Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Шипаєвої Діани Володимирівни*
(ПІБ)

академічної групи *121-19-2*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка веб-інтерфейсу медичної картки пацієнта*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф. Лактіонов І. С.*	*95*	*відмінно*	
розділів:				
спеціальний	*проф. Лактіонов І. С.*			
економічний	*доц. Касьяненко Л.В.*	*93*	*відмінно*	
Рецензент				
Нормоконтролер	*ст.викл. Мартиненко А.А.*			

Дніпро
2023

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » _____ 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-19-2* *Шипаєвої Діани Володимирівни*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка веб-інтерфейсу медичної картки пацієнта*

затверджена наказом ректора НТУ «ДП» від *16.05.2023* № *350-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2023 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2023 р.***

Завдання видав _____ *проф. Лактіонов І.С.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Шипаєва Д.В.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2023 р.*

Термін подання кваліфікаційної роботи до ЕК: *12.06.2023*

РЕФЕРАТ

Пояснювальна записка: 82 с., 21 рис., 3 дод., 21 джерел.

Об'єкт розробки: Розробка веб-інтерфейсу медичної картки пацієнта.

Мета кваліфікаційної роботи: створення потужного та інтуїтивно зрозумілого веб-інтерфейсу, який дозволить зручно та ефективно працювати з персональними даними пацієнтів і забезпечить їх безпеку в даному середовищі. Веб-інтерфейс дозволяє здійснювати опрацювання даних, зокрема редагування, видалення та створення записів, забезпечуючи медичному персоналу зручний спосіб управління інформацією.

У вступі висвітлено аналіз та сучасний стан проблеми, мета кваліфікаційної роботи та галузь її застосування, наведено зразки реалізації подібних систем.

У першому розділі проаналізовано предметну галузь, визначено актуальність завдання та призначення розробки. Сформульовано постановку завдання та вказано вимоги до програмної реалізації, технологій та програмних засобів.

В другому розділі були проаналізовані існуючі рішення, підібрані платформи розробки, виконано проектування та розробку програми. Описано роботу програми, включаючи її алгоритми і структуру функціонування, а також проаналізовано процес виклику та завантаження програми. Були визначені вхідні та вихідні дані, а також наданий огляд параметрів технічних засобів.

В економічному розділі визначено трудомісткість розроблення програмного забезпечення, проведений підрахунок вартості робіт по створенню програми та розраховано час на його створення.

Практичним значенням є створення веб-інтерфейсу для роботи з медичними даними пацієнтів та забезпечення їх безпеки. Поліпшити ефективність та точність обробки медичних даних, що посприяє збереженню часу та зменшенню помилок при обробці даних. А також, забезпечити безпеку даних, шляхом впровадження різноманітних заходів безпеки, таких як аутентифікація та авторизація, що дозволить тільки уповноваженим користувачам з достатніми привілеями мати доступ до інформації.

Актуальність даного веб-інтерфейсу полягає в розвитку медичної сфери та кібербезпеки цієї сфери. Поліпшення процесу управління медичною інформацією, що посприяє покращенню аналізу та інтерпретації даних, і врешті-решт, вдосконалення загальної якості медичних послуг.

Ключові слова: ВЕБ-ІНТЕРФЕЙС, МЕДИЧНА ІНФОРМАЦІЙНА СИСТЕМА, КІБЕРБЕЗПЕКА, АЛГОРИТМ, ПРОЕКТУВАННЯ, РОБОТА З ДАНИМИ.

ABSTRACT

Explanatory note: 82 pages., 21 fig., 3 appendix, 21 sources.

Object of development: Web-interface of the patient's medical card.

The goal of the qualification work: creation of a powerful and intuitive web interface that will allow convenient and efficient work with personal data of patients and ensure their safety in this environment. The web interface allows for data processing, including editing, deleting, and creating records, providing medical personnel with a convenient way to manage information.

The introduction highlights the analysis and current state of the problem, the purpose of the qualification work and the field of its application, examples of the implementation of similar systems are given.

In the first section, the subject area is analyzed, the relevance of the task and the purpose of development are determined. The statement of the task is formulated and the requirements for software implementation, technologies and software tools are specified.

In the second section, existing solutions were analyzed, development platforms were selected, design and development of the program was performed. The operation of the program is described, including its algorithms and structure of functioning, as well as the process of calling and downloading the program is analyzed. The input and output data were defined, and an overview of the parameters of the technical means was provided.

