6	Нормоконтроль пояснювальної записки, її перевірка антиплагіатною системою	01.11.2023-08.12.2023	
7	Оформлення супутніх матеріалів (розробка графічного матеріалу, написання доповіді, тощо) та рецензування роботи	09.12.2023-21.12.2023	
8	Захист випускної кваліфікаційної роботи	22.12.2023	

Студент  __ Дудкін А.О__
(підпис) (прізвище та ініціали)

Керівник роботи _ _Наталія МАСЛОВА
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дудкін А. О. Дослідження особливостей реалізації принципів захисту інформації в електронних медичних системах / Випускна кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 122 Комп’ютерні науки. – ДВНЗ ДонНТУ, Луцьк, 2023.

Об'єктом дослідження є технології побудови електронних медичних систем (ЕМС), вимоги та підходи до принципів захисту інформації.

Предметом дослідження є конкретні аспекти та компоненти, що стосуються реалізації принципів захисту інформації в електронних медичних системах (ЕМС).

Мета роботи полягає в дослідженні існуючих рішень забезпечення безпеки та конфіденційності медичної інформації в електронних медичних системах й наданні пропозицій щодо підвищення якості медичної допомоги та довіри пацієнтів.

Методи досліджень базуються на комплексному підході, включаючи аналіз літератури, збір та аналіз конкретних даних, проведення експериментів та тестування, експертні опитування, аналіз законодавства та вивчення кейс-студій. Цей підхід дозволяє здобути глибше розуміння особливостей захисту інформації в електронних медичних системах, оцінити ефективність заходів безпеки та розробити практичні рекомендації для покращення безпеки медичних даних.

Наукова новизна полягає у адаптації підходів до захисту інформації в електронних медичних системах, з урахуванням найновіших технологічних та правових викликів. Дослідження розкриває нові підходи до методів збору, обробки та передачі медичних даних, а також впровадження інноваційних засобів захисту, забезпечуючи високий рівень конфіденційності та безпеки медичної інформації.

Практичне значення дослідження полягає в тому, що воно надає конкретні рекомендації та стратегії для покращення захисту інформації в електронних медичних системах. Результати дослідження можуть бути використані розробниками систем електронного зберігання медичних даних, адміністраторами мережевої інфраструктури та фахівцями із безпеки для вдосконалення технічних, адміністративних та організаційних аспектів забезпечення безпеки в охороні здоров'я.

Ключові слова: Електронні медичні системи, захист інформації, конфіденційність даних, методи збору даних, обробка медичних даних, безпека медичних систем.

ABSTRACT

Dudkin A.O. Study of the peculiarities of implementing the principles of information security in electronic medical systems / Final qualification work for the degree of Master's Degree in specialty 122 Computer Science - SHEI DonNTU, Lutsk, 2023.

The object of research is the technology of building electronic medical systems (EMS), requirements and approaches to the principles of information security.

The subject of research is specific aspects and components related to the implementation of information security principles in electronic medical systems (EMS).

The purpose of the study is to investigate existing solutions for ensuring the security and confidentiality of medical information in electronic health systems and to provide proposals for improving the quality of medical care and patient confidence.

The research methods are based on a comprehensive approach, including literature review, collection and analysis of specific data, experiments and testing, expert surveys, analysis of legislation, and case studies. This approach allows us to gain a deeper understanding of the peculiarities of information protection in electronic healthcare systems, assess the effectiveness of security measures and develop practical recommendations for improving the security of medical data.

The scientific novelty lies in the adaptation of approaches to information security in electronic medical systems, taking into account the latest technological and legal challenges. The study reveals new approaches to the methods of collecting, processing and transmitting medical data, as well as the introduction of innovative security tools, ensuring a high level of confidentiality and security of medical information.

The practical significance of the study is that it provides specific recommendations and strategies for improving information security in electronic healthcare systems. The results of the study can be used by developers of electronic medical data storage systems, network infrastructure administrators and security specialists to improve the technical, administrative and organizational aspects of security in healthcare.

Keywords: Electronic medical systems, information protection, data privacy, data collection methods, medical data processing, medical system security.

Зміст

ВСТУП 9
1 ОГЛЯД ЕЛЕКТРОННИХ МЕДИЧНИХ СИСТЕМ 11
1.1 Поняття та приклади ЕМС 11
1.2 Класифікація ЕМС та їх проблеми 12
1.3 Вимоги до ЕМС 14
1.4 Система Helsi – переваги та недоліки 16
1.5 Висновки за розділом 1 17
2 РЕАЛІЗАЦІЯ ПРИНЦИПІВ ЗАХИСТУ ІНФОРМАЦІЇ В ЕЛЕКТРОННИХ МЕДИЧНИХ СИСТЕМАХ 19
2.1 Огляд Принципів Захисту Інформації 19
2.1.1 Основні Вимоги до Конфіденційності, Цілісності та Доступності 19
2.1.2 Переваги Інтеграції Сучасних Технологій 20
2.1.3 Забезпечення Захисту Інформації в Реальному Часі 21
2.1.4 Регулювання та Стандарти Захисту Даних в Медичній Сфері 22
2.1.5 Аудит та Моніторинг заходів Захисту 23
2.2 Висновок за розділом 2 24
3 МЕТОДИ ЗАСТОВНІ В ЕЛЕКТРОННИХ МЕДИЧНИХ СИСТЕМАХ 25
3.1 Методи збору та введення даних в ЕМС і безпека вхідних даних 25
3.2 Методи обробки даних та їх безпека 28
3.3 Методи захисту каналів передачі медичних даних. 29
3.4 Метод федеративного навчання для аналізу медичних даних схеми їх методів та прив’язки до Helsi 31
3.5 Висновки за розділом 3 33
4 ТЕХНОЛОГІЇ РОЗРОБКИ ЕМС 35
4.1 Технології Java для Розробки Електронних Медичних Систем 35
4.2 Spring Boot для реалізації бекенду електронних медичних систем 36
4.2.1 Розробка "Інтелектуального Агента" 37
4.3.1.2 Налаштування Telegram Bot через BotConfig 38
4.3 Інтеграція з Базами Даних: Postgres та MongoDB. 42
4.3.1 Конфігурація Postgresql 44

4.3.2 Конфігурація MongoDB 46
4.4 Tymeleaf для Реалізації Фронтенду Електронних Медичних Систем 47
4.5 Висновки за розділом 4 49
5 КОНТРОЛЬ ПРАЦЕЗДАТНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 50
5.1 Інструкція користувача для взаємодії з “Інтелектуальним агентом” 50
5.2 Обробка та збереження файлів 53
5.3 Переваги над існуючими аналогами. 56
5.4 Висновки за розділом 5 59
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 60
Додаток А 65
Додаток Б 66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЕМС – Електронні медичні системи

ЕМК – Електронна медична картка

EMR – Electronic Medical Record (EMR) - зберігає інформацію щодо конкретної медичної галузі (наприклад, стоматологія), зберігачем є клініка або практикуючий лікар

МОЗ - Міністерства охорони здоров'я

HIPAA - Health Insurance Portability and Accountability Act

GDPR - General Data Protection Regulation

ВСТУП

Актуальність тематики: Питання захисту інформації в електронних медичних системах стає надзвичайно актуальним у сучасному світі, оскільки медична сфера стає все більш залежною від інформаційних технологій. Діагностика, лікування, облік медичних даних і звітність – усе це відбувається за допомогою електронних систем. Однак разом з цим зростає загроза порушення конфіденційності, цілісності та доступності медичних даних. Недостатній рівень захисту може призвести до витоку конфіденційної інформації, порушення прав пацієнтів та негативно вплинути на безпеку медичного процесу в цілому.

Мета роботи полягає в дослідженні існуючих рішень забезпечення безпеки та конфіденційності медичної інформації в електронних медичних системах й наданні пропозицій щодо підвищення якості медичної допомоги та довіри пацієнтів.

Задачі роботи:

1. аналіз поточного стану системи захисту даних;
2. оцінка рівня захищеності медичної інформації;
3. вивчення стандартів і регуляторних вимог;
4. розробка та впровадження стратегій і заходів безпеки;
5. моніторинг та аудит безпеки;
6. підготовка рекомендацій та звіту;
7. інформаційна освіта і навчання персоналу.

Очікувана наукова та практична цінність: проект передбачає розробку програмного продукту, який буде автоматично моніторити роботу ЕМС у медичних закладах і виявляти незвичайні або підозрілі активності, які можуть бути пов'язані з кібератаками чи порушенням безпеки даних.

Ключові функції та завдання.

Моніторинг активності: відстеження дій користувачів та системних подій в ЕМС, таких як доступ до медичних записів, зміни у даних, спроби несанкціонованого доступу тощо.

Виявлення аномальних патернів: розробка алгоритмів для виявлення аномальних патернів активності, які можуть вказувати на кіберзагрози або недостатній рівень захисту.

Система сповіщень та реагування: створення системи сповіщень для адміністраторів та відповідальних осіб у випадку виявлення підозрілих активностей. Розробка інтерфейсу для вжиття заходів щодо реагування на інциденти.

Журнал подій та аналіз інцидентів: зберігання журналу подій та можливість аналізувати інциденти для подальшого вдосконалення системи безпеки.

Інтеграція з іншими ЕМС та системами: Можливість інтегрувати систему моніторингу з різними ЕМС і системами безпеки.

Загальний обсяг роботи складає 83 сторінки, що містять: _5__ розділів, графічні матеріали у вигляді __25__ рисунків, __2__ таблиць, 50 джерел інформації.

1 ОГЛЯД ЕЛЕКТРОННИХ МЕДИЧНИХ СИСТЕМ

1.1 Поняття та приклади ЕМС

Електронні медичні системи (ЕМС) - це інформаційні технології та програмні засоби, які використовуються в сфері охорони здоров'я для збору, збереження, обробки, передачі та аналізу медичних даних пацієнтів. ЕМС спрощують роботу медичного персоналу, забезпечують кращий доступ до інформації про пацієнтів, а також сприяють збереженню медичних даних в електронному форматі для поліпшення ефективності та точності надання медичної допомоги. Ось деякі поняття та приклади ЕМС [1]:

1) електронна медична картка (ЕМК) - це цифровий аналог паперової медичної картки пацієнта, де зберігаються історія хвороби, діагнози, лікування та інші медичні дані пацієнта;
2) лабораторні інформаційні системи - ці системи використовуються для збору та обробки результатів лабораторних досліджень, таких як аналізи крові, сечі, біопсії та інші медичні аналізи;
3) системи керування прийомами вони допомагають лікарям та медичному персоналу керувати розкладами прийому пацієнтів, реєструвати прийоми та нагадувати про їхній прийом;
4) системи управління лікарськими рецептами (для ефективного виписування та обробки лікарських рецептів), включаючи взаємодію з аптеками та страховими компаніями;
5) телемедицина (системи телемедицини дозволяють лікарям та пацієнтам взаємодіяти віддалено за допомогою відеозв'язку та інших технологій);
6) системи обліку медичного обладнання (допомагають відстежувати стан та ремонт медичного обладнання, а також планувати його обслуговування);

7) архівування медичних зображень (системи, які дозволяють зберігати та керувати медичними зображеннями, такими як рентгенівські знімки, УЗД, МРТ тощо);
8) системи електронного ведення обліку пацієнтів (програмні рішення для обліку та адміністрування інформації про пацієнтів, включаючи їхній стан здоров'я та лікування);
9) системи аналізу даних в охороні здоров'я (використовуються для аналізу великих обсягів медичних даних для виявлення патернів та покращення медичних практик);
10) системи та електронні звіти для документування медичної практики (допомагають лікарям і медичному персоналу ведення медичної документації та генерації звітів для потреб медичної практики та статистики).

ЕМС різняться за обсягом та функціональністю, і вони призначені для полегшення роботи медичного персоналу та поліпшення якості надання медичної допомоги пацієнтам, з дотриманням вимог щодо збереження конфіденційності та захисту персональних даних.

1.2 Класифікація ЕМС та їх проблеми

Класифікація електронних медичних систем (ЕМС) може бути проведена з різних точок зору, враховуючи різні аспекти функціональності і використання. Ось одна з можливих класифікацій разом із зазначенням основних проблем, пов'язаних з кожним класом [2].

За призначенням:

1) електронні медичні записи (EMR) - засоби для збереження та обробки медичної інформації пацієнтів в медичних установах. Проблеми - неоднорідність стандартів обміну даними між різними системами, ризик втрати даних через технічні неполадки або кібератаки;

2) системи керування лікарськими рецептами спрощують електронний обмін рецептами між лікарем, аптекою та пацієнтом. Проблеми - потрібна інтеграція з різними системами, забезпечення безпеки для запобігання маніпуляціям з рецептами;
3) телемедицина дозволяє віддалені консультації, моніторинг та діагностику. Проблеми - низька доступність в регіональних областях, потреба в високошвидкісному Інтернеті, забезпечення безпеки та конфіденційності під час передачі даних;
4) лабораторні інформаційні системи використовуються для керування та обробки результатів лабораторних тестів. Проблеми - інтероперабельність між різними лабораторіями, безпека даних та стандарти обміну даними[42].

За рівнем доступу:

1) пацієнтські портали дозволяють пацієнтам доступ до власної медичної інформації та спілкування з медичним персоналом. Проблеми - захист персональних даних, освіта пацієнтів для використання порталів;
2) лікарські портали забезпечують лікарям доступ до медичних записів та інших інструментів для обслуговування пацієнтів. Проблеми - безпека даних, висока вимога до доступності системи.

За типом інформації:

1) ЕМС для клінічних даних збирають та обробляють медичні дані, такі як діагнози, лікування та аналізи. Проблеми - стандартизація даних, забезпечення точності та доступності інформації;
2) ЕМС для фінансової та адміністративної інформації використовуються для фінансового управління та адміністрування медичних установ. Проблеми - безпека фінансових даних, інтеграція з іншими фінансовими системами.

Загальні проблеми, пов'язані з ЕМС, включають[3]:

1) конфіденційність та безпека даних забезпечеють захист медичних даних від несанкціонованого доступу, крадіжки даних та кібератак є однією з найбільших проблем;
2) інтероперабельність - різні системи медичної інформації часто мають різні стандарти та формати, що ускладнює обмін даними між ними;
3) вартість і власність - розробка та впровадження ЕМС може бути витратним процесом, а питання щодо власності медичних даних може бути складним;
4) освіта та навчання персоналу потребує навчання щодо використання ЕМС та забезпечення безпеки даних;
5) законодавство та регуляція потрібно дотримуватися законодавства щодо збереження та обробки медичних даних, що може бути складним завданням.

Ці проблеми потребують уваги та розв'язання для успішного впровадження та функціонування ЕМС у сфері охорони здоров'я.

1.3 Вимоги до ЕМС

Вимоги до електронних медичних систем (ЕМС) включають різні аспекти, включаючи безпеку та захист даних пацієнтів. Ось основні вимоги до ЕМС, зокрема до безпеки та захищеності даних:

Конфіденційність даних:

1) обмежений доступ, лише авторизованим користувачам, таким як медичний персонал та пацієнти, повинен бути наданий доступ до медичної інформації[4];
2) аудит доступу, логування та моніторинг доступу до даних для виявлення та відстеження незаконного доступу.

Цілісність даних надає Захист від модифікації забезпечує те, щоб дані не могли бути змінені без авторизації та обліку змін.

Доступність даних Запобігає відмовам в обслуговуванні використовують заходи для забезпечення доступності системи та даних, навіть у випадку технічних неполадок або кібератак.

Шифрування даних:

1) шифрування в покої (at rest) в якому дані повинні бути зашифровані при зберіганні, щоб захистити їх у випадку фізичного доступу до серверів або зберігання на носіях;
2) шифрування в русі (in transit) використовується для захисту даних під час передачі між системами за допомогою шифрування, наприклад, за допомогою протоколу HTTPS.

Захист від кібератак:

1) брандмауери та системи виявлення вторгнень використовують брандмауери та системи виявлення вторгнень для запобігання та виявлення кібератак;
2) регулярні оновлення та патчі постійно оновлюють програмне забезпечення та внесення патчів для закриття відомих уразливостей.

Захист від внутрішніх загроз:

1) авторизація та аутентифікація користувачів забезпечую те, щоб лише авторизовані користувачі мали доступ до системи, аутентифікація на основі паролів, біометричних даних, двофакторної аутентифікації тощо;
2) обмеження прав доступу присвоює користувачам мінімально необхідних прав доступу до даних та функціональності[43].

Резервне копіювання та відновлення даних. Регулярні резервні копії забезпечують регулярного резервного копіювання медичних даних для можливості відновлення у випадку втрати або пошкодження[5].

Ці вимоги допомагають забезпечити високий стандарт безпеки та захисту даних в електронних медичних системах, щоб забезпечити конфіденційність та цілісність медичної інформації пацієнтів.

1.4 Система Helsi – переваги та недоліки

"Helsi.me" - це інформаційна система для пацієнтів, призначена для полегшення доступу пацієнтів до їхньої медичної інформації та управління здоров'ям. Ось переваги та недоліки такої системи:

Переваги "Helsi.me":

1) зручний доступ до медичної інформації забезпечують пацієнтам можливість легко переглядати свої медичні записи, результати аналізів, діагнози та іншу інформацію в одному центрі;
2) моніторинг здоров'я надає системі можливості для ведення денних журналів здоров'я, фіксації прийому ліків та інших медичних параметрів;
3) нагадування і планування надає пацієнтам можливість отримувати нагадування про прийом ліків, відвідування лікаря та інші медичні заходи;
4) зручне спілкування зі спеціалістами надає пацієнтам можливість легко спілкуватися зі своїми лікарями, питати питання та отримувати консультації в онлайн-форматі;
5) доступність даних з будь-якого пристрою може бути доступна через веб-браузер або мобільний додаток, що дозволяє пацієнтам переглядати свою медичну інформацію з будь-якого пристрою з Інтернетом.

Недоліки "Helsi.me":

1) приватність та безпека даних зберігає та передає медичну інформацію пацієнтів в інтернеті може підвищувати ризик порушення конфіденційності та кібератак;
2) не всі медичні установи інтегруються, деякі лікарні і лікарі можуть не бути підключені до системи "Helsi.me", що обмежує її корисність для пацієнта;
3) потрібен доступ до Інтернету, що може бути недоступним в деяких регіонах або для деяких користувачів.

4) пацієнти можуть потребувати часу, щоб оволодіти користуванням системою та розуміти її можливості;
5) залежність від технології, пацієнти можуть відчувати себе залежними від технології та системи "HealSi.me" для управління своїм здоров'ям.

Загалом, "Helsi.me" та подібні системи можуть мати значущий потенціал для покращення доступності медичної інформації та управління здоров'ям пацієнтів, але важливо враховувати і розв'язувати проблеми безпеки та конфіденційності даних для забезпечення захисту особистої інформації пацієнтів.

1.5 Висновки за розділом 1

За допомогою інформаційних систем для пацієнтів, таких як "HealSi.me", маємо можливість відкривати нові можливості для пацієнтів у керуванні своїм здоров'ям та отриманні доступу до медичної інформації. Переваги цих систем включають зручний доступ до даних, моніторинг здоров'я, нагадування та зручне спілкування зі спеціалістами. Вони сприяють покращенню якості надання медичних послуг і можуть зробити взаємодію між пацієнтами та медичними установами більш ефективною.

Проте важливо пам'ятати, що такі системи також стикаються з низкою викликів і обмежень. Один із найбільших з них - це приватність та безпека даних, оскільки зберігання медичної інформації в інтернеті може призвести до загроз порушення конфіденційності та втрати персональних даних. Крім того, не всі медичні установи можуть інтегрувати такі системи, що обмежує їхню загальну корисність для пацієнтів.

Успіх медичних інформаційних систем залежить від вирішення вказаних викликів і належної охорони даних. Забезпечення безпеки та конфіденційності даних повинно мати високий пріоритет. Крім того, необхідно забезпечити

доступність систем для різних категорій користувачів і забезпечити навчання пацієнтів щодо їх використання.

Усе це показує, що інформаційні системи для пацієнтів мають великий потенціал для поліпшення охорони здоров'я, але їхнє впровадження та використання повинні супроводжуватися ретельним плануванням та заходами забезпечення безпеки, щоб забезпечити максимальну користь для пацієнтів і лікарів.

2 РЕАЛІЗАЦІЯ ПРИНЦИПІВ ЗАХИСТУ ІНФОРМАЦІЇ В ЕЛЕКТРОННИХ МЕДИЧНИХ СИСТЕМАХ

2.1 Огляд Принципів Захисту Інформації

У цьому розділі здійснено глибокий аналіз принципів захисту інформації, специфічно в контексті електронних медичних систем.

2.1.1 Основні Вимоги до Конфіденційності, Цілісності та Доступності

Принципи конфіденційності, цілісності та доступності визначають найбільш критичні аспекти безпеки медичної інформації в електронних медичних системах.

Конфіденційність. Цей аспект гарантує, що медична інформація залишається доступною лише для авторизованих осіб. Застосування механізмів автентифікації та авторизації забезпечує контроль доступу до конфіденційної інформації, уникнення несанкціонованого доступу та забезпечення приватності пацієнтів[16].

Цілісність. Цілісність медичних даних важлива для того, щоб уникнути неправомірних або несанкціонованих змін. Використання хеш-функцій та цифрових підписів дозволяє виявляти будь-які небажані модифікації та забезпечує невіддільність медичної інформації.

Доступність. Забезпечення постійного та надійного доступу до інформації для авторизованих користувачів є критичним елементом. Використання механізмів резервного копіювання, реплікації даних та високодоступних архітектур гарантує, що система завжди доступна, навіть у випадку тимчасових відмов чи катастроф[17].

Всі ці аспекти взаємодіють для створення комплексної системи безпеки, що забезпечує конфіденційність, цілісність та доступність медичної

інформації в електронних медичних системах. Їхній правильний баланс є важливим для забезпечення високого рівня захисту та довіри у сфері електронної медицини.

2.1.2 Переваги Інтеграції Сучасних Технологій

Досліджено, як сучасні технології можуть взаємодіяти для забезпечення ефективного захисту інформації в електронних медичних системах. Цей розділ поглиблює розуміння того, як використання передових технологічних рішень визначає високий рівень захисту в сучасному ландшафті електронної медицини[18].

Шифрування. Використання сучасних алгоритмів шифрування дозволяє захищати медичні дані під час їхньої передачі та зберігання. Велика увага приділяється використанню асиметричного та симетричного шифрування, забезпечуючи найвищий ступінь конфіденційності.

Двофакторна аутентифікація. Впровадження системи двофакторної аутентифікації стає додатковим шаром захисту. Крім традиційного введення паролю, використання додаткових елементів, таких як SMS-коди або біометричні дані, зроблює процес аутентифікації більш надійним[19].

Моніторинг та виявлення загроз. Застосування систем моніторингу та виявлення загроз дозволяє оперативно реагувати на потенційні атаки чи аномалії в системі. Алгоритми машинного навчання дозволяють визначати ненормальні патерни та вчасно реагувати на них.

Безпечне зберігання інформації. Використання сховищ даних, таких як Postgres та MongoDb, з інтегрованими механізмами шифрування та регулярним аудитом, створює захищене середовище для зберігання медичної інформації.

Цей розділ визначає важливість інтеграції передових технологій у сфері електронної медицини для забезпечення надійного та сучасного захисту медичної інформації. Висвітлюються переваги використання таких технологій як ключового елементу в системах електронної медицини.

2.1.3 Забезпечення Захисту Інформації в Реальному Часі

В даному розділі висвітлено важливість забезпечення захисту інформації в реальному часі, особливо в контексті електронних медичних систем, де оперативний доступ до даних є критичним для якісної медичної допомоги.

Значення захисту в реальному часі. У світлі електронних медичних систем важливо забезпечити негайний та надійний доступ до інформації. Враховуючи динаміку медичних процесів, швидка реакція на загрози є критично важливою для запобігання можливим проблемам та забезпечення безпеки пацієнтів.

Аналіз реального часу. В розділі розглядаються методи та техніки аналізу інформації в реальному часі. Системи, здатні виявляти та аналізувати аномалії в реальному часі, забезпечують вчасну реакцію на події та можливі загрози.

Ефективне виявлення загроз. Розглядаються можливості використання сучасних алгоритмів та інструментів для ефективного виявлення загроз у режимі реального часу. Використання технологій штучного інтелекту та машинного навчання дозволяє автоматизувати цей процес та покращити точність виявлення[20].

Важливість оперативності. Зазначається, що в реальному часі оперативність є ключовою характеристикою для ефективного функціонування електронних медичних систем. Аналізується вплив затримок або відмов на

доступ до даних на медичний процес та як це може впливати на якість надання медичних послуг.

2.1.4 Регулювання та Стандарти Захисту Даних в Медичній Сфері

В даному розділі розглядаються регуляторні вимоги та стандарти, які є визначальними для електронних медичних систем у забезпеченні відповідності з законодавством та вірогідністю довіри з боку користувачів[21].

HIPAA (Health Insurance Portability and Accountability Act). Визначено, як система повинна відповідати стандартам HIPAA, спрямованим на захист конфіденційності та безпеки медичних інформаційних даних пацієнтів. Аналізуються вимоги до контролю доступу, аудиту та інших параметрів, які забезпечують відповідність з цим стандартом.

GDPR (General Data Protection Regulation). Розглядаються заходи, які електронна медична система повинна вживати для відповідності стандартам GDPR у забезпеченні прав користувачів на захист персональних даних та контроль над їхнім використанням[22].

Інші стандарти та регулювання. Виділено інші регуляторні вимоги та стандарти, які можуть впливати на електронні медичні системи, такі як стандарти для забезпечення безпеки обробки платіжних даних або стандарти для забезпечення інтегрованості з системами обміну даними в медичній сфері.

Заходи з відповідності та аудиту. Розглянуті заходи, які система може впроваджувати для постійного відстеження та оцінювання своєї відповідності до регуляторних вимог. Аудит та системи моніторингу грають ключову роль у забезпеченні постійної відповідності та ідентифікації можливих відхилень.

2.1.5 Аудит та Моніторинг заходів Захисту

В цьому розділі детально розглядається важливість проведення аудиту та моніторингу заходів захисту в реальному часі в електронних медичних системах. Особлива увага приділяється системам, які можуть автоматично виявляти аномалії та негайно реагувати на можливі загрози[23].

Системи аудиту та моніторингу. Розглядаються сучасні системи аудиту та моніторингу, спроектовані спеціально для електронних медичних систем. Аналізуються їхні здатності виявляти незвичайні патерни, надто активні або неправомірні дії в системі.

Автоматична реакція на загрози. Визначено, як системи моніторингу можуть автоматично реагувати на виявлені загрози. Розглядаються можливості використання штучного інтелекту та алгоритмів машинного навчання для негайного реагування на потенційно небезпечні ситуації[24].

Часові рамки та періодичність аудиту. Обговорюється необхідність встановлення часових рамок та періодичності аудиту. Враховуються особливості медичного середовища, де інформація може змінюватися дуже швидко.

Забезпечення цілісності та безпеки аудитних журналів. Розглядається важливість забезпечення цілісності та безпеки аудитних журналів. Детально вивчається, як система може захистити цю інформацію від неправомірного доступу чи модифікацій.

Реактивні та превентивні заходи. Обговорюється баланс між реактивними та превентивними заходами, які можуть бути впроваджені на основі результатів аудиту та моніторингу. Підкреслюється важливість не тільки виявлення загроз, але й узгоджених заходів для їх запобігання.

2.2 Висновок за розділом 2

У цьому розділі було ретельно досліджено та висвітлено переваги інтеграції сучасних технологій в електронних медичних системах для забезпечення надійного та ефективного захисту медичної інформації. Шифрування, двофакторна аутентифікація, моніторинг та виявлення загроз, а також безпечне зберігання інформації є ключовими складовими системи захисту, які забезпечують конфіденційність та цілісність медичних даних.

Особлива увага була приділена важливості забезпечення захисту інформації в реальному часі в контексті електронних медичних систем. Швидка реакція на загрози та ефективне виявлення їх у режимі реального часу визначаються як критичні для якісного надання медичних послуг та безпеки пацієнтів.

Розділ також розглянув регулювання та стандарти захисту даних в медичній сфері, зокрема в контексті HIPAA та GDPR. Виконання цих стандартів визначає високий рівень довіри з боку користувачів та відповідність з законодавством.

Нарешті, обговорено важливість проведення аудиту та моніторингу заходів захисту в реальному часі. Сучасні системи аудиту та моніторингу, спроектовані спеціально для електронних медичних систем, дозволяють виявляти аномалії та негайно реагувати на можливі загрози, забезпечуючи постійну відповідність та високий рівень безпеки.

У висновку можна сказати, що інтеграція передових технологій, дотримання стандартів безпеки та постійний моніторинг є важливими елементами для створення ефективних та надійних електронних медичних систем, які забезпечують високий рівень захисту медичної інформації.

3 МЕТОДИ ЗАСТОВНІ В ЕЛЕКТРОННИХ МЕДИЧНИХ СИСТЕМАХ

3.1 Методи збору та введення даних в ЕМС і безпека вхідних даних

Збір і введення даних в електронну медичну систему (ЕМС) є критичними процесами, які вимагають високого рівня точності, безпеки та надійності. Ось деякі методи збору та введення даних в ЕМС, а також заходи безпеки для вхідних даних[6]:

Ручне введення даних:

1) лікарські записи - медичний персонал вручну вводить дані пацієнта, такі як симптоми, діагнози, лікування тощо;
2) лабораторні результати - лабораторні техніки вводять результати аналізів та інші лабораторні дані;
3) інші документи - документи, які містять медичну інформацію, такі як звіти про обстеження, також можуть бути вручну введені в ЕМС;
4) безпека використовується для запобігання помилкам та забезпечення точності даних, ЕМС може використовувати перевірки на достовірність даних, подвійне перевіряти важливі дані, а також використовувати системи контролю якості даних.

Автоматичний збір даних[7]:

1) медичні прилади, такі як монітори вітальних показників, автоматично збирають дані та передають їх в ЕМС;
2) системи обробки зображень можуть автоматично аналізувати зображення з медичних обстежень, таких як рентгенограми або МРТ, і вводити дані в ЕМС.

Забезпечення цілісності даних та впевненість у правильності вимагає застосування шифрування даних, перевірки цифрових підписів та заходів безпеки в медичних системах[8]

Інтеграція з іншими системами:

1) лабораторії та аптеки інтегрують з системами лабораторій і аптек, що дозволяє автоматично вводити результати аналізів і рецепти ліків;
2) телемедицина надає дані з віддалених консультацій можуть автоматично вводитися в ЕМС.

Забезпечення захисту даних під час їх передачі між різними системами є важливою частиною безпеки вхідних даних.

Системи відстеження ідентичності аутентифікація пацієнтів використання систем аутентифікації, таких як біометричні дані, картки доступу або паролі, для перевірки ідентичності пацієнта перед введенням даних [44] (рис. 3.1).

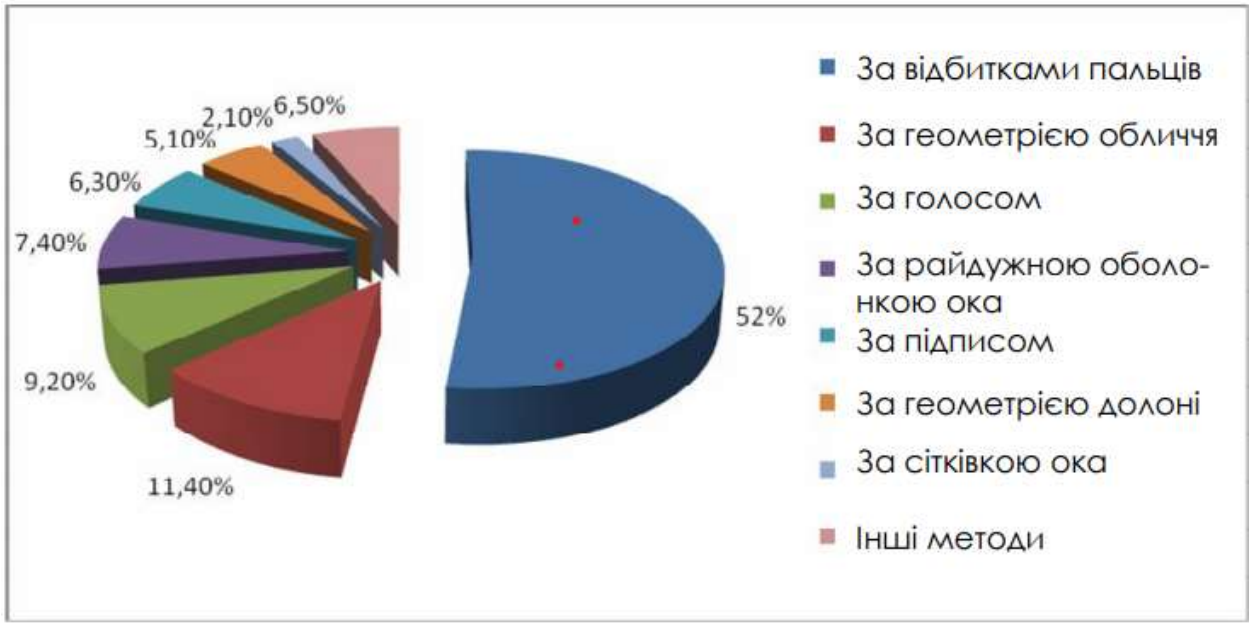


Рисунок 3.1 - Системи відстеження ідентичності аутентифікація

Забезпечення безпеки вхідних даних вимагає надійних систем аутентифікації та контролю доступу.

Збір і введення даних в ЕМС - це складний процес, який потребує не тільки точності, але і високого рівня безпеки, оскільки медична інформація є чутливою і конфіденційною. Забезпечення безпеки вхідних даних включає в себе широкий спектр заходів від контролю доступу до шифрування та перевірки цифрових підписів, щоб забезпечити цілісність та конфіденційність цих даних (рис. 3.2).

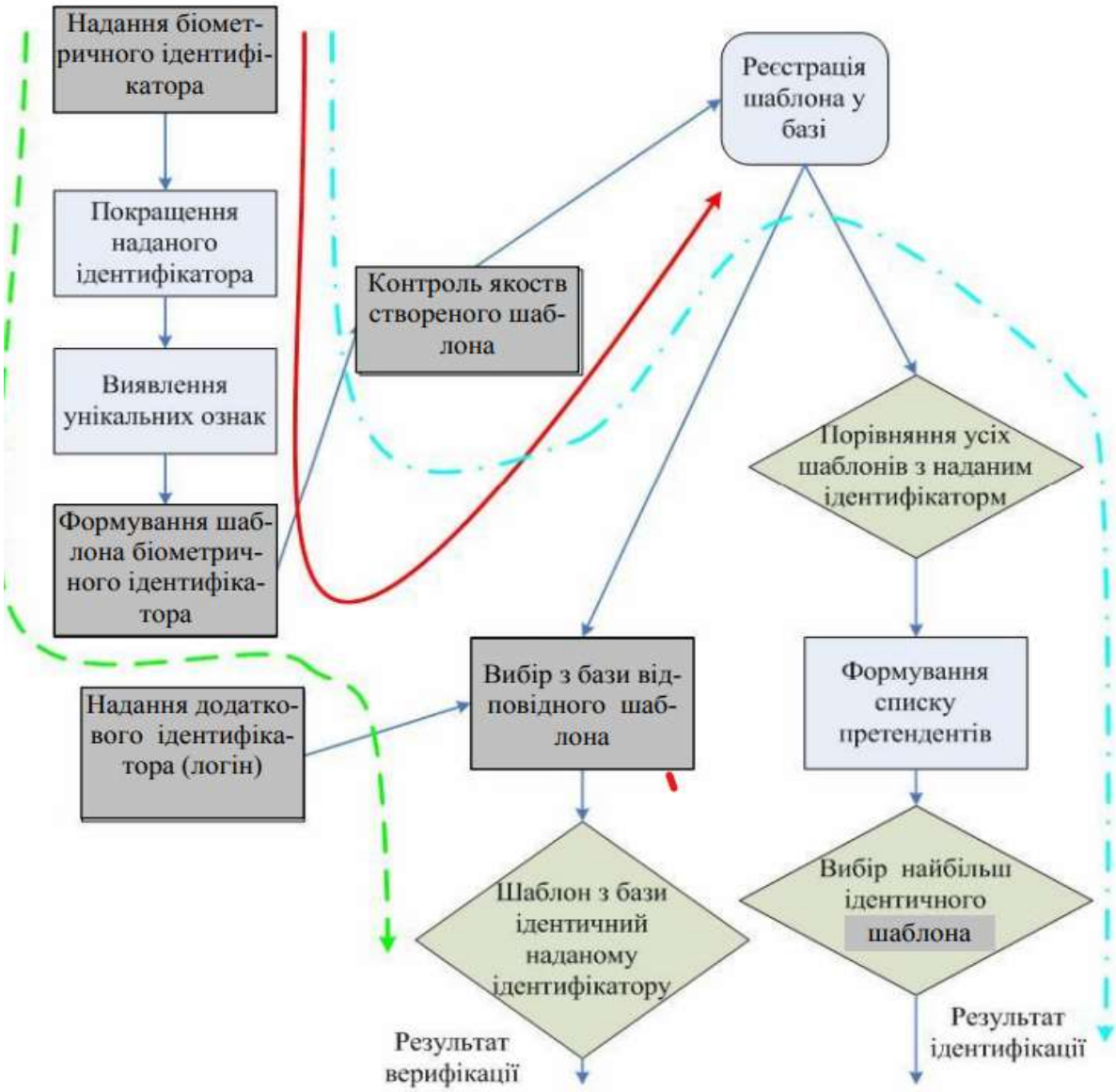


Рисунок 3.2 - Загальна схема функціонування системи біометричної ідентифікації

3.2 Методи обробки даних та їх безпека

Обробка даних в електронних медичних системах (ЕМС) включає в себе різні методи та процеси для аналізу, зберігання та використання медичної інформації. Тут розглядаються основні методи обробки даних та заходи безпеки, пов'язані з ними[9]:

Методи обробки даних:

1) аналіз та агрегація даних. Обробка великих обсягів медичної інформації для виділення корисних патернів, трендів та статистичних аналізів. Цей метод допомагає виявити патерни хвороб, ефективність лікування та інші клінічні залежності;
2) інтеграція даних. Поєднання інформації з різних джерел, таких як лабораторні дані, зображення, лікарські записи, для створення повної картини стану пацієнта;
3) класифікація та категоризація даних. Групування даних за певними параметрами для полегшення аналізу та доступу до них;
4) машинне навчання і штучний інтелект. Використання алгоритмів машинного навчання та штучного інтелекту для автоматизації аналізу даних, виявлення аномалій, прогнозування захворювань та інші цілі;
5) створення звітів і документації. Автоматизована генерація медичних звітів та документації для полегшення процесу медичної документації[45].