In the economic section, the labor intensity of software development is determined, the cost of robots for creating the program is calculated, and the time for its creation is calculated.

The practical significance is the creation of a web interface for working with patients' medical data and ensuring their security. Improve the efficiency and accuracy of medical data processing, which will help save time and reduce errors in data processing. Also, ensure data security by implementing various security measures, such as authentication and authorization, which will allow only authorized users with sufficient privileges to access information.

The relevance of this web interface lies in the development of the medical field and cyber security of this field. Improving the process of managing health information, which will contribute to the improvement of analysis and interpretation of data, and ultimately, the improvement of the overall quality of health services.

Keywords: WEB-INTERFACE, MEDICAL INFORMATION SYSTEM, CYBER SECURITY, ALGORITHM, DESIGN, WORKING WITH DATA.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

МІС – медична інформаційна система;

БД – база даних;

ЦБД – центральна база даних;

КСЗІ – комплексна система захисту інформації;

TCCC – tactical combat casualty care;

ВСТУП

Завдання цієї кваліфікаційної роботи та її об'єкт дослідження відповідають напрямку «Інформаційні технології» та включають до загальної тематики кваліфікаційних робіт, переліку виробничих функцій, типових задача діяльності, а також умінь та компетенцій, якими мають володіти бакалаври спеціальності 121 «Інженерія програмного забезпечення».

Тематика даної кваліфікаційної роботи є високоактуальною, оскільки спрямована на розробку веб-інтерфейсу медичної картки пацієнта. Даний інтерфейс надає доступ до важливих даних та забезпечує ефективне управління медичними записами.

Головною метою даної кваліфікаційної роботи є покращення ефективності захисту програмних засобів, що використовуються в медичних установах для обліку пацієнтів. Розробивши веб-інтерфейс, який забезпечує зберігання та надання віддаленого доступу до даних відповідно до наданих прав користувача. Акцент роботи розташований на підвищення рівня кіберзахисту для безпеки та конфіденційності медичної інформації.

МІС для медичного закладу - це інструмент, призначений не тільки для зберігання та обробки даних, але й для вирішення локальних питань з управління медичним закладом.

На сьогоднішній час, в Україні, існує велика кількість медичних інформаційних систем: ТОВ «ХЕЛСІ ЮА», МІС «ЕМСіМЕД», МІС «Доктор Елекс» та інші. Електронна система працює за модульним принципом, що дозволяє окремо впроваджувати потрібні компоненти залежачи від їх потрібності та фінансування. Кожен функціонал має пройти тестування після чого МІС буде підключено до ЦБД eHealth.

Кількість МІС в різних медичних сферах зростає, але більшість все ще на етапі доопрацювання, через що не можуть забезпечити повноцінний функціонал. Першорядним завданням є створити таке електронну систему охорони здоров’я, що дасть змогу для медичного персоналу швидко та лаконічно розпоряджатись часом та даними.

Суттєвим є запровадження інформаційної безпеки даних, які використовуються в МІС. Особливо, в умовах війни з росією, кібербезпека в Україні є одним з визначних аспектів в сфері національної безпеки.

Як результат виконання даної кваліфікаційної роботи є веб-інтерфейс медичної картки пацієнта, до якого отримують доступ тільки зареєстровані користувачі. Забезпечення безпеки та доступу до цієї інформації буде здійснювати за допомогою платформи Java Spring Security. Таким чином, гарантується конфіденційність та надійність медичних даних, що зберігаються в системі.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості з предметної галузі

Система eHealth - система, яка забезпечує роботу з медичними даними в електронному вигляді. Це центральна база даних державної системи охорони здоров'я, що містить дані в центральному сховищу. Доступ до даних є у Міністерства охорони здоров'я, НСЗУ і у медичних установ, які підключенні до системи. Інформація до бази даних надходить з лікарняних установ, що передають дані через медичні інформаційні системи (МІС).

МІС - призначений для передавання даних до ЦБД (eHealth), але ще й для вирішення питань управління медичним закладом. Сюди входить онлайн-запис до фахівця, отримання електронних направлень, телемедицина та інше. Під час роботи з великими об'ємами інформації медустанови мають змогу отримувати якісну аналітику для відстеження змін або запобігання ускладнень у лікуванні. Впровадження електронної картки пацієнта дає змогу зменшити час, що витрачається на паперову документацію до 50% та істотно знижує витрати на пошук даних минулих обстежень пацієнта.