Заходи безпеки обробки даних:

1) шифрування даних. Використання шифрування для захисту даних в спокої (at rest) та під час передачі (in transit);
2) контроль доступу. Обмеження доступу до медичних даних лише авторизованим користувачам. Використання системи ролей та прав доступу;

3) аудит доступу. Логування та моніторинг доступу до даних з можливістю аналізу відходів та виявлення незаконного доступу;
4) фізична безпека серверів і обладнання. Захист серверних приміщень та обладнання від несанкціонованого доступу;
5) резервне копіювання і відновлення даних. Регулярне створення резервних копій даних для можливості відновлення в разі втрати або пошкодження;
6) патчі та оновлення. Регулярне оновлення програмного забезпечення та внесення патчів для закриття відомих уразливостей;
7) навчання персоналу. Навчання медичного та адміністративного персоналу щодо правил безпеки та обробки даних[46].

3.3 Методи захисту каналів передачі медичних даних.

Захист каналів передачі медичних даних є критичним завданням для забезпечення конфіденційності, цілісності та доступності цих даних під час їхнього передачі в мережі[10].

Ось деякі методи та заходи для захисту каналів передачі медичних даних:

1) шифрування даних. TLS/SSL (Transport Layer Security/Secure Sockets Layer): Використання протоколів шифрування TLS або SSL для захисту даних під час їхньої передачі через Інтернет. Це особливо важливо при віддалених консультаціях та телемедицині;
2) віртуальні приватні мережі (VPN). Встановлення VPN-з'єднань між медичними установами та іншими точками доступу для створення шифрованого тунелю для передачі даних. VPN забезпечує безпеку та конфіденційність мережевого зв'язку[11];

3) захищені мережеві протоколи. Використання безпечних мережевих протоколів, таких як IPsec (Internet Protocol Security), для забезпечення шифрування та ідентифікації джерела даних;
4) аутентифікація та авторизація. Забезпечення двоступеневої аутентифікації для користувачів, які отримують доступ до медичних даних, і встановлення строгих прав доступу для обмеження можливостей користувачів;
5) брандмауери та інтрузійний виявлення. Використання брандмауерів та систем інтрузійного виявлення для моніторингу мережевого трафіку та виявлення незвичайної або підозрілої активності;
6) фізичний захист кабелів і обладнання. Захист кабелів передачі даних та мережевого обладнання від фізичного доступу несанкціонованих осіб[12];
7) захист від вірусів і зловмисного програмного забезпечення. Використання антивірусного програмного забезпечення та інших заходів для захисту від зловмисних програм, які можуть спробувати атакувати мережу;
8) шифрування точки-точки (End-to-End Encryption). Використання шифрування, яке застосовується на рівні кінцевих точок (наприклад, на мобільних пристроях або на комп'ютерах пацієнтів), щоб навіть провайдери мережі не могли декодувати передані дані;
9) захист від отрутоносних посилань (Zero-Day Exploits). Постійна актуалізація і вдосконалення мережевого обладнання та програмного забезпечення для запобігання використанню вразливостей "нульового дня"[47].

Заходи безпеки для захисту каналів передачі медичних даних мають захищати ці дані від несанкціонованого доступу, модифікації або витоку. Це особливо важливо в контексті медицини, оскільки медична інформація є вкрай чутливою і підлягає високим стандартам конфіденційності та безпеки.

3.4 Метод федеративного навчання для аналізу медичних даних схеми їх методів та прив’язки до Helsi

Метод федеративного навчання (Federated Learning) є важливим інструментом для аналізу медичних даних, особливо в контексті системи, подібної до "Helsi." Цей метод дозволяє об'єднувати дані від різних джерел або користувачів, для навчання моделей машинного навчання, не передаючи самі дані централізовано. Ось схема та методи використання федеративного навчання в контексті "Helsi"[13]:

Схема використання федеративного навчання в "Helsi":

1) збір даних. Медичні дані, такі як клінічні записи, зображення, лабораторні дані, вводяться в систему "Helsi" від різних пацієнтів і джерел;
2) обробка та анонімізація. Дані анонімізуються та підготовлюються для федеративного навчання. Вони можуть бути розділені на частини для подальшого навчання моделей;
3) навчання моделей локально. Кожен користувач системи "Helsi" може навчати свою локальну модель машинного навчання на основі своїх медичних даних, не відправляючи їх централізовано. Це забезпечує конфіденційність даних пацієнтів;
4) агрегація моделей. Після навчання моделей локально, параметри цих моделей агрегуються на сервері "Helsi" для створення глобальної моделі;
5) використання глобальної моделі. Глобальна модель може використовуватися для різних завдань, таких як передбачення захворювань, аналіз зображень, планування лікування тощо;
6) періодичне оновлення моделі. Федеративне навчання може бути періодичним процесом, де локальні моделі навчаються на нових

даних індивідуальних користувачів, і ці зміни агрегуються в глобальну модель для її постійного вдосконалення[14].

Методи використання федеративного навчання в "Helsi":

1) аналіз клінічних записів. Федеративне навчання може використовуватися для створення моделей, які аналізують клінічні записи пацієнтів і допомагають лікарям визначати діагнози та рекомендувати лікування;
2) аналіз медичних зображень. Глобальна модель, навчена на основі федеративного навчання, може використовуватися для аналізу медичних зображень, таких як рентгенограми, МРТ або знімки КТ;
3) персоналізована медицина. Федеративне навчання дозволяє створювати персоналізовані моделі для прогнозування ризику захворювань та розроблення індивідуальних лікувальних планів для кожного пацієнта;
4) попередження епідемій і дослідження публічного здоров'я. Федеративне навчання може використовуватися для вчасного виявлення та відстеження епідемій та захворювань шляхом аналізу симптомів та розповсюдження хвороб[15].

Прив’язки до "Helsi":

"Helsi" може використовувати федеративне навчання для створення моделей, які аналізують медичні дані, зібрані від різних пацієнтів, і надають індивідуальні рекомендації щодо лікування та догляду за здоров'ям.

Важливо враховувати питання конфіденційності і захисту даних пацієнтів при використанні федеративного навчання, оскільки цей метод вимагає безпечного оброблення і агрегації даних з різних джерел.

3.5 Висновки за розділом 3

У процесі збору та введення медичних даних в електронні медичні системи (ЕМС) використовуються методи, які вимагають високого рівня точності та безпеки. Основні методи включають ручне введення даних, автоматичний збір даних, інтеграцію з іншими системами та системи відстеження ідентичності. З метою безпеки вхідних даних, важливо використовувати перевірки на достовірність даних, контроль якості, шифрування даних, системи аутентифікації та контролю доступу. Забезпечення цілісності та конфіденційності медичної інформації - це критичний аспект у роботі ЕМС[48].

Обробка медичних даних в ЕМС включає в себе різні методи, такі як аналіз, агрегація, інтеграція та застосування штучного інтелекту. Для забезпечення безпеки даних, важливо використовувати шифрування, контроль доступу, аудит доступу, фізичний захист обладнання та резервне копіювання даних. Медична інформація має бути оброблена з дотриманням найвищих стандартів конфіденційності та безпеки.

Захист каналів передачі медичних даних є критичним для забезпечення конфіденційності, цілісності та доступності цих даних під час їхньої передачі в мережі. Основні методи включають в себе шифрування даних, використання віртуальних приватних мереж (VPN), захищених мережевих протоколів, аутентифікацію та авторизацію, фізичний захист мережевого обладнання та контроль доступу. Ці заходи забезпечують безпеку медичних даних під час їхньої передачі[49].

Федеративне навчання - це метод, який дозволяє аналізувати медичні дані з різних джерел, не розголошуючи самі дані централізовано. В контексті системи "Helsi," федеративне навчання може бути використано для створення моделей машинного навчання, які аналізують дані від різних пацієнтів та надають індивідуальні рекомендації щодо лікування та догляду. Головні методи включають в себе збір та анонімізацію даних, локальне навчання

моделей, агрегацію моделей та періодичне оновлення моделі. Федеративне навчання забезпечує конфіденційність даних пацієнтів та дозволяє створювати персоналізовані рішення для кожного пацієнта.

Таким чином інтелектуальний агент може досягти кілька важливих цілей:

1. Медичні консультації та поради:

1) Здатність надавати базові консультації та поради з певних медичних питань.

2) Вміння рекомендувати заходи самодопомоги або вказувати на необхідність звертання до лікаря.

2. Спостереження за станом здоров'я:

1) Можливість відслідковувати певні показники здоров'я користувача.

2) Повідомлення про рекомендації та заходи щодо поліпшення стану здоров'я[50].

3. Повідомлення про ліки та лікування:

1) Надання інформації про призначені ліки.

2) Нагадування про прийом ліків та проведення лікування.

4. Постійне навчання:

1) Можливість систематичного оновлення бази даних та алгоритмів на основі нових медичних досліджень та відгуків користувачів.

2) Впровадження штучного інтелекту для постійного вдосконалення якості відповідей та порад.

5. Конфіденційність та безпека:

1) Забезпечення високого рівня конфіденційності медичної інформації.

2) Захист від несанкціонованого доступу та зловживань.

6. Інтерактивні можливості:

1) Можливість взаємодії з користувачем у формі діалогу.

4 Розуміння і врахування індивідуальних особливостей користувачів.

4 ТЕХНОЛОГІЇ РОЗРОБКИ ЕМС

4.1 Технології Java для Розробки Електронних Медичних Систем

У цьому розділі розглядаються переваги та особливості використання мови програмування Java в контексті розробки електронних медичних систем.

Висока надійність. Java відзначається високим рівнем надійності завдяки своїм вбудованим механізмам обробки помилок та автоматичному управлінню пам'яттю. Це особливо важливо в контексті електронних медичних систем, де точність та стабільність виконання коду впливають на безпеку та ефективність системи.

Переносимість коду між платформами. Java пропонує велику переносимість коду, що означає, що програми, написані на Java, можуть працювати на різних платформах без змін. Це є важливим фактором у відношенні до електронних медичних систем, оскільки вони можуть використовуватися на різних пристроях та операційних системах[25].

Багатофункціональність. Java надає розгалужені інструменти та бібліотеки, що полегшують створення різноманітних функцій та модулів в електронних медичних системах. Це включає в себе підтримку роботи з базами даних, мережевими операціями, графічним інтерфейсом користувача та інші ключові компоненти.

Стабільність та продуктивність. Java відома своєю високою продуктивністю та оптимізацією ресурсів. Це дозволяє створювати великі та складні електронні медичні системи, які працюють ефективно та без перебоїв.

Застосування в інтерфейсах користувача. JavaFX та Swing - фреймворки для створення графічного інтерфейсу користувача в Java, дозволяють

розробникам легко створювати інтуїтивно зрозумілі та естетично зовнішні інтерфейси для електронних медичних систем[26].

Широке сприйняття в галузі розробки. Багато розробників мають досвід роботи з Java, що полегшує формування команд для розробки та підтримки електронних медичних систем.

4.2 Spring Boot для реалізації бекенду електронних медичних систем

Spring Boot є потужним фреймворком для розробки бекенду в Java-додатках, і його використання в електронних медичних системах надає низку переваг для створення ефективних та безпечних рішень.

Легкість створення RESTful сервісів. Spring Boot надає прості та ефективні інструменти для створення RESTful сервісів. Його анотації та конфігураційні можливості дозволяють розробникам легко визначати ендпоінти, обробляти HTTP-запити та взаємодіяти з клієнтами[27].

Безпека. Spring Security, що є частиною екосистеми Spring, дозволяє ефективно впроваджувати механізми аутентифікації та авторизації. Це важливо в контексті електронних медичних систем, де забезпечення конфіденційності та доступу до медичної інформації є критично важливим.

Інтеграція з базами даних. Spring Boot спрощує інтеграцію з різноманітними базами даних. Використання Spring Data JPA дозволяє розробникам легко взаємодіяти з базами даних, використовуючи об'єктно-реляційне відображення (ORM)[28].

Масштабованість. Створення масштабованих систем є однією з ключових переваг Spring Boot. Його інтеграція з іншими елементами екосистеми Spring, такими як Spring Cloud для управління конфігурацією та

реєстрацією служб, робить його ідеальним вибором для великих та розподілених систем.

Спрощене конфігурування. Spring Boot пропонує конфігурацію за замовчуванням, що робить процес створення та налаштування бекенду більш простим та зрозумілим.

Розширюваність. За допомогою модульної архітектури та можливостей інтеграції з різними бібліотеками та фреймворками, Spring Boot дозволяє розробникам розширювати функціональність та впроваджувати нові можливості в систему[29].

Загальною метою використання Spring Boot в електронних медичних системах є надання швидкої, безпечної та масштабованої архітектури бекенду для забезпечення надійності та ефективності системи.

4.2.1 Розробка "Інтелектуального Агента"

У цьому підрозділі проведено розгляд розробки інтелектуального агента для взаємодії з електронною медичною системою. Розглянемо конфігурацію Spring Boot Security та налаштування для взаємодії з Telegram Bot API через BotConfig[30].

Конфігурація Spring Boot Security. Для забезпечення безпеки інформації в електронних медичних системах використовуємо Spring Boot Security (рис. 4.1). Конфігурація SecurityConfig буде відповідати за визначення правил доступу і обробку аутентифікації.

Рисунок 4.1 - UML діаграма configs spring boot

Опис класу WebSecurityConfig:

1) клас, що налаштовує безпеку веб-додатка;
2) passwordEncoder getPasswordEncoder(). Метод для створення об'єкта PasswordEncoder з використанням BCrypt;
3) securityFilterChain securityFilterChain(HttpSecurity http). Метод для налаштування безпеки через об'єкт HttpSecurity, відключає захист CSRF.

4.3.1.2 Налаштування Telegram Bot через BotConfig

Для взаємодії з Telegram Bot API створимо клас BotConfig, який міститиме налаштування інтелектуального агента[31].

Опис класу BotConfig (рис. 4.2):

1) конфігураційний клас для налаштування інтелектуального агента та інших ресурсів;
2) string botUsername, botToken, webHook. Змінні для зберігання параметрів агента (ім'я, токен, вебхук);

3) @Bean public PokrmedBot bot(TelegramFacade telegramFacade). Метод для створення об'єкта агента для взаємодії з Telegram;
4) @Bean public ReloadableResourceBundleMessageSource messageSource(). Метод для створення об'єкта ReloadableResourceBundleMessageSource для роботи з мовними ресурсами;
5) @Bean public MessageSourceAccessor messageSourceAccessor(). Метод для налаштування об'єкта MessageSourceAccessor для роботи з мовними ресурсами з кодуванням UTF-8.

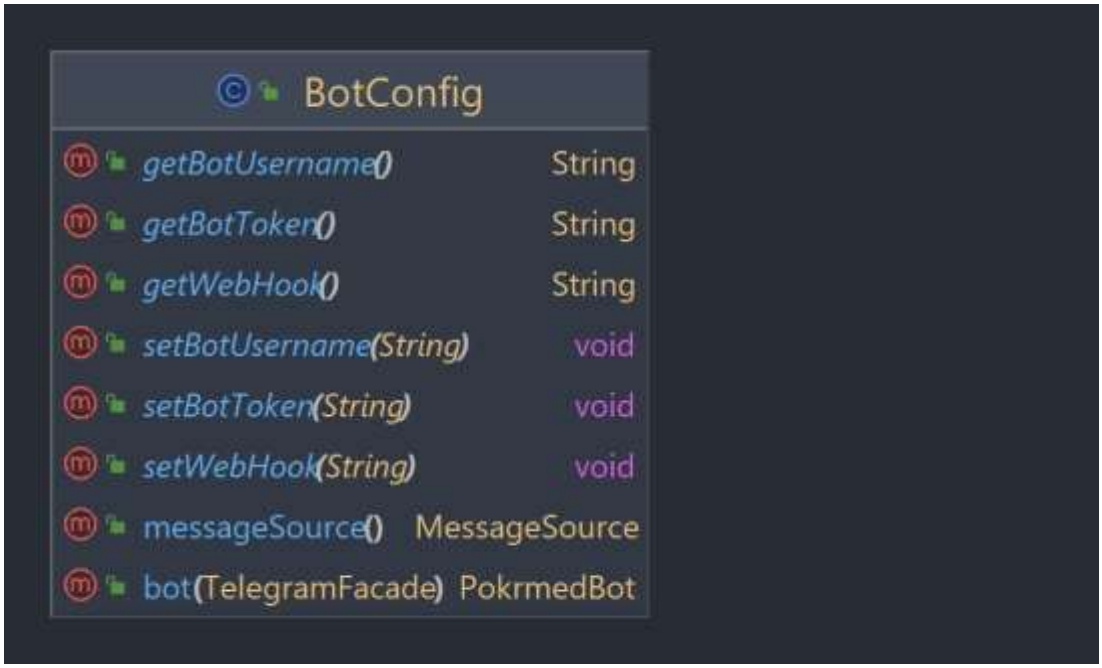


Рисунок 4.2 – Клас BotConfig

У цих класах (рис. 4.3) ми виконуємо оброку повідомлень від користувачів за допомогою методу handleUpdate:

1) handleUpdate(Update update). Метод для обробки отриманих оновлень від Telegram та надсилання відповідей;
2) if (update.hasCallbackQuery()). Перевірка, чи є в отриманому оновленні callback-запит, і якщо так, то виклик методу для обробки цього запиту;

3) message message = update.getMessage(). Отримання повідомлення з оновлення;
4) if (message != null && (message.hasText() || message.hasContact())). Перевірка наявності текстового повідомлення чи контакту;
5) final Document document = update.getMessage().getDocument(). Отримання документу (файлу) з повідомлення;
6) return replyMessage. Повернення відповіді інтелектуального агента.