Важливим постає питання кібербезпеки інформації, особливої ваги воно набуває під час військового стану. Кібератаки не завжди мають на меті викрадення інформації, може бути здійснена DDoS-атака, з наміром перевантажити сервера та зробити їх недоступними для користувачів, пошкодження інформації чи її фальсифікація. За даними Міністерства та Комітету цифрової трансформації України вночі тільки вночі з 13 на 14 січня було атаковано понад 70 держресурсів, 10 з яких зазнали несанкціонованого втручання.

Медичні установи є одні з важливих інституцій в Україні, передусім у військовий час, коли в лікарнях знаходяться військові, конфіденційність яких є одним з пріоритетів. Ім'я пацієнта, домашня адреса, телефонний номер,

біометрична ідентифікація, такі як відбиток пальця, фотографія обличчя, медичний план тощо. Основною ціллю є забезпечення кібербезпеки, для цього використовують хмарні сховища, кодування даних (шифрування), план по відновленню системи після інцидента, надання інформації за роллю користувача та інші.

Розглянемо та проаналізуємо кілька відомих на даний момент систем МІС: ТОВ «ХЕЛСІ ЮА», МІС «ЕМСіМЕД» та МІС «Доктор Елекс».

ТОВ «ХЕЛСІ ЮА» - медична інформаційна система, що використовується як медичними закладами, так і пацієнтам. Один з головних аспектів Хелсі є те що система розгорнута в хмарному середовищі. Зберігання даних у хмарі є одним з ефективних способів кіберзахисту інформації. По-перше, працівники медичного закладу не мають прямого доступу до серверів. По-друге, інформація що знаходиться у хмарі зашифрована, що значно ускладнює доступ кіберзлочинцям. Хелсі є веб-інтерфейсом (рис.1.1), що є спрощенням для лікарняних установ, позбавленням потреби встановлювати програмне забезпечення на комп'ютери. Має різноманітні під'єднанні модулі так як: звіти та статистика, доступ до ЦБД eHealth, електронна медична картка пацієнта та інше. Система має групу користувачів у кожного з яких є свої права та доступи.

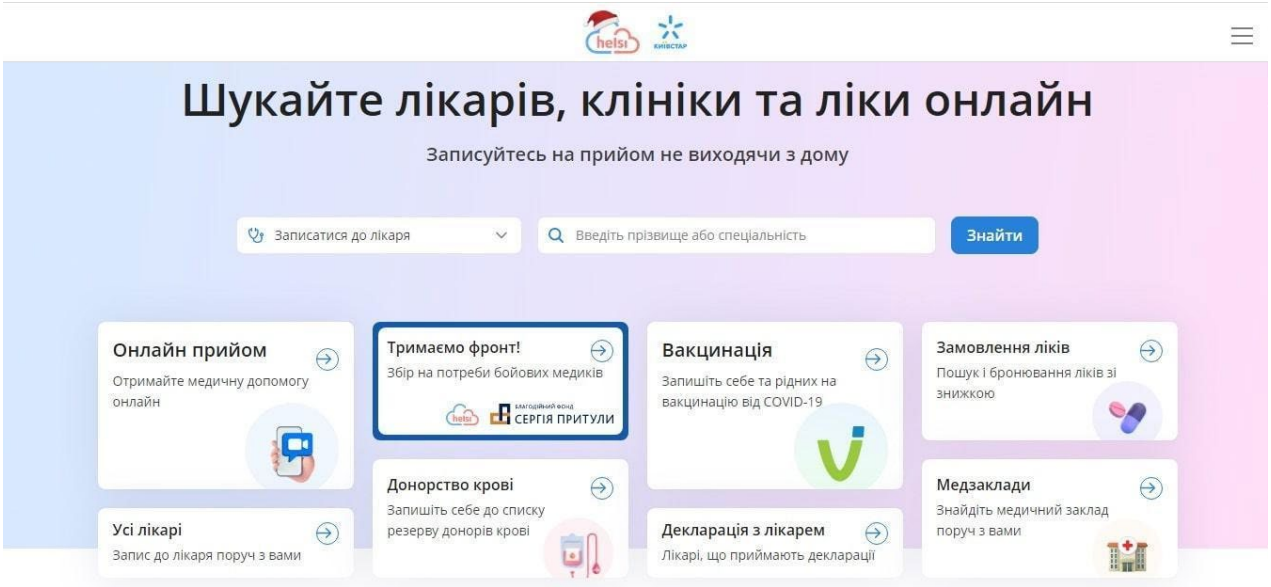

Рис. 1.1. Зовнішній вигляд веб-інтерфейса Хелсі

МІС «ЕМСіМЕД» - медична інформаційна система, що включена до переліку систем, рекомендованими Міністерством охорони здоров’я України. ЕМСіМЕД об’єднався з медичною інформаційною системою Health24 та є підключеним до ЦБД eHealth. Запроваджено механізм захисту інформації КСЗІ, інформація зберігається в хмарному сховищі. Великий об’єм модулів та компонентів.