Саме цей метод надає нам можливість зручно налаштувати обробку відповідей використовуючи handlers.

Рисунок 4.3 - Класи для обробки повідомлень які ми отримали

За допомоги цих контролерів (рис. 4.4) ми приймаємо кожну дію від користувача.

Рисунок 4.4 – контролери

Розглянемо приклади зображені на рис. 4.5, 4.6.

```
public BotState handleMessage(Message mess, Contact contact) {
        String message = mess.getText();
        BotState botState;
        switch (message) {
            case "/start":
                botState = BotState.START;
                break;
            case "Add contact":
                createUserOnDb.create(mess.getChatId(), mess.getFrom().getUserName(), mess.getFrom()
                botState = BotState.NEW_PASSWORD;
                break;
            default:
                if (userDataCache.getUsersCurrentBotState(mess.getFrom().getId()) != null) {
                    botState = userDataCache.getUsersCurrentBotState(mess.getFrom().getId());
                } else {
                    botState = BotState.NOT_MESSAGE_HANDLER;
                }
                break;
```

Рисунок 4.5 – Приклад програмного коду

Ми очікуємо першої дії від користувача, отримаємо команду страт коли користувача натиснув кнопку старт і далі робимо обробку у нашому StartHendler

```
BotState botState = userDataCache.getUsersCurrentBotState(userId);
if (botState.equals(BotState.CHECK_REG)) {
    if (userByTgId.isEmpty()) {
        replyToUser = messagesService.getReplyMessage(chatId, replyMessage: "check.reg.null");
        replyToUser.setReplyMarkup(buttonsMenuService.getContactKeyboard());
    } else {
        replyToUser = messagesService.getReplyMessage(chatId, replyMessage: "check.reg.access");
        userDataCache.setUsersCurrentBotState(userId, BotState.CHECK_USER_PASS);

    }
} else if (botState.equals(BotState.NEW_PASSWORD)) {
    replyToUser = messagesService.getReplyMessage(chatId, replyMessage: "new.pass");
    userDataCache.setUsersCurrentBotState(userId, BotState.SET_PASSWORD);
```

Рисунок 4.6 - Приклад програмного коду

Спочатку перевіряємо чи зареєстрований користувач, якщо ні то запускаємо умову реєстрації, а якщо користувач вже зареєстрований просимо його ввести пароль для входу та направляємо його до головного меню якщо пароль був вірний.

4.3 Інтеграція з Базами Даних: Postgres та MongoDB

Postgres та MongoDB обрані для забезпечення надійності та ефективності зберігання медичних даних. Postgres використовується для зберігання структурованих даних, тоді як MongoDB використовується для зберігання документ-орієнтованих даних. Це дозволяє оптимізувати швидкість доступу та забезпечує гнучкість у роботі з різними видами інформації[32].

PostgreSQL для Структурованих Даних:

PostgreSQL, як реляційна база даних (рис. 4.7), ідеально підходить для зберігання структурованих медичних даних, таких як інформація про пацієнтів, медичні історії, лабораторні результати та інші табличні дані. Вона забезпечує сильну консистентність та відносини між різними таблицями, що є критичним для зберігання структурованої медичної інформації. Крім того, PostgreSQL має вбудовані засоби для виконання транзакцій та підтримки складних операцій з базою даних[33].

MongoDB для Документ-Орієнтованих Даних:

MongoDB використовується для зберігання документ-орієнтованих даних, таких як зображення, файлові документи, та інші дані, які можуть бути представлені у вигляді JSON-подібних документів. Його гнучкість і можливість роботи з неструктурованою інформацією дозволяють ефективно взаємодіяти з великим обсягом недокументованих даних, що часто зустрічається в медичній сфері. MongoDB також може бути використаний для зберігання великих обсягів даних, таких як зображення з медичних зображень, що забезпечує ефективний доступ до цих ресурсів[34].

Комбінація Postgres і MongoDB дозволяє оптимально використовувати переваги обох систем зберігання даних та забезпечити повноту та різноманітність медичної інформації, що може бути важливою для різних клінічних сценаріїв та досліджень.

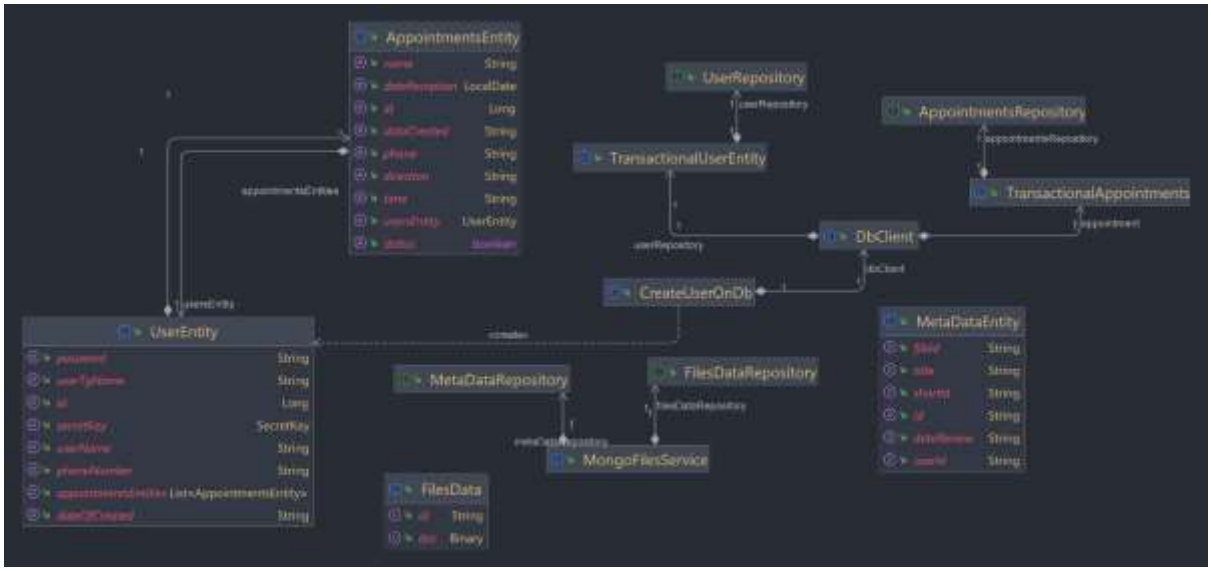


Рисунок 4.7 - Uml Persistence - взаємодія з базами даних.

Далі розглянемо конфігурації для роботи з базами.

4.3.1 Конфігурація Postgresql

Опис класу PersistenceUserAutoConfiguration:

1. @Configuration. Аннотація, що позначає клас як клас конфігурації для Spring;
2. @PropertySource({"classpath:config/multiple-db-boot.properties"}). Вказує на розташування файлу конфігурації для цього класу;
3. @EnableJpaRepositories. Активує спрощену конфігурацію JPA-репозиторіїв для інтерфейсів репозиторіїв користувачів;
4. basePackages = "com.pokrmed.persistence.user". Вказує базовий пакет для пошуку репозиторіїв користувачів;
5. entityManagerFactoryRef = "userEntityManagerFactory". Вказує на ім'я фабрики сутностей для користувачів;
6. transactionManagerRef = "userTransactionManager". Вказує на ім'я менеджера транзакцій для користувачів;

7. @Primary. Вказує, що це основне джерело даних у випадку конфлікту більше одного джерела даних;
8. @Bean. Вказує, що метод повертає об'єкт, який буде управлятися контейнером Spring.
9. @ConfigurationProperties(prefix="spring.datasource"). Зчитує властивості з файлу конфігурації для налаштування джерела даних користувачів;
10. public DataSource userDataSource(). Метод, який повертає конфігурацію джерела даних користувачів.

Опис класу TransactionalUserEntity:

1) @Service. Аннотація, що позначає клас як сервісний компонент Spring;
2) @Autowired. Автоматичне впровадження залежностей, у цьому випадку репозиторію користувачів;
3) @Transactional. Аннотація, що вказує на використання транзакцій для методів сервісу;
4) public void createUser(UserEntity userEntity). Метод для збереження нового користувача в базі даних;
5) public Optional<UserEntity> findByTgId(long id). Метод для знаходження користувача за його ідентифікатором у базі даних та повернення результату у формі Optional;
6) public UserEntity findByUserId(long id). Метод для знаходження користувача за його ідентифікатором у базі даних;
7) public List<UserEntity> findAllUsers(). Метод для отримання списку всіх користувачів у базі даних. (рис. 4.8).

```
spring.datasource.name=pokmed_bot
spring.datasource.jdbc-url=jdbc:postgresql://localhost:5432/pokmed_bot
spring.datasource.username=postgres
spring.datasource.password=root
```

Рисунок 4.8 - Properties - для підключення к БД.

4.3.2 Конфігурація MongoDB

Опис класу PersistenceFileAutoConfiguration:

1) @Configuration: Аннотація, що позначає клас як конфігураційний клас Spring;
2) @PropertySource({"classpath:config/multiple-db-boot.properties"}): Зазначає ресурс, з якого буде завантажено конфігураційні властивості;
3) @EnableMongoRepositories: Аннотація для вказівки базових пакетів репозиторіїв MongoDB;
4) @Bean(name = "primaryProperties"): Бін для властивостей конфігурації основної бази даних;
5) @ConfigurationProperties(prefix = "mongodb.primary"): Зазначає префікс властивостей конфігурації та пов'язує його з об'єктом MongoProperties;
6) @Primary: Вказує, що цей бін є пріоритетним в разі декількох бінів того ж типу;
7) @Bean(name = "primaryMongoClient"): Бін для створення клієнта MongoDB для основної бази даних;
8) public MongoClient mongoClient(@Qualifier("primaryProperties") MongoProperties mongoProperties): Метод для створення клієнта (рис. 4.10)

MongoDB з використанням конфігурації та об'єкта MongoProperties. Цей сервіс взаємодіє з MongoDB (рис. 4.9) для зберігання та отримання метаданих та файлових даних. Методи цього сервісу дозволяють зберігати, отримувати та видаляти дані з обох репозиторіїв (MetaData та FilesData)[35].

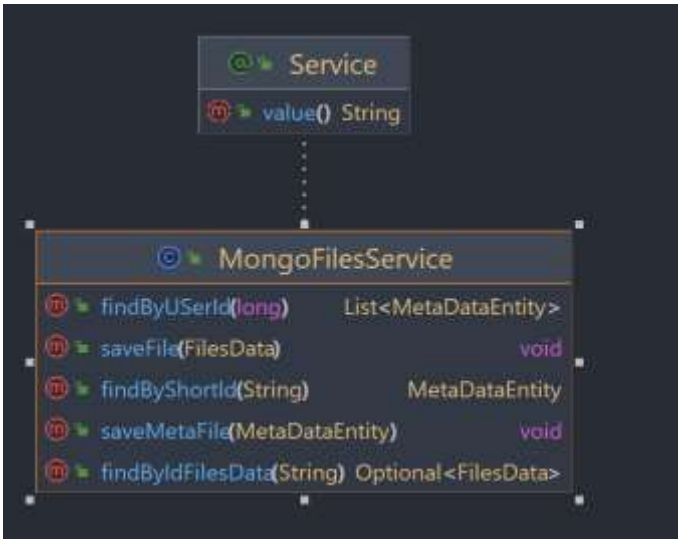


Рисунок 4.9 - UML MongoFilesService - методи для роботи з файлами.

```
mongodb.primary.uri=mongodb+srv://user:password@cluster0.9gcofhq.mongodb.net/?retryWrites=true&w=majority
mongodb.primary.database=pokrmed
mongodb.primary.authenticationDatabase=admin
mongodb.primary.username=user
mongodb.primary.password=password
mongodb.primary.port=27017
```

Рисунок 4.10 - Properties - для підключеня до БД mongo.

4.4 Tymeleaf для Реалізації Фронтенду Електронних Медичних Систем

Tymeleaf – це двигун шаблонів, спрямований на веб-розробку в середовищі Java. Використання Tymeleaf у розробці фронтенду електронних медичних систем дозволяє створювати динамічні та естетичні користувацькі інтерфейси, що забезпечують зручність взаємодії з користувачем[36].

Основні переваги використання Tymeleaf у фронтенді електронних медичних систем включають:

1) динамічні шаблони Tymeleaf (рис. 4.11) дозволяє легко вбудовувати змінні та вирази безпосередньо в HTML-шаблони, що полегшує створення динамічного вмісту на стороні клієнта;
2) зручні теги та атрибути Tymeleaf надає розширені HTML-теги та атрибути, які полегшують обробку даних та їх відображення на сторінці;
3) інтеграція з Java. Будучи спрямованим на Java, Tymeleaf забезпечує зручну інтеграцію з бекенд-частиною електронних медичних систем, сприяючи ефективній взаємодії між фронтендом та бекендом;
4) підтримка міжнародизації Tymeleaf має вбудовану підтримку міжнародизації, що робить локалізацію та інтернаціоналізацію фронтенду більш зручною;
5) легка інтеграція з Spring. Оскільки Tymeleaf розроблено для використання в Spring Framework, інтеграція з іншими частинами системи, розробленими на Spring, виходить природньою та зручною[37].

Загальною метою використання Tymeleaf у фронтенді є створення інтуїтивно зрозумілих, естетичних та функціональних інтерфейсів для поліпшення взаємодії користувачів з електронними медичними системами.

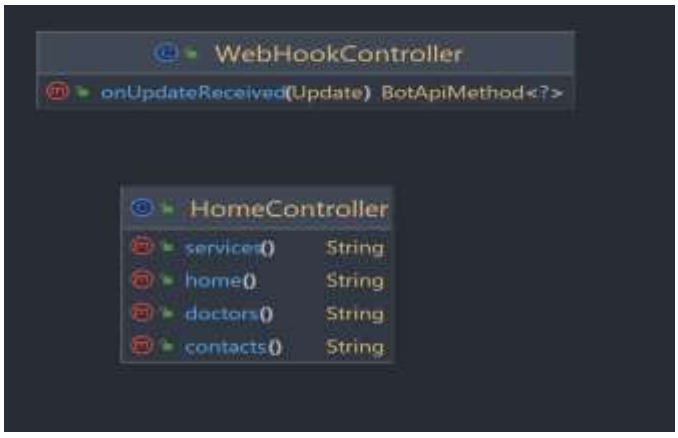


Рисунок 4.11 - Контролери для обробки шаблонів thymeleaf.

4.5 Висновки за розділом 4

У даному розділі розглянуто переваги та особливості використання мови програмування Java в контексті розробки електронних медичних систем, а також фреймворку Spring Boot для реалізації бекенду цих систем. Зазначено, що висока надійність Java, її переносимість між платформами, багатофункціональність, стабільність та продуктивність роблять цю мову ідеальним вибором для розробки електронних медичних систем.

Зокрема, JavaFX та Swing надають зручні інструменти для створення інтуїтивно зрозумілих та естетичних інтерфейсів користувача. Переваги використання Spring Boot для бекенду, такі як легкість створення RESTful сервісів, безпека, інтеграція з базами даних, масштабованість, спрощене конфігурування та розширюваність, роблять його ефективним і безпечним вибором для забезпечення надійності та ефективності електронних медичних систем.

Загальний висновок полягає в тому, що поєднання Java та Spring Boot надає розробникам зручні та потужні інструменти для створення сучасних, надійних та ефективних електронних медичних систем, забезпечуючи високий рівень безпеки, доступності та масштабованості.

5 КОНТРОЛЬ ПРАЦЕЗДАТНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Інструкція користувача для взаємодії з “Інтелектуальним агентом”

Створення інтелектуального агента в контексті розгляду розробки електронної медичної системи та його інтеграції з Spring Boot Security та базами даних Postgres та MongoDB має на меті поліпшення взаємодії користувачів з системою. Цей агент може використовуватися для надання додаткового зручного інтерфейсу для взаємодії з медичною інформацією[38].

Перш за все, агент може використовуватися для аутентифікації користувачів та управління правами доступу до медичної інформації, що регулюється Spring Boot Security. Користувачі можуть отримувати доступ до своїх медичних даних, результатів аналізів чи інших інформаційних ресурсів через взаємодію з агентом.

Додатково, агент може використовуватися для надсилання повідомлень та нагадувань про медичні процедури або важливих подій у пацієнтів. Це може покращити пацієнтський догляд та забезпечити ефективну комунікацію між пацієнтами та медичним персоналом[39].

Також, агент може стати інструментом для зручного отримання консультацій від лікарів чи надання основних медичних порад. Це може сприяти покращенню доступу до медичної інформації та послуг для користувачів. Нижче описана більш детальніша робота самого сайту[40].

Перехід з сайту за посиланням.

1. Натискання кнопки старт (рис. 5.1).