МІС «Доктор Елекс» - електронна система охорони здоров’я, яка забезпечує автоматизацію ведення обліку медичних послуг та управління медичною інформацією в електронному вигляді. «Доктор Елекс» має здатність співпрацювати з медичним обладнанням. Інформаційна безпека становить збереження даних в хмарному сховищі.

1.2 Призначення розробки та галузь застосування

Веб-інтерфейс медичної картки пацієнта призначений для лікарняних установ, що надає можливість переглядати, змінювати або видаляти інформацію. Доступ до цієї інформації отримуватимуть лише зареєстровані користувачі, що гарантує більш високий рівень безпеки.

В програмі реалізован розділ картки пораненого ТССС, яка містить таку інформацію як: дані постраждалого, інформацію про поранення, ідентифікаційний номер військового, надану допомогу, ліки.

Галузь застосування може бути як і державна так і приватна лікарняна установа.

1.3. Підстава для розробки

В кінці навчання, студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою.

Підставою для розробки кваліфікаційної роботи на тему «Розробка веб-інтерфейсу медичної картки пацієнта» є наказ по Національному технічному університету «Дніпровська політехніка» від 350-с від 16.05.2023.

1.4. Постановка завдання

Метою кваліфікаційної роботи є створення веб-інтерфейсу для медичних закладів, що дозволить передивлятись, змінювати або видаляти інформацію, пов'язану з медичними даними пацієнтів. Для гарантування безпеки та конфіденційності інформація надається лише зареєстрованим користувачам. Інтерфейс забезпечить збереженню медичних даних та відстежуванню їх зміни.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для отримання доступу до веб-інтерфейсу, лікарі або медичний персонал повинні зареєструватись в системі, з метою перевірки і автентифікації користувача.

Процес роботи з веб-інтерфейсом та керування ним, здійснюється за допомогою таких пристроїв введення, як клавіатура та компьютерна миш.

1.5.2. Вимоги до інформаційної безпеки

Для забезпечення інформаційної безпеки запроваджено форму реєстрації та вхід до веб-інтерфейсу, лише за наявністю логіну та пароля.

База даних H2 запроваджує шифрування паролів користувачів, які зберігаються в таблиці.

За допомогою програмного середовища Spring Security, вказані у коді сторінки, знаходяться під закритим доступом для незареєстрованих користувачів або користувачів, які не володіють певними привілеями.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для повноцінного функціонування програмного забезпечення необхідні:

- персональний комп’ютер,
- клавіатура;
- миша.

Фіксованих системних вимог до комп’ютера немає, оскільки достатньо мати комп'ютер з установленою операційною системою та браузер за вподобанням.

1.5.4. Вимоги до інформаційної та програмної сумісності

Даний веб-інтерфейс сумісний з операційними системами Windows 10, 32 чи 64-розрядних версій, Ubuntu Interim та LTS. Також, сумісний з браузерами Google Chrome, Opera, Internet Explorer, Mozilla.

Для надійності роботи з медичними даними бекенд веб-інтерфейса має бути розроблений за допомогою мови програмування Java, а фронтенд з використанням мови розмітки HTML і таблиці стилів CSS.

РОЗДІЛ 2.
ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1 Функціональне призначення програми

Функціональним призначенням веб-інтерфейсу є можливість працювати з даними текстового формату. Наприклад, виконувати такі дії, як ввод та вивід інформації, оновлення, видалення та їх зберігання у базі даних.

Такий спосіб використання даних має суттєво зменшити час, який медичний персонал витрачає на опрацювання даних в паперовому форматі.

Для роботи достатньо пройти процес реєстрації або бути вже зареєстрованим користувачем та здійснити вхід до сайту.

2.2 Опис застосованих математичних методів

При розробленні програмного забезпечення для кваліфікаційної роботи застосовувався, такий математичний метод, як статистичний аналіз.

Його застосування виявляється у вивченні процесів створення МІС та дослідженню медичної сфери в цілому, а також кібербезпеки МІС. Метою було аналізувати вже існуючі аналоги МІС та запровадження методи кібербезпеки в системі.

Збір та аналіз отриманої інформації, дозволив ретельно розглянути можливі інциденти витоку інформації та методи забезпечення її безпеки. Цей аналіз сприяв виявленню потенційних загроз та уразливостей у системах, а також вивченню превентивних заходів для запобігання таким інцидентам.

Використання статистичного аналізу дозволило зробити об'єктивну оцінку ризиків та визначити оптимальні дії для забезпечення кібербезпеки МІС.

2.3 Опис використаної архітектури та шаблонів проектування

В якості архітектури додатка використана клієнт-серверна архітектура. Клієнт відправляє запит на сервер, де він обробляється і готовий результат відправляється клієнту. Також, реалізовані зберігання, читання та захист даних.

При розробці даної кваліфікаційної роботи застосовується мова програмування Java, яка є об'єктно-орієнтованою. Вагомим інструментом при розробці є фреймворк Spring Security, що дає змогу здійснювати аутентифікацію та контролювати доступ до інформації.

В якості бази даних взято СУБД H2, де використовуються типи відношення, такі як: один-до-багатьох та багато-до-багатьох.

2.4 Опис використаних технологій та мов програмування

Веб-інтерфейс для кваліфікаційної роботи розроблено на мові програмування Java у середовищі розробки IntelliJ IDEA Ultimate.

Java - є мовою програмування, яка базується на об'єктно-орієнтованому підході, тому підтримує поліморфізм, капсуляцію, створення класу та його об'єкта. Вона широко використовується для розробки різноманітного програмного забезпечення, такого як веб-додатки, мобільні додатки та інші типи програм.

Вагома перевага Java - її адаптивність, що дає здатність запускатися на будь-якій платформі, яка підтримує віртуальну машину Java (JVM). Це охоплює такі платформи як Windows, Linux, Mac OS та інші.

Функціонування програми повністю обмежується віртуальної машиною JVM. Java код може бути виконаний тільки в безпечній віртуальній машині, тим самим забезпечуючи відокремлення від потенційно небезпечних операцій.

Spring Framework - це потужна інфраструктура, яка розширює можливості у створенні надійних веб-додатків. Вона спрощує процес передачі даних,

оскільки прямо взаємодіє з JDBC, Hibernate та іншими, що дозволяє забезпечити ефективну роботу з даними.

Його частиною є фреймворк Spring Security, що дозволяє працювати з аутентифікацію та авторизацію, а також з іншими можливими забезпеченнями безпеки для розробляємих додатків.

Далі представлено ключові об’єкти Spring Security:

- SecurityContextHolder, що містить в собі інформацію про дану безпеку додатка, котрий включає в себе детальну інформацію про користувача, котрий на даний момент часу працює з додатком.
- SecurityContext, містить об’єкт Authentication і необхідну інформацію системи безпеки, пов’язану з надісланим запитом від користувача.
- GrantedAuthority відображає ролі розподіленні користувачам у представленому додатку, такі ролі як ROLE_USER, ROLE_ADMIN.
- UserDetails зберігає в собі інформацію, яка пізніше буде використана для аутентифікації та авторизації користувача.

2.5 Опис структури програми та алгоритмів її функціонування

Загальну схему роботи веб-інтерфейсу наведено на наступному рисунку (рис.2.1):