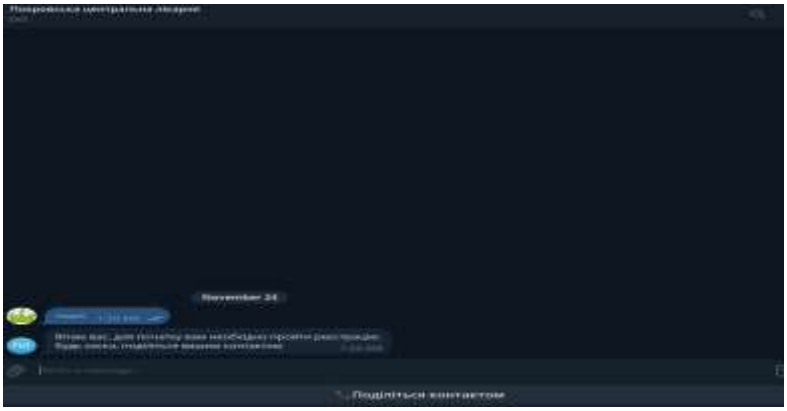


Рисунок 5.1 – Початкова сторінка

2. Реєстрація проходить за натисканням однієї кнопки.
3. Створюємо пароль (рис. 5.2) (Пароль має співпадати з усіма критеріями. Мати спец символ і тд.).
4. Після цього потрапляємо до головного меню.

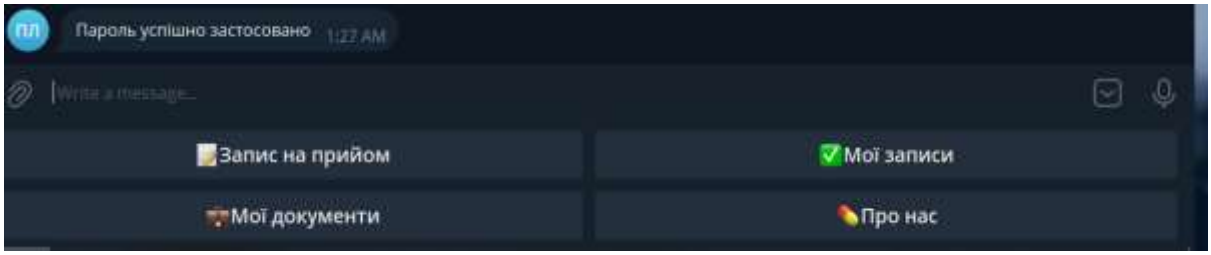


Рисунок 5.2 – Затвердження паролю

5. Коли натискаємо кнопку “Запис на прийом” запускається скрипт для запису. (рис. 5.3)

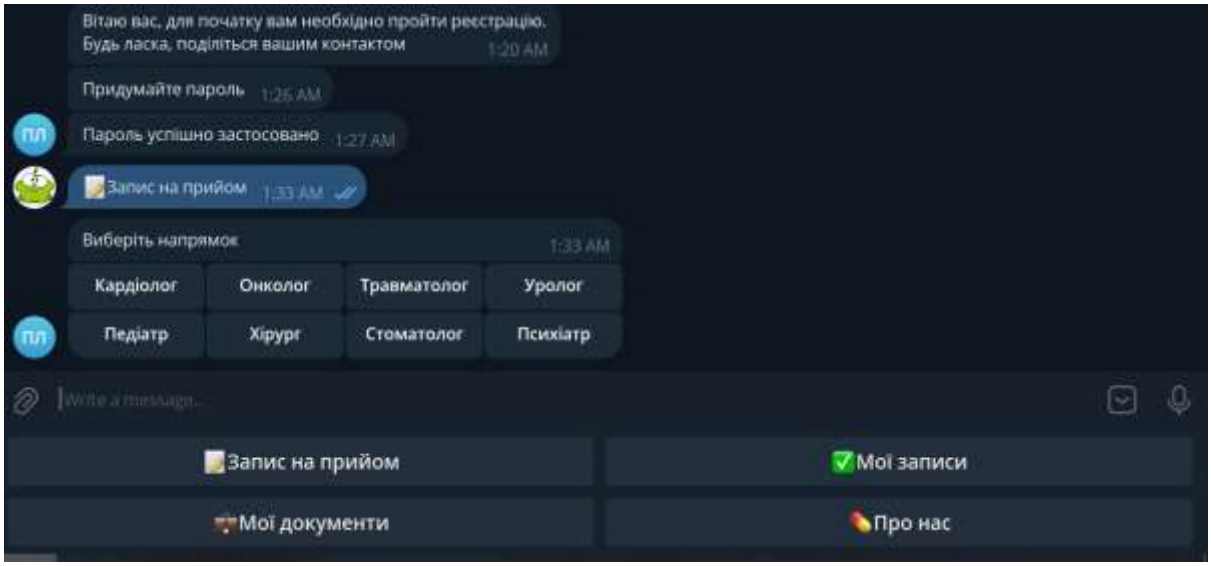


Рисунок 5.3 – Запис на прийом

6. Далі обираємо напрямок та потім дату і час. (рис. 5.4)

Рисунок 5.4 – Обирання дати та часу

7. Та після цього отримуємо сповіщення об успіху.

Якщо вибрати дату яка вже минула отримаємо про це сповіщення (рис. 5.5)

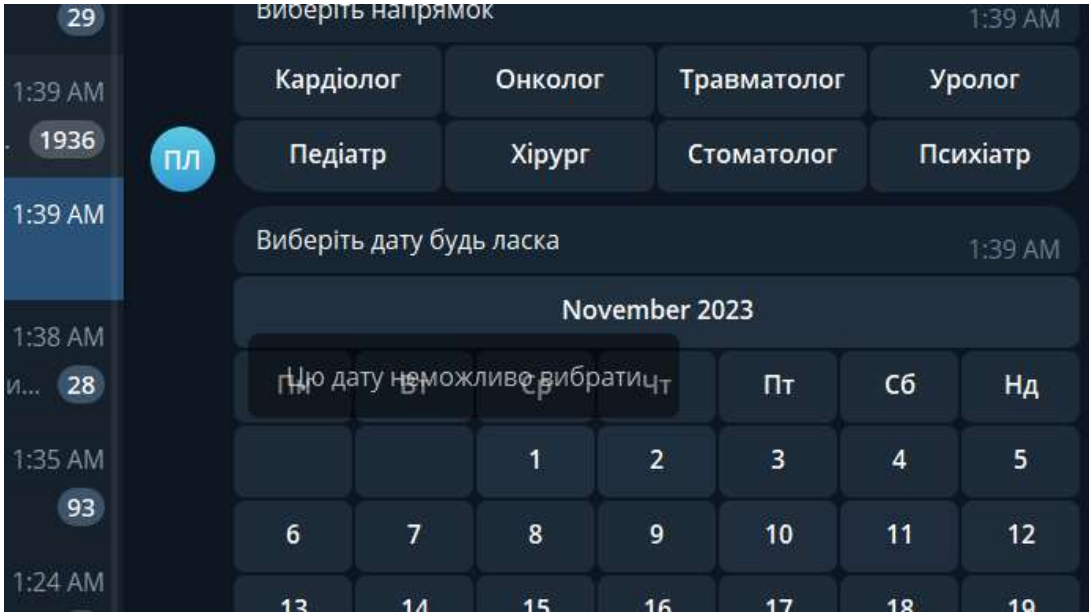


Рисунок 5.5 – Сповіщення про помилку

Неможливо створити два записи по тому самому напрямку в день

5.2 Обробка та збереження файлів

При натисканні кнопки “Мої документи” отримуємо вибір (рис. 5.6)

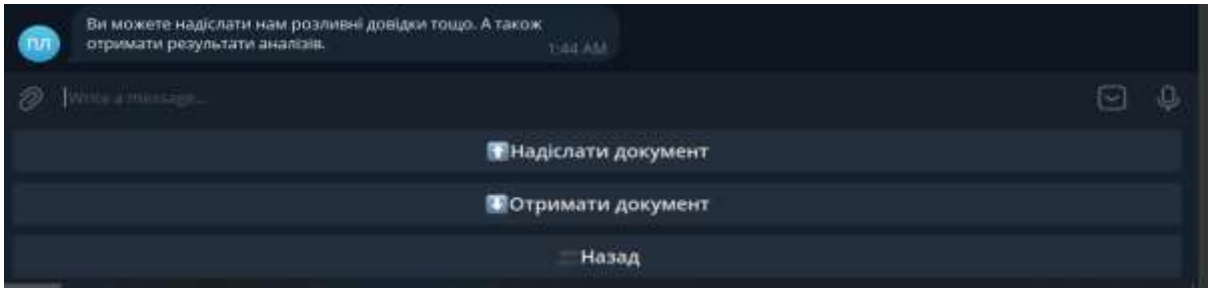


Рисунок 5.6 – Мої документи

Важливо пам’ятати що файл не може бути більше 10мб. Коли користувач надсилає файл (рис. 5.7) він автоматично зберігається в БД у зашифрованому виді (рис. 5.8), за допомоги RSA шифрування.

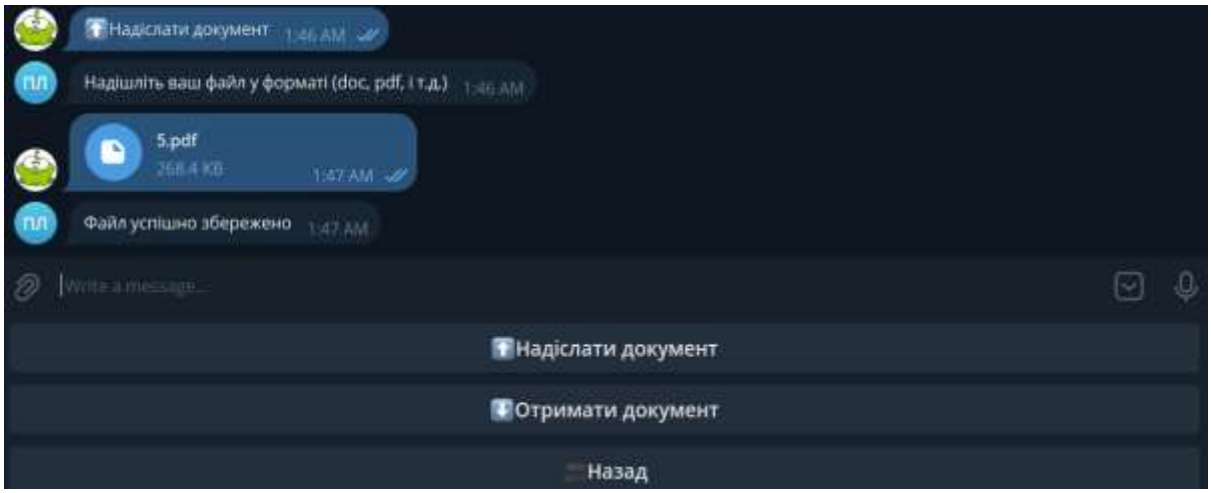


Рисунок 5.7 – Надсилання файлу

Рисунок 5.8 – Зашифрований вигляд

Якщо натискається кнопка отримати файл то проводиться операція дешифрування файла і відправка користувачу (рис. 5.9).

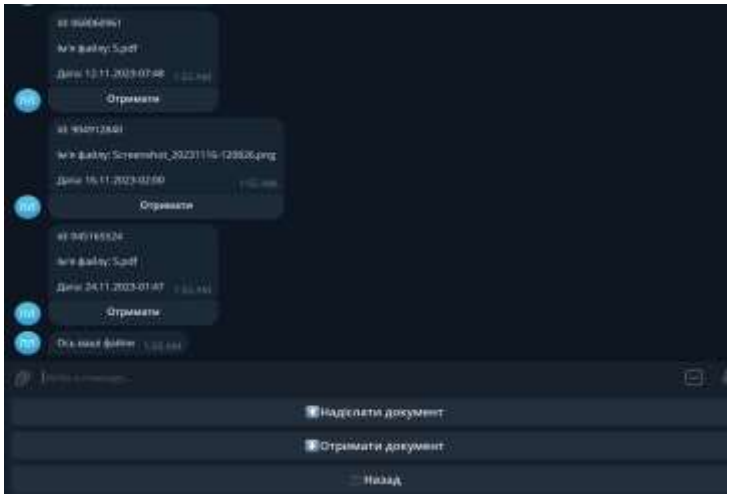


Рисунок 5.9 – Отримання файлу

Важливо, файли можуть бути з однаковим ім’ям тому звертайте увагу на дату щоб отримати саме той файл который вам потрібен.

Кнопка “Мої записи” завантажує актуальні записи. Важливо: коли запис становиться з реальною датою він буде автоматично помічати як не актуальний і користувач не буде їх бачити. (рис. 5.10)

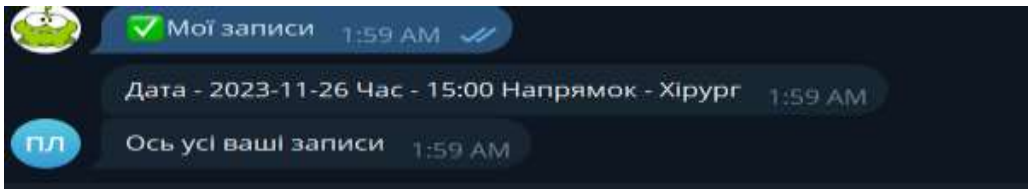


Рисунок 5.10 – Мої записи

Кнопка “Про нас” передає посилання на сайт лікарні. (рис. 5.11)

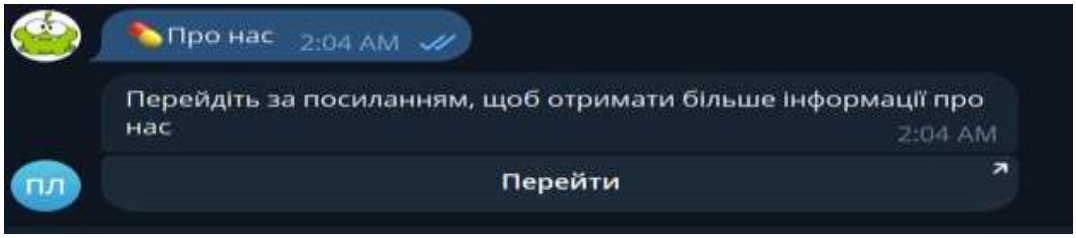


Рисунок 5.11 – Про нас

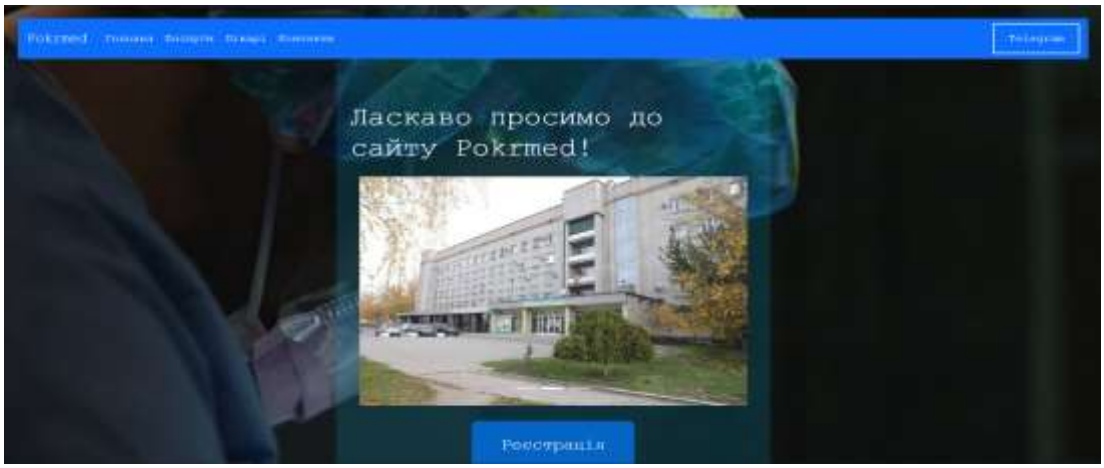


Рисунок 5.12 – Головна сторінка сайту

5.3 Переваги над існуючими аналогами.

Порівняння інтелектуального агента, написаного на Spring Boot з шифруванням документів, і системи HELSI вимагає врахування відмінностей у функціональності, безпеці, зручності використання та інших аспектах.

Функціональність:

1. інтелектуальний агент. Основний функціонал орієнтований на взаємодію з медичною клінікою через месенджер. Може надавати розклад, робити записи на прийом, відправляти шифровані документи;
2. HELSI. Медична інформаційна система, яка може надавати ширший спектр функцій, включаючи управління медичною історією, аналіз даних, взаємодію з пацієнтами та медперсоналом.

Шифрування документів:

1. інтелектуальний агент. Можливість шифрування документів може забезпечувати додатковий рівень конфіденційності;
2. HELSI. Ймовірно, також забезпечує високий рівень шифрування для медичних даних.

Безпека:

1. інтелектуальний агент. Залежить від того, наскільки добре реалізовані заходи безпеки в самому агенті та використовуваних технологіях;
2. HELSI. Як медична інформаційна система, повинна відповідати високим стандартам безпеки в медичній сфері.

Зручність використання:

1. інтелектуальний агент. Інтеграція з месенджером може бути зручною для пацієнтів, але може бути обмежена функціональністю порівняно з повноцінною системою.
2. HELSI. Ймовірно, надає більше можливостей, але може вимагати навчання персоналу та пацієнтів.

Сумісність та масштабованість:

1. інтелектуальний агент. Залежно від архітектури може бути обмежений можливостями масштабування.
2. HELSI. Ймовірно, розроблена з урахуванням масштабованості та сумісності з іншими системами в охороні здоров'я.

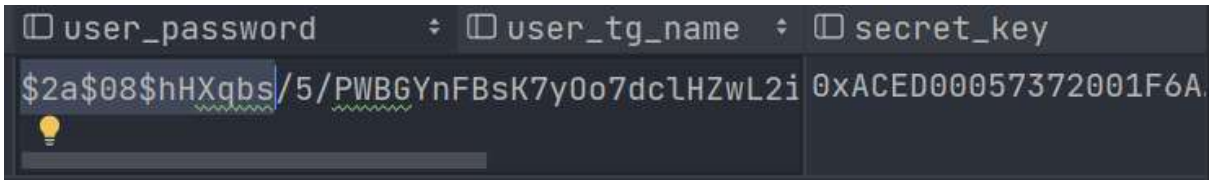
Остаточний вибір залежить від конкретних потреб медичної клініки, рівня безпеки, наданих функцій, а також зручності використання для пацієнтів та медперсоналу[41].