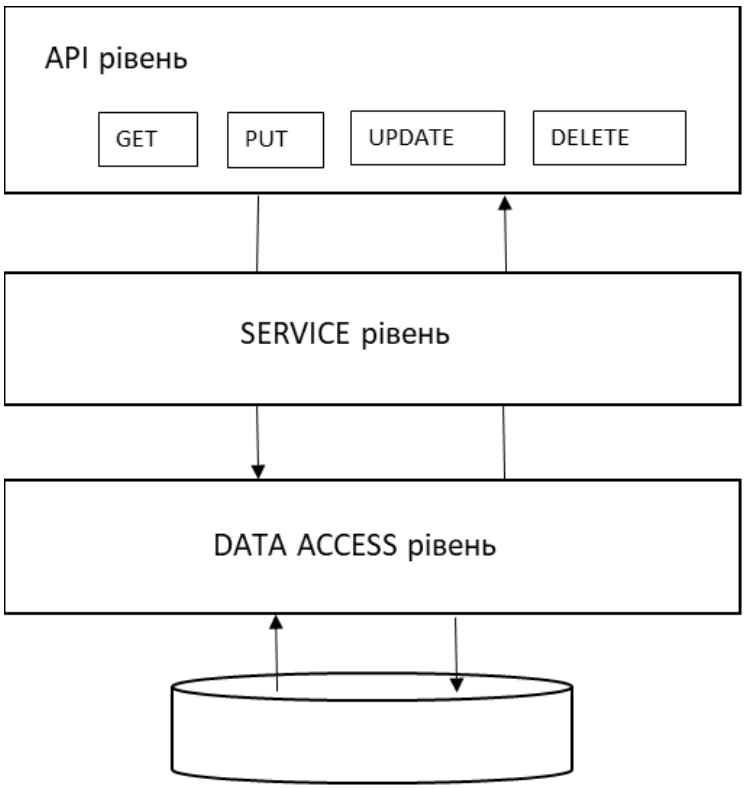

Рис. 2.1. Загальна схема роботи веб-інтерфейсу

Отже, першим є прикладний програмний інтерфейс або API рівень, він потрібен для роботи з даними, такими як, надсилання та отримання. API рівень дозволяє уникнути дублювання коду, що полегшує читання коду.

В розробленому веб-інтерфейсі було створено такі опрацювання даних як: отримання, читання, редагування та видалення інформації.

У класі PostController, що підтримує обробку на запит користувача, використовуються такі основу відображення, тобто mapping, як:

- PostMapping - відповідає за створення HTTP запиту на певний метод обробки;
- GetMapping - отримання певного HTTP запиту на певний метод обробки;
- PutMapping - редагування певного HTTP запиту на певний метод обробки;

Вже на цьому етапі додається Spring Security, метод @PreAuthorize(), де в дужках вказується яку роль має мати користувач, щоб отримати доступ до того чи іншого HTTP запиту.

Наступним йде сервісний рівень. Якщо Controller відповідав за певні HTTP запити та отримання інформації, то сервісний рівень відповідає за бізнес-логіку. Сервісний рівень пов’язаний з контроллером, контролює робочий процес. Наприклад, в сервісному рівні задається отримання даних, збереження, їх видалення тощо.

Рівень доступу до бази даних - це рівень, що запроваджує доступ до БД. Цей рівень використовується окремо від бізнес-логіки, щоб при випадку зміни сховища даних, не було потреби переписувати увесь код.

Далі на рисунку (рис. 2.2) наведено схему БД «сутність-зв’язок»:

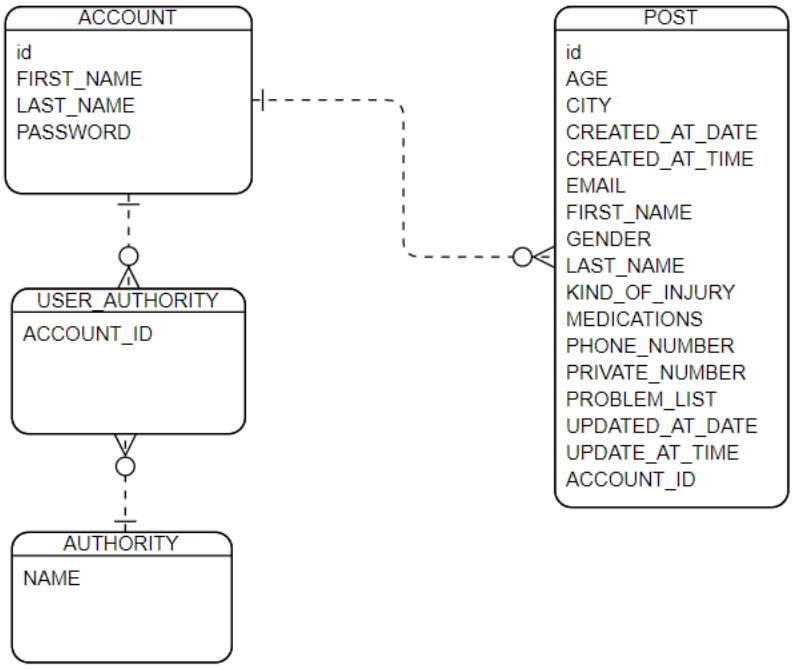

Рис. 2.2. Модель «сутність-зв’язок»

Тобто, користувач може мати свою роль, а роль прив’язана до користувача. Один користувач може створювати безліч постів, але пост може мати тільки одного користувача, що є зв’язком «один-до-багатьох». Так само, один користувач може мати декілька прав, а роль може мати декілька користувачів.

Створення зв’язків та колонок для БД реалізується через об’єкти до яких ми анотуєму @Entity.

У файлі application.properties прописується адреса за якою доступна БД, логін та пароль для входу, має такий вигляд:

spring.datasource.url=jdbc:h2:file:./data/medapp

spring.datasource.username=admin

spring.datasource.password=password

На наступному рисунку зображено таблицю зв’язків між користувачем та правами (рис.2.3).

SELECT * FROM USER_AUTHORITY;

ACCOUNT_ID	AUTHORITY_NAME
1	ROLE_USER
2	ROLE_ADMIN
2	ROLE_USER

Рис. 2.3. Зв’язок між користувачем та правами

Тобто, перший користувач з власним номером один, має лише одне право - користувач, другий користувач з власним номером два, має два права - користувача та адміністратора.

2.5.1 Back-end технології

Back-end реалізований мовою програмування Java та з застосуванням фреймворку Java Spring Boot. Задля забезпечення аутентифікації, авторизації та інших форм безпеки використовується фреймворк Spring Security, що є частиною Java Spring Boot.

В якості бази даних використовується H2, що є Java SQL базою даних.

2.5.2 Front-end технології

При розробці Front-end використовувались HTML, CSS та JavaScript. Також, Thymeleaf - що є серверним шаблоном Java, який забезпечує коректне відображення HTML, CSS, JavaScript в браузерах.

2.6 Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними в даному веб-інтерфейсі, є дані що користувач вносить у текстові поля. Дані вносяться за допомогою пристрою введення - клавіатури. До того ж, дані, наприклад при реєстрації користувача, можуть вносити тільки ті користувачі, що авторизовані як адміністратор.

Вхідні дані можуть бути тільки текстовими або чисельними. Для чисельних типів даних використовується тип Long, що дає можливість зберегти великий об’єм інформації.

Вихідними даними є дані, що відображаються у формах, таблицях. Це дані, які до цього було введені у текстові форми, користувачем. Вихідними даними, так само, є текстові або чисельні типи даних.

Вся інформація про людей, що представлена в кваліфікаційній роботі є вигаданою і не має стосунку до реальних.

2.7 Опис роботи програмного продукту

На наступних рисунках зображено блок-схема алгоритму дій програмного забезпечення (Рис. 2.4 - 2.5).