Шифрування даних. Телеграм використовує протоколи шифрування для захисту пересиланих повідомлень, а це може додатково захищати конфіденційні дані в обміні з інтелектуальним агентом. Також є реалізація шифрування документів користувача.(рис. 5.13)

```
public Binary encryptToBinary(InputStream input, SecretKey key, int mod) throws Exception {
    Key secretKey = new SecretKeySpec(key.getEncoded(), key.getAlgorithm());

    Cipher cipher = Cipher.getInstance(key.getAlgorithm());
    cipher.init(mod, secretKey);


    ByteArrayOutputStream output = new ByteArrayOutputStream();

    byte[] buffer = new byte[8192];
    int bytesRead;
    while ((bytesRead = input.read(buffer)) != -1) {
        byte[] encryptedBytes = cipher.update(buffer, inputOffset: 0, bytesRead);
        if (encryptedBytes != null) {
            output.write(encryptedBytes);
        }
    }

    byte[] finalEncryptedBytes = cipher.doFinal();
    if (finalEncryptedBytes != null) {
        output.write(finalEncryptedBytes);
    }


    return new Binary(BsonBinarySubType.BINARY, output.toByteArray());
}
```

Рисунок 5.13 - Метод з класу CryptoService який виконує шифрування, та дешифрування.

Аутентифікація та авторизація. Авторизація дотримується всіх правил безпеки. Інша людина зможе отримати доступ тільки, якщо вона отримує несанкціонований доступ до аккаунту телеграм користувача.



user_password	user_tg_name	secret_key
$2a$08$hHXqbs/5/PWBGYnFBsK7yOo7dclHZwL2i		0xACED00057372001F6A

Рисунок 5.14 – Зберігання паролю

Ось так зберігається пароль в БД (рис. 5.14). Також поруч бачимо secretkey для алгоритму шифрования.

5.4 Висновки за розділом 5

У розділі 5 викладено деталі та кроки взаємодії з електронною медичною системою через інтелектуальний агент, який інтегровано з месенджером Telegram. Процес включає реєстрацію, створення та підтвердження пароля, запис на прийом, обробку та збереження файлів, а також інші функції, які полегшують взаємодію пацієнта з медичною системою.

Важливим елементом є шифрування документів для забезпечення конфіденційності медичних даних. Шифрування файлів здійснюється за допомогою алгоритму RSA, що додає додатковий рівень захисту для передаваних інформаційних ресурсів.

Програмна архітектура забезпечує зручний та безпечний доступ до функцій, таких як запис на прийом, перегляд медичних записів та надсилання документів. Також враховані можливі помилки та надано відповідні сповіщення користувачам.

Порівнюючи із існуючими аналогами, система має свої переваги, такі як зручність використання через месенджер, шифрування документів для збереження конфіденційності та інші функціональні можливості. Важливо враховувати, що вибір між інтелектуальним агентом та іншими системами повинен базуватися на конкретних потребах та вимогах медичної установи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дудік А. А. Автоматизована система моніторингу життєдіяльності в медичних закладах:дип. проект/ Дудік А. А. – Київ, 2023 року – 75с.
2. Хто відповідає за достовірність даних «електронного паспорта» пацієнта? [Електроний ресурс]. – Режим доступу: https://www.vz.kiev.ua/hto-vidpovidaye-za-dostovirnist-danyh-elektronogo-pasporta-patsiyenta/
3. І.П. Заяць. Алгоритми розпізнавання рентгенологічних зображень для постановки діагнозу / X-ray image recognition algorithms for diagnosis/ І.П. Заяць – Тернопіль, 2021, 66с.
4. Загоруйко Б. Г. Інформаційна система управління роботою медичного закладу: дип. проект/ Загоруйко Б. Г. – Київ, 2023 року – 70с.
5. Гулящий І. С. Система управління медичним персоналом та звітністю: дип. проект/ Гулящий І. С. - Київ, 2023 року – 69с.
6. Як електронна медична картка пацієнта спрощує роботу лікаря. [Електроний ресурс]. – Режим доступу: https://medcentercrm.com/blog/kak-elektronnaya-mediczinskaya-karta-paczienta-uproshhaet-rabotu-vracha/
7. Стандарт інформаційної безпеки в охороні здоров'я від Українського науково-методичного центру охорони здоров'я населення. [Електроний ресурс]. – Режим доступу: https://mon.gov.ua/storage/app/media/vishcha-osvita/zatverdzeni%20standarty/12/21/229-gromadske-zdorovya-magistr.pdf
8. Бухарєв В. В. Кібербезпека в охороні здоров'я: виклики та рішення: дис. канд. юр. Наук / Владислав Вікторович Бухарєв. – Суми, 2018 року – 221с.
9. "Закон України "Про основи кібербезпеки"" - збірник законодавчих актів. [Електроний ресурс]. – Режим доступу: https://cedem.org.ua/news/zakon-pro-osnovy-kiberbezpeky-ukrayiny-nabude-chynnosti-cherez-pivroku/

10. А.Ю. Нашинець-Наумова. Інформаційна безпека в е-здоров'ї наукова публікація від Інституту інформаційної безпеки. / Нашинець-Наумова А.Ю. – Київ, 2017 року – 168с.
11. І.А. Мальчевського. Інформаційна безпека систем обробки медичної інформації/ І.А. Мальчевського – Київ, 2017 року – 45с.
12. "Стандарт інформаційної безпеки в охороні здоров'я України" - документ від Міністерства охорони здоров'я України. [Електроний ресурс]. – Режим доступу: https://zakononline.com.ua/documents/show/506038___691879
13. Бурячок В. Л. Основи інформаційної та кібернетичної безпеки. [Навчальний посібник]. / В. Л. Бурячок , Р. В. Киричок, П. М. Складанний – К. , 2018. – 320 с.
14. Управління інформаційною безпекою: конспект лекцій [Електронний ресурс] : навч. посіб. для студ. спец. 125 «Кібербезпека» / КПІ ім. Ігоря Сікорського; уклад.: С. О. Носок, О. М. Фаль, В. М. Ткач. – Електронні текстові дані (1 файл: 1114 Кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 258 с.
15. Радзішевська Є. Б., Висоцька О. В. Інформаційні технології в медицині. E-health / за ред. В. Г. Кнігавка. – Харків : ХНМУ, 2019. – 72 с.
16. Кібербезпека в інформаційному суспільстві: Інформаційно-аналітичний дайджест / відп. ред. О.Довгань; упоряд. О.Довгань, Л.Литвинова, С.Дорогих; Державна наукова установа «Інститут інформації, безпеки і права НАПрН України»; Національна бібліотека України ім. В.І.Вернадського. – К., 2023.– №9 (вересень) . – 351 с
17. "Electronic Health Records: Challenges in Design and Implementation" by Dean F. Sittig and Hardeep Singh. Published 2013 – 420p.
18. "Cryptography and Network Security: Principles and Practice" by William Stallings. Pearson Education Limited 2017 – 767p.
19. "Protecting Patient Information: A Decision-Maker's Guide to Risk, Prevention, and Damage Control" by Paul Cerrato. Published 2016 – 162p.

20. "Guide to Computer Network Security" by Joseph Migga Kizza. Published 2005 – 620p.
21. "Biometric System and Data Analysis: Design, Evaluation, and Data Mining" by Ted Dunstone and Qiuhua Liu. Published 2010 – 350p.
22. "Electronic Health Records: Understanding and Using Computerized Medical Records" by Richard Gartee. Published 2006 – 561p.
23. "Security Metrics: Replacing Fear, Uncertainty, and Doubt" by Andrew Jaquith. Published 2006 – 336p
24. "Handbook of Healthcare Delivery Systems" edited by Yuehwern Yih. Published 2010 – 812p.
25. "Computer Security: Principles and Practice" by William Stallings and Lawrie Brown. Published 2006 – 815p
26. "Biometric Recognition: Challenges and Opportunities" edited by Joseph N. Pato and Lynette I. Millett. Published 2010 – 182p
27. "The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws" by Dafydd Stuttard and Marcus Pinto. Published 2011 – 912p
28. "Healthcare Information Systems, Second Edition" by Marion J. Ball, Charlotte Weaver, and Joan M. Kiel. Published 2013 – 532p.
29. "Cybersecurity and Cyberwar: What Everyone Needs to Know" by P.W. Singer and Allan Friedman. Published 2014 – 320p.
30. "Access Control, Security, and Trust: A Logical Approach" by Shiu-Kai Chin and Susan Older. Published 2010 – 352p.
31. "Health Informatics: Practical Guide for Healthcare and Information Technology Professionals" by Robert E. Hoyt and Ann K. Yoshihashi. Published 2014 – 533p.
32. "Applied Cryptography: Protocols, Algorithms, and Source Code in C" by Bruce Schneier. Published 1995 – 792p.

33. "Healthcare Information Technology Exam Guide for CompTIA Healthcare IT Technician and HIT Pro Certifications" by Kathleen McCormick and Brian Gugerty. Published 2013 – 250p.
34. "Cybersecurity for Hospitals and Healthcare Facilities: A Guide to Detection and Prevention" by Luis Ayala. Published 2016 – 500p.
35. "Healthcare Information Management Systems: Cases, Strategies, and Solutions" by Marion J. Ball and Judy Murphy. Published 2016 – 346p
36. "Biometrics: Identity Verification in a Networked World" by Samir Nanavati, Michael Thieme, and Raj Nanavati. Published 2002 – 320p.
37. "Critical Infrastructure Protection in Homeland Security: Defending a Networked Nation" by Ted G. Lewis. Published 2006 – 486p.
38. "Introduction to Computer Security" by Michael Goodrich and Roberto Tamassia. Published 2010 – 556p.
39. "Health Informatics: An Interprofessional Approach" by Ramona Nelson and Nancy Staggers. Published 2023 – 568p.
40. "Biometrics: Personal Identification in Networked Society" by Anil K. Jain, Arun Ross, and Karthik Nandakumar. Published 2011 – 328p.
41. "Healthcare Information Security and Privacy" by Sean P. Murphy. Published 2015 – 560p.
42. Гребенюк А.М. Основи управління інформаційною безпекою: навч. посібник / А.М. Гребенюк, Л.В. Рибальченко. Дніпро: Дніпроп. держ. унт внутріш. справ, 2020. – 144 с.
43. Сьомак Роман Володимирович. Кібербезпека медичних інформаційних систем в Україні: актуальні проблеми та перспективи вирішення: дип. роб. / Сьомак Роман Володимирович. Київ, 2019р – 73с.
44. Копійка Марія Валеріївна. Кібербезпека в охороні здоров'я: аспекти захисту медичної інформації в Україні: дис. / Копійка Марія Валеріївна. Київ, 2020 – 205с.

45. Кібергігієна. Кібербезпека. Безпека держави : матеріали наукових семінарів (Київ, 27 листопада 2020 р.) / відп. ред. А. М. Десятко. – Київ : Київ. нац. торг.-екон. ун-т, 2020. – 101 с.
46. Інформаційна безпека в сучасному світі та її вплив на конституційний лад в Україні: теорія й практика: матеріали всеукраїнської конференції (м. Івано-Франківськ, 20 червня 2019 року) / упорядник В.І. Розвадовський. Івано-Франківськ : ВДНЗ «Прикарпатський національний університет імені Василя Стефаника» 2019. 116 с.
47. Доктрина збалансованого розвитку. Видання друге. — Львів: Кальварія, 2017. — 164 с.
48. ЗОЛОТАР О.О. Інформаційна безпека людини:теорія і практика : монографія. – Київ : ТОВ «Видавничий дім «АртЕк», 2018 – 446 с.
49. Електронні бібліотечні інформаційні системи наукових і навчальних закладів: монографія / [Спірін О.М., Іванова С.М., Новицький О.В. та ін.]; за наук. ред. проф. В.Ю. Бикова, О.М. Спіріна. – К. : Педагогічна думка, 2012. – 176 с.
50. "Кібербезпека в сфері охорони здоров'я: основні виклики та рішення" - стаття від Національного медичного університету імені О.О. Богомольця. [Електроний ресурс]. – Режим доступу: https://www.rnbo.gov.ua/ua/Diialnist/6365.html

Додаток А

Зауваження нормоконтролера

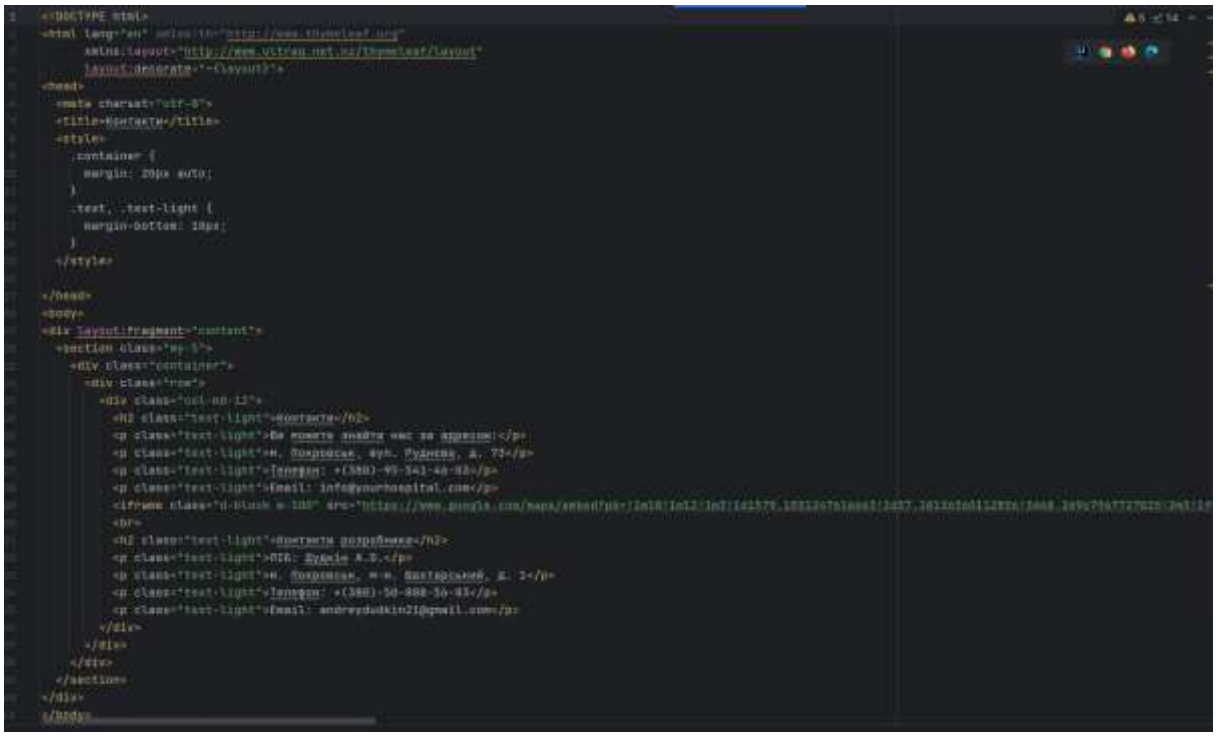
Таблиця А.1 – Зауваження нормоконтролера

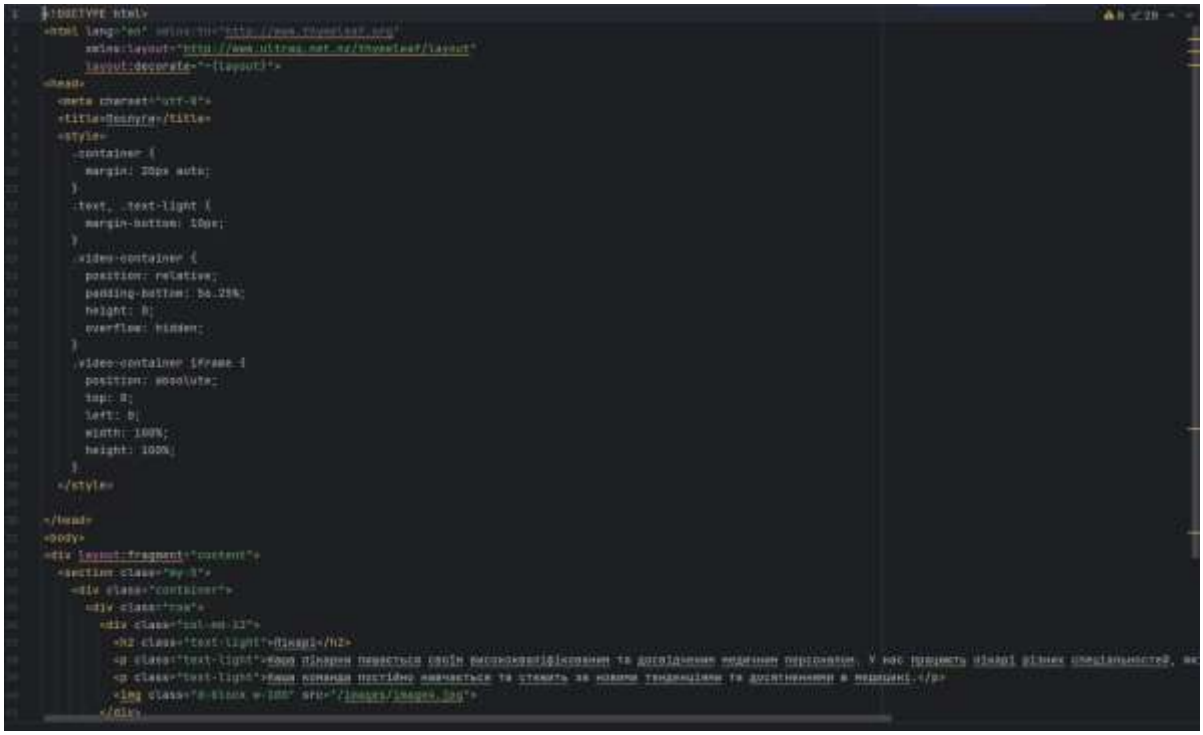
Сторінка	Позначення	Зміст зауваження
Зміст		Розділи мають службове слово «Розділ» в змісті і в тексті ПЗ, нема назв додатків, номери сторінок не відповідають номерам в тексті
ПЗ		Невірні переліки та посилання на літературу
ПЗ		Не щільне заповнення сторінок

Додаток Б

Додатки лістингу програми

Лістинг. Програмний код сайту





```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{layout}">
<head>
    <meta charset="utf-8">
    <title>Головна</title>
    <style>
        .container {
            margin: 20px auto;
        }
        .text, .text-light {
            margin-bottom: 10px;
        }
        .video-container {
            position: relative;
            padding-bottom: 56.25%;
            height: 0;
            overflow: hidden;
        }
        .video-container iframe {
            position: absolute;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
        }
        .btn-more-info {
            display: inline-block;
            padding: 20px 40px;
            font-size: 24px;
            text-align: center;
            text-decoration: none;
            background-color: rgba(0, 123, 255, 0.7); /* Полупрозрачный синий */
            color: #fff;
            border-radius: 8px;
            transition: background-color 0.3s;
        }
        .btn-more-info:hover {
            background-color: rgba(0, 123, 255, 1); /* Синий при наведении */
        }
```

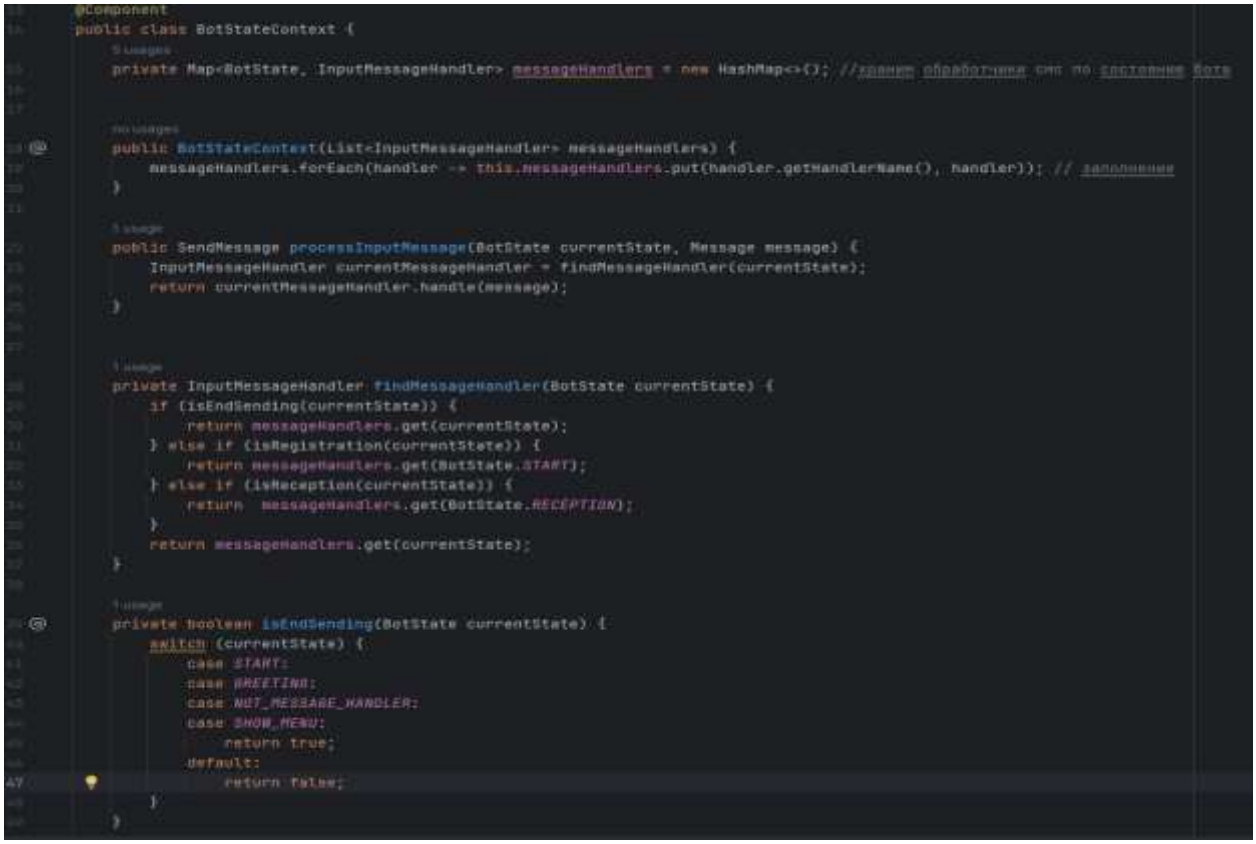
```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486
    <style>
        body {
            margin: 0;
            overflow: auto; /* Изменено с overflow: hidden; */
            font-family: 'Courier New', monospace;
            padding: 20px;
        }

        #video-background {
            position: fixed;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
            object-fit: cover;
            z-index: -1;
        }

        #content {
            max-width: 800px; /* Максимальная ширина контента */
            margin: 0 auto;
            padding: 20px;
            background-color: rgba(0, 200, 255, 0.15); /* Цвет фона контейнера */
            border-radius: 10px;
        }
        nav button {
            background-color: transparent;
            border: 2px solid white;
            color: white;
            padding: 10px 20px;
            cursor: pointer;
            transition: background-color 0.3s, color 0.3s;
            float: right; /* Добавленный стиль для выравнивания справа */
        }
```

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{layout}">
<head>
    <meta charset="utf-8">
    <title>Послуги</title>
    <style>
        .container {
            margin: 20px auto;
        }
        .text, .text-light {
            margin-bottom: 10px;
        }
        .video-container {
            position: relative;
            padding-bottom: 56.25%;
            height: 0;
            overflow: hidden;
        }
        .video-container iframe {
            position: absolute;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
        }
    </style>
</head>
<body>
<div layout:fragment="content">
    <section class="my-5">
        <div class="container">
            <div class="row">
                <div class="col-md-12">
                    <h2 class="text-light">Послуги</h2>
                    <p class="text-light">Ми пропонуємо широкий спектр медичних послуг, включаючи:</p>
                    <ul class="text-light">
                        <li>Консультації лікарів різних спеціальностей</li>
                        <li>Діагностичні дослідження</li>
                        <li>Лабораторні аналізи</li>
```

Лістинг. BotStateContext

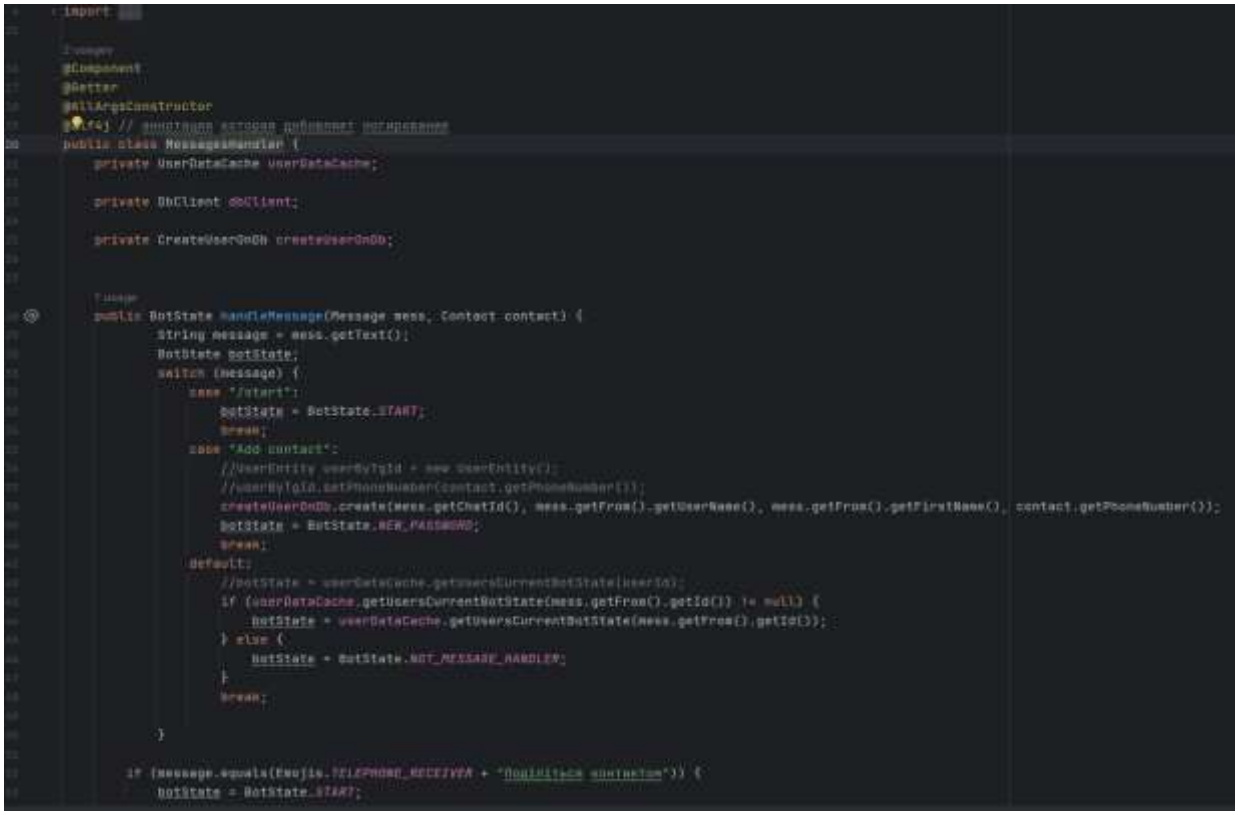
```
@Component
public class BotStateContext {
    private Map<BotState, InputMessageHandler> messageHandlers = new HashMap<>(); //хранит обработчики сообщ по состоянию бота

    public BotStateContext(List<InputMessageHandler> messageHandlers) {
        messageHandlers.forEach(handler -> this.messageHandlers.put(handler.getHandlerName(), handler)); // заполнение
    }

    public SendMessage processInputMessage(BotState currentState, Message message) {
        InputMessageHandler currentMessageHandler = findMessageHandler(currentState);
        return currentMessageHandler.handle(message);
    }

    private InputMessageHandler findMessageHandler(BotState currentState) {
        if (isEndSending(currentState)) {
            return messageHandlers.get(currentState);
        } else if (isRegistration(currentState)) {
            return messageHandlers.get(BotState.START);
        } else if (isReception(currentState)) {
            return  messageHandlers.get(BotState.RECEPTION);
        }
        return messageHandlers.get(currentState);
    }

    private boolean isEndSending(BotState currentState) {
        switch (currentState) {
            case START:
            case GREETING:
            case NOT_MESSAGE_HANDLER:
            case SHOW_MENU:
                return true;
            default:
                return false;
        }
    }
```

Лістинг. Додавання файлів та відправка



```
@Component
public class BotStateContext {
    private Map<BotState, InputMessageHandler> messageHandlers = new HashMap<>(); //хранит обработчик смс по состоянию бота


    public BotStateContext(List<InputMessageHandler> messageHandlers) {
        messageHandlers.forEach(handler -> this.messageHandlers.put(handler.getHandlerName(), handler)); // заполнение
    }

    public SendMessage processInputMessage(BotState currentState, Message message) {
        InputMessageHandler currentMessageHandler = findMessageHandler(currentState);
        return currentMessageHandler.handle(message);
    }


    private InputMessageHandler findMessageHandler(BotState currentState) {
        if (isEndSending(currentState)) {
            return messageHandlers.get(currentState);
        } else if (isRegistration(currentState)) {
            return messageHandlers.get(BotState.START);
        } else if (isReception(currentState)) {
            return  messageHandlers.get(BotState.RECEPTION);
        }
        return messageHandlers.get(currentState);
    }

    private boolean isEndSending(BotState currentState) {
        switch (currentState) {
            case START:
            case GREETING:
            case NOT_MESSAGE_HANDLER:
            case SHOW_MENU:
                return true;
            default:
                return false;
        }
    }
```

Лістинг. Робота кнопок



```
import ...

@Component
@Getter
@AllArgsConstructor
@Slf4j // аннотация которая добавляет логирование
public class MessageHandler {
    private UserDataCache userDataCache;

    private DbClient dbClient;

    private CreateUserOnDb createUserOnDb;


    public BotState handleMessage(Message mess, Contact contact) {
            String message = mess.getText();
            BotState botState;
            switch (message) {
                case "/start":
                    botState = BotState.START;
                    break;
                case "Add contact":
                    //UserEntity userById = new UserEntity();
                    //userById.setPhoneNumber(contact.getPhoneNumber());
                    createUserOnDb.create(mess.getChatId(), mess.getFrom().getUserName(), mess.getFrom().getFirstName(), contact.getPhoneNumber());
                    botState = BotState.NEW_PASSWORD;
                    break;
                default:
                    //botState = userDataCache.getUsersCurrentBotState(userId);
                    if (userDataCache.getUsersCurrentBotState(mess.getFrom().getId()) != null) {
                        botState = userDataCache.getUsersCurrentBotState(mess.getFrom().getId());
                    } else {
                        botState = BotState.NOT_MESSAGE_HANDLER;
                    }
                    break;

            }

        if (message.equals(Emojis.TELEPHONE_RECEIVER + "Поделиться контактом")) {
            botState = BotState.START;
```

Лістинг. Конфігурація Інтелектуального агента

```java
package com.pokrmed.config;

import ...

no usages
@Setter
@Getter
@Configuration
@ComponentScan({"com.pokrmed.botapi.handlers", "com.pokrmed.service"})
@ConfigurationProperties("application.properties")
public class BotConfig {
    private String botUsername;
    private String botToken;
    private String webHook;

    no usages
    @Bean
    public PokrmedBot bot(TelegramFacade telegramFacade) {
     PokrmedBot pokrmedBot = new PokrmedBot(telegramFacade);
     pokrmedBot.setUserName(botUsername);
     pokrmedBot.setBotToken(botToken);
     pokrmedBot.setWebhook(webHook);
     return pokrmedBot;
    }

    no usages
    @Bean
    public MessageSource messageSource() {
        ReloadableResourceBundleMessageSource messageSource
                = new ReloadableResourceBundleMessageSource();

        messageSource.setBasenames("classpath:messages");
        messageSource.setDefaultEncoding("UTF-8");
        return messageSource;
    }
}
```

```java
package com.pokrmed.config;

import ...

no usages
@Configuration
@PropertySource({"classpath:config/multiple-db-boot.properties"})
@EnableMongoRepositories(
        basePackageClasses = {MetaDataRepository.class, FilesDataRepository.class} )
public class PersistenceMultipleDbConfiguration {

    no usages
    @Bean(name = "primaryProperties")
    @ConfigurationProperties(prefix = "mongodb.primary")
    @Primary
    public MongoProperties primaryProperties() { return new MongoProperties(); }

    no usages
    @Bean(name = "primaryMongoClient")
    public MongoClient mongoClient(@Qualifier("primaryProperties") MongoProperties mongoProperties) {

        ServerApi serverApi = ServerApi.builder()
                .version(ServerApiVersion.V1)
                .build();

        MongoCredential credential = MongoCredential
                .createCredential(mongoProperties.getUsername(), mongoProperties.getAuthenticationDatabase(), mongoProperties.getPassword());

        return MongoClients.create(MongoClientSettings.builder()
                .applyConnectionString(new ConnectionString(mongoProperties.getUri()))
                .serverApi(serverApi).credential(credential)
                .build());
    }

}
```

Лістинг. Кнопки «Про нас»

```java
package com.pokrmed.botapi.handlers;

import ...

no usages
@Component
@AllArgsConstructor
public class AboutUsHandler implements InputMessageHandler {

    private ReplyMessagesService replyMessagesService;

    private ButtonsMenuService buttonsMenuService;

    @Override
    public SendMessage handle(Message message) {
        SendMessage reply = new SendMessage();
        reply.setText(replyMessagesService.getReplyText("about.us"));
        reply.setChatId(message.getChatId());
        reply.setReplyMarkup(buttonsMenuService.getGo(replyMessagesService.getReplyText("about.us.link")));
        return reply;
    }

    @Override
    public BotState getHandlerName() { return BotState.ABOUT_US; }
}
```

Лістинг. Кнопки «Запис на прийом»

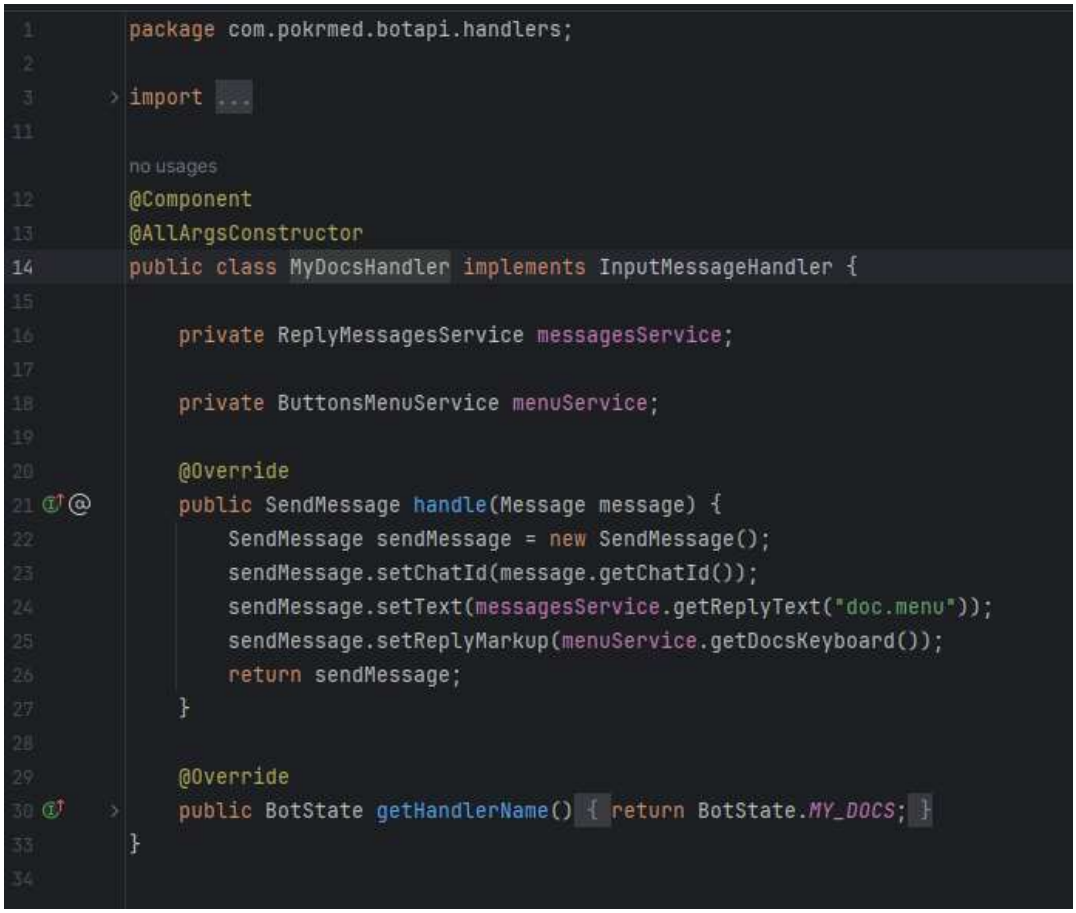
```java
package com.pokrmed.botapi.handlers;

import ...

no usages
@Component
@AllArgsConstructor
public class GetDocHandler implements InputMessageHandler {

    private ReplyMessagesService messagesService;

    private MongoFilesService mongoFilesService;

    private ApiService apiService;

    private CreateFileAndSend createFileAndSend;


    @Override
    public SendMessage handle(Message message) {
        List<MetaDataEntity> byUserId = mongoFilesService.findByUserId(message.getChatId());
        if (!byUserId.isEmpty()) {
            createFileAndSend.getFiles(String.valueOf(message.getChatId()));
            return messagesService.getReplyMessage(message.getChatId(), replyMessage: "get.file");
        }
        return messagesService.getReplyMessage(message.getChatId(), replyMessage: "not.file");
    }

    @Override
    public BotState getHandlerName() { return BotState.GET_DOC; }
}
```

```java
package com.pokrmed.botapi.handlers;

import ...

no usages
@Component
@AllArgsConstructor
public class GetReceptionHAndler implements InputMessageHandler {

    private ReplyMessagesService messagesService;

    private DbClient dbClient;

    private ApiService apiService;

    private ButtonsMenuService buttonsMenuService;
    @Override
    public SendMessage handle(Message message) {
        if (dbClient.findByUserId(message.getChatId()).getAppointmentsEntities().size() == 0) {
            return messagesService.getReplyMessage(message.getChatId(), replyMessage: "reception.not.found");
        }
        getReceptions(message.getChatId());
        return messagesService.getReplyMessage(message.getChatId(), replyMessage: "all.receptions");
    }

    @Override
    public BotState getHandlerName() { return BotState.GET_USER_RECEPRIONS; }

    1 usage
    public void getReceptions(long userID) {
        UserEntity byUserId = dbClient.findByUserId(userID);
        List<AppointmentsEntity> appointmentsEntities = byUserId.getAppointmentsEntities();
        List<String> receptions = receptions(appointmentsEntities);
        for (int i = 0; i < appointmentsEntities.size(); i++) {
            apiService.sendMessage(userID, receptions.get(i));
        }

    }

    1 usage
    private List<String> receptions(List<AppointmentsEntity> reception) {
        List<String> messages = new ArrayList<>();
```

Лістинг. Кнопки «Головне меню»

```java
package com.pokrmed.botapi.handlers;

import ...

no usages
@Component
@AllArgsConstructor
public class MainMenuHandler implements InputMessageHandler {

    private ReplyMessagesService messagesService;

    private ButtonsMenuService menuService;
    @Override
    public SendMessage handle(Message message) {
        SendMessage sendMessage = new SendMessage();
        sendMessage.setChatId(message.getChatId());
        sendMessage.setText(messagesService.getReplyText("main.menu"));
        sendMessage.setReplyMarkup(menuService.getMainKeyboard());
        return sendMessage;
    }

    @Override
    public BotState getHandlerName() { return BotState.SHOW_MENU; }
}
```

Лістинг. Кнопки «Мої документи»



```java
package com.pokrmed.botapi.handlers;

import ...

no usages
@Component
@AllArgsConstructor
public class MyDocsHandler implements InputMessageHandler {

    private ReplyMessagesService messagesService;

    private ButtonsMenuService menuService;

    @Override
    public SendMessage handle(Message message) {
        SendMessage sendMessage = new SendMessage();
        sendMessage.setChatId(message.getChatId());
        sendMessage.setText(messagesService.getReplyText("doc.menu"));
        sendMessage.setReplyMarkup(menuService.getDocsKeyboard());
        return sendMessage;
    }

    @Override
    public BotState getHandlerName() { return BotState.MY_DOCS; }
}
```

Додаток В

Матеріали презентації

Тема: Дослідження особливостей реалізації принципів захисту інформації в електронних медичних системах

Автор: ст. гр. КНм-22 Дудкін А.О.
Керівник: Маслова Н.О., зав.кафедри ПМІ, к.т.н., доц.

Об’єкт та предмет дослідження

- **Актуальність:** є надзвичайно актуальною в контексті сучасних тенденцій розвитку медичних інформаційних технологій. З впровадженням електронних медичних систем, збільшенням обсягів зберігання та обміну медичною інформацією, проблема захисту конфіденційності пацієнтських даних стає на перший план.
- **Об'єктом дослідження:** є технології побудови електронних медичних систем (ЕМС), вимоги та підходи до принципів захисту інформації.
- **Предметом дослідження:** є конкретні аспекти та компоненти, що стосуються реалізації принципів захисту інформації в електронних медичних системах (ЕМС).

Мета та задачі роботи

- **Мета роботи:** полягає в дослідженні існуючих рішень забезпечення безпеки та конфіденційності медичної інформації в електронних медичних системах й наданні пропозицій щодо підвищення якості медичної допомоги та довіри пацієнтів.
- Задачі роботи:
  - аналіз поточного стану системи захисту даних;
  - оцінка рівня захищеності медичної інформації;
  - вивчення стандартів і регуляторних вимог;
  - розробка та впровадження стратегій і заходів безпеки;
  - моніторинг та аудит безпеки;
  - підготовка рекомендацій та звіту;
  - інформаційна освіта і навчання персоналу.

ОГЛЯД ЕЛЕКТРОННИХ МЕДИЧНИХ СИСТЕМ

Електронні медичні системи (ЕМС) - це інформаційні технології для збору, зберігання, обробки та аналізу медичних даних. Вони включають електронні медичні картки (ЕМК), лабораторні інформаційні системи, системи керування прийомами, управління рецептами, телемедицину та інші. ЕМС полегшують роботу медперсоналу, забезпечують доступність та збереження даних в електронному форматі. Забезпечення конфіденційності та захист персональних даних є ключовим аспектом їхнього впровадження.

РЕАЛІЗАЦІЯ ПРИНЦИПІВ ЗАХИСТУ ІНФОРМАЦІЇ В ЕЛЕКТРОННИХ МЕДИЧНИХ СИСТЕМАХ

Принципи конфіденційності, цілісності та доступності є критичними для безпеки медичної інформації в електронних медичних системах (ЕМС).

- **Конфіденційність:** Гарантує обмежений доступ до медичної інформації через механізми автентифікації та авторизації.
- **Цілісність:** Забезпечує непорушеність даних, використовуючи хеш-функції та цифрові підписи.
- **Доступність:** Гарантує стабільний доступ до інформації через резервне копіювання та високодоступні архітектури.

Переваги Інтеграції Технологій:

- **Шифрування:** Захист медичних даних за допомогою сучасних алгоритмів шифрування.
- **Двофакторна аутентифікація:** Забезпечення надійного захисту через системи двофакторної аутентифікації.
- **Моніторинг та виявлення загроз:** Оперативна реакція на потенційні атаки через системи виявлення загроз.
- **Безпечне зберігання інформації:** Забезпечення захисту даних за допомогою шифрування та регулярного аудиту.
  - Цей підхід підкреслює важливість використання сучасних технологій для ефективного захисту медичної інформації в ЕМС.

РЕАЛІЗАЦІЯ ПРИНЦИПІВ ЗАХИСТУ ІНФОРМАЦІЇ В ЕЛЕКТРОННИХ МЕДИЧНИХ СИСТЕМАХ

- Важливість захисту даних в реальному часі для електронних медичних систем (ЕМС) визначається швидким та безпечним доступом до інформації. Системи аналізу в реальному часі та виявлення загроз, використовуючи штучний інтелект і машинне навчання, грають ключову роль у запобіганні проблем та забезпеченні безпеки пацієнтів. Важливість оперативності визначається впливом затримок на медичні процеси та якість медичних послуг. Розділ також розглядає регуляторні вимоги, такі як HIPAA і GDPR, і заходи з аудиту та моніторингу для виявлення та усунення можливих загроз та відхилень.

МЕТОДИ ЗАСТОВНІ В ЕЛЕКТРОННИХ МЕДИЧНИХ СИСТЕМАХ

Введення даних в електронні медичні системи (ЕМС):

1.Ручне введення:

1. Лікарські записи, лабораторні результати та інші документи вручну вводяться медперсоналом.
2. Заходи безпеки включають перевірки на достовірність та подвійне перевірення важливих даних.

2.Автоматичний збір даних:

1. Медичні прилади та системи обробки зображень автоматично збирають та вводять дані в ЕМС.

3.Захист даних:

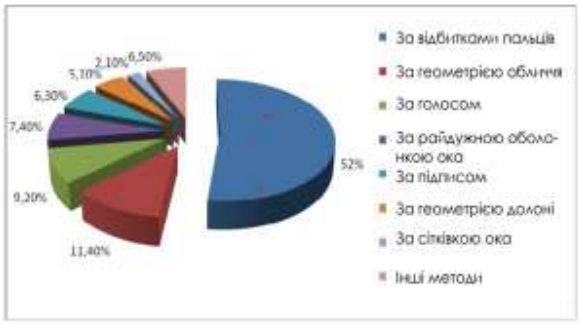
1. Шифрування, перевірка цифрових підписів та інші заходи безпеки забезпечують цілісність та впевненість у правильності даних.

4.Інтеграція з іншими системами:

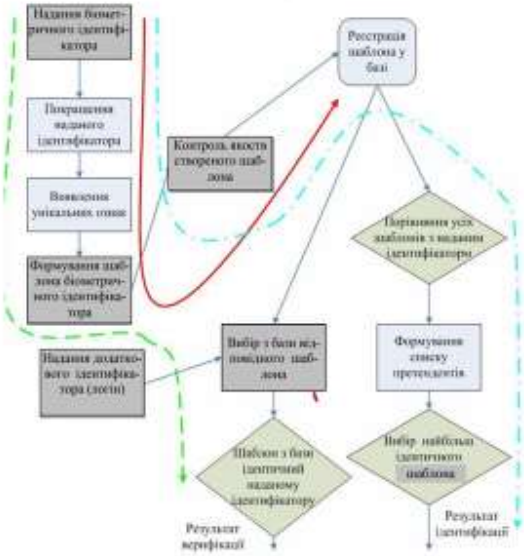
1. Інтеграція з лабораторіями, аптеками та телемедициною для автоматичного введення результатів аналізів та інших даних.

5.Відстеження ідентичності пацієнтів:

1. Використання систем аутентифікації, таких як біометрія чи картки доступу, для перевірки ідентичності перед введенням даних.



Загальна схема функціонування системи біометричної ідентифікації



ТЕХНОЛОГІЇ РОЗРОБКИ ЕМС

Java в медичних системах:

- **Надійність:**
  - Обробка помилок та управління пам'яттю для стабільності.
- **Переносимість:**
  - Можливість роботи на різних платформах.
- **Багатофункціональність:**
  - Розгалужені інструменти для різноманітних функцій.
- **Стабільність та продуктивність:**
  - Висока продуктивність та ресурсозбереження.
- **Інтерфейс користувача:**
  - JavaFX та Swing для легких інтерфейсів.
- **Широке сприйняття:**
  - Популярність серед розробників.
- **Spring Boot для бекенду:**
- **RESTful сервіси:**
  - Проста обробка HTTP-запитів.
- **Безпека:**
  - Spring Security для аутентифікації та авторизації.
- **Інтеграція з базами даних:**
  - Проста взаємодія з різними базами даних.
- **Масштабованість:**
  - Створення та обслуговування великих систем.
- **Спрощене конфігурування:**
  - Конфігурація за замовчуванням для легкого старту.
- **Розширюваність:**

  Модульна архітектура для розширення функціоналу.

  Java та Spring Boot гарно поєднуються в медичних системах, забезпечуючи ефективність та безпеку.

Розробка "Інтелектуального Агента"

UML діаграма configs spring boot

Клас BotConfig

Класи для обробки повідомлень які ми отримали

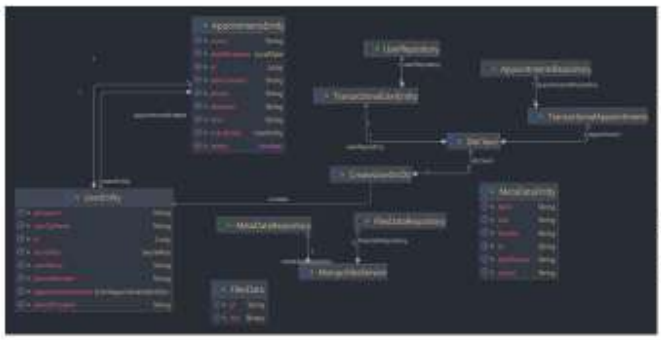
контролери

Розробка "Інтелектуального Агента"

Приклад програмного коду

Приклад програмного коду

Uml Persistence - взаемодія з базами даних.



Комбінація Postgres і MongoDB дозволяє оптимально використовувати переваги обох систем зберігання даних та забезпечити повноту та різноманітність медичної інформації, що може бути важливою для різних клінічних сценаріїв та досліджень.

Properties - для підключення к БД.

Конфігурація MongoDB

UML MongoFilesService - методи для роботи з файлами.

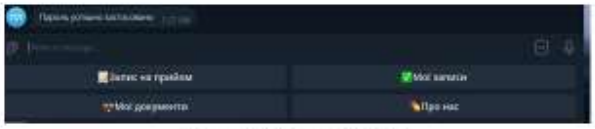
Properties - для підключеня до БД mongo.

Контролери для обробки шаблонів thymeleaf.

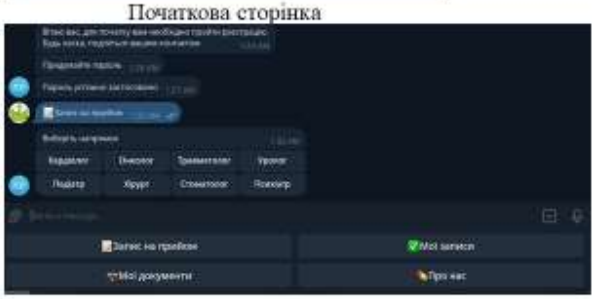
MongoDB з використанням конфігурації та об'єкта MongoProperties. Цей сервіс взаємодіє з MongoDB для зберігання та отримання метаданих та файлових даних. Методи цього сервісу дозволяють зберігати, отримувати та видаляти дані з обох репозиторіїв (MetaData та FilesData).

КОНТРОЛЬ ПРАЦЕЗДАТНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Початкова сторінка



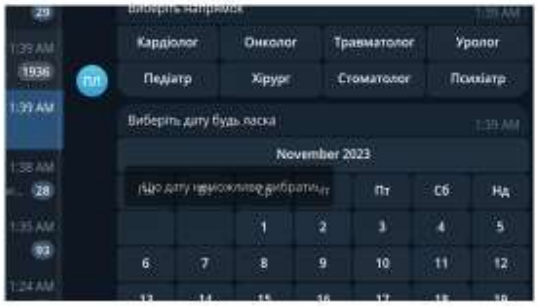
Затвердження паролю



Запис на прийом

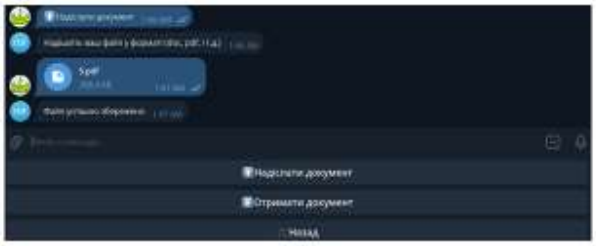
Обирання дати та часу

КОНТРОЛЬ ПРАЦЕЗДАТНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



Сповіщення про помилку

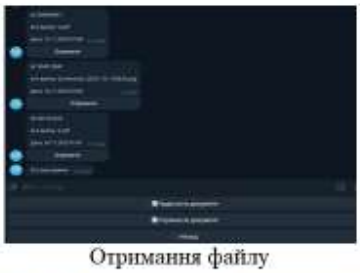
Мої документи



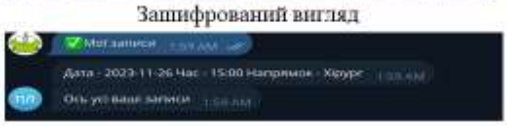
Надсилання файлу

КОНТРОЛЬ ПРАЦЕЗДАТНОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Зашифрований вигляд



Отримання файлу



Мої записи

Про нас

Переваги над існуючими аналогами.

- **Функціональність:**
  - *Інтелектуальний агент:* Зорієнтований на месенджер, записи, та шифровані документи.
  - *HELSI:* Управління медичною історією, аналіз даних, взаємодія з пацієнтами.
- **Шифрування документів:**
  - *Інтелектуальний агент:* Додатковий рівень конфіденційності.
  - *HELSI:* Ймовірно, високий рівень шифрування.
- **Безпека:**
  - *Інтелектуальний агент:* Залежить від реалізації безпеки.
  - *HELSI:* Високі стандарти безпеки в медичній сфері.
- **Зручність використання:**
  - *Інтелектуальний агент:* Зручний для пацієнтів, обмежений функціональністю.
  - *HELSI:* Більше можливостей, можливе навчання персоналу.
- **Сумісність та масштабованість:**
  - *Інтелектуальний агент:* Залежить від архітектури.
  - *HELSI:* Ймовірно, розроблена з урахуванням масштабованості та сумісності.
- **Остаточний вибір:**
- Залежить від потреб клініки, рівня безпеки, функцій та зручності використання.
- **Шифрування даних в Телеграм:**
- Використовує протоколи шифрування для захисту пересиланих повідомлень.
- Додаткова захист конфіденційних даних в обміні з інтелектуальним агентом.

Метод з класу CryptoService який виконує шифрування, та дешифрування.

Зберігання паролю

Отримані результати

- Інтеграція інтелектуального агента (ІА) з месенджером Telegram в електронну медичну систему. Основні етапи включають реєстрацію, створення та підтвердження пароля, запис на прийом, обробку та збереження файлів, а також інші функції для полегшення взаємодії пацієнтів із системою.
- Важливим елементом є використання шифрування документів за допомогою алгоритму RSA для забезпечення конфіденційності медичних даних. Програмна архітектура забезпечує зручний та безпечний доступ до функцій, а система надає сповіщення користувачам у разі можливих помилок.
- Порівняно з існуючими аналогами, система має свої переваги, такі як зручність використання через месенджер, шифрування документів для збереження конфіденційності та інші функціональні можливості. При виборі між інтелектуальним агентом та іншими системами важливо враховувати конкретні потреби та вимоги медичної установи.

ДЯКУЮ ЗА УВАГУ