Рис. 2.4. Блок-схема алгоритму дій

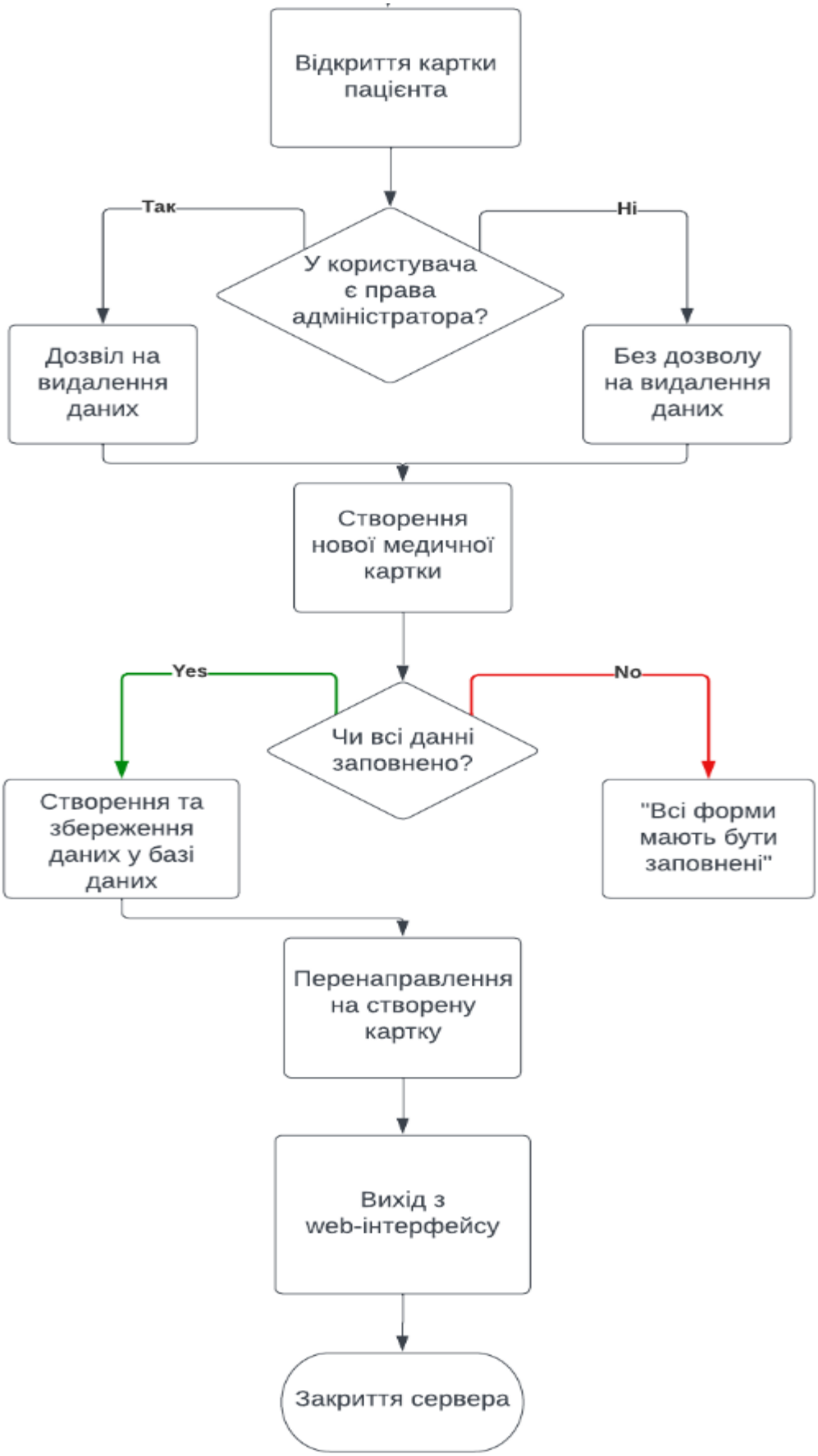

Рис. 2.5. Блок-схема алгоритму дій

2.7.1 Використані технічні засоби

Для запуску веб-інтерфейсу достатньо мінімальних ресурсів комп’ютера, таких як:

- персональний комп’ютер чи ноутбук з встановленою версією Java 17;
- клавіатура;
- миша.

2.7.2 Використані програмні засоби

Веб-інтерфейс працює на комп’ютерному обладнання з операційною системою Windows 10, 32 чи 64-розрядних версій та в браузерах Google, Mozilla, Opera, Internet Explorer. На комп’ютер має бути встановлена версія Java 17.

2.7.3 Виклик та завантаження програми

Для запуску веб-інтерфейсу необхідно запустити основу конфігурацію WebMedicalApplication (рис.2.6).

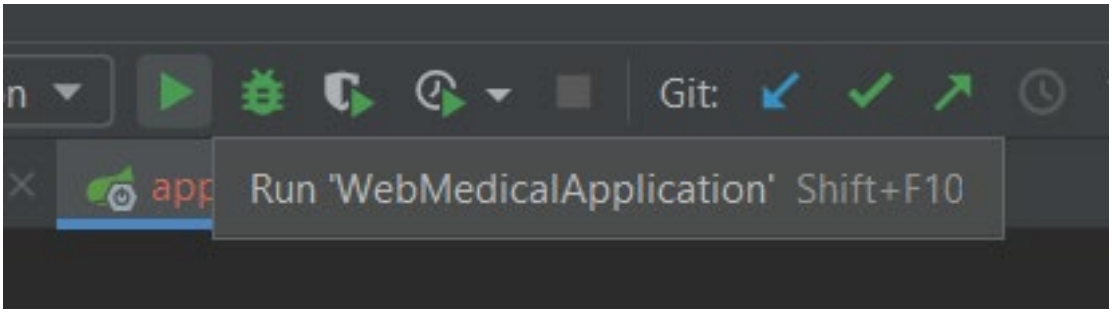

Рис. 2.6. Запуск додатка

В Spring Boot вбудованим сервером за замовчуванням є Tomcat. У файлі application.properties вказуємо за яким саме портом буде доступен додаток:

#server.port=3000

Тепер для доступу до веб-інтерфейсу в поле пошуку ввести localhost:3000 та буде виведено головну сторінку з пропозицією автентифікуватись (рис.2.7).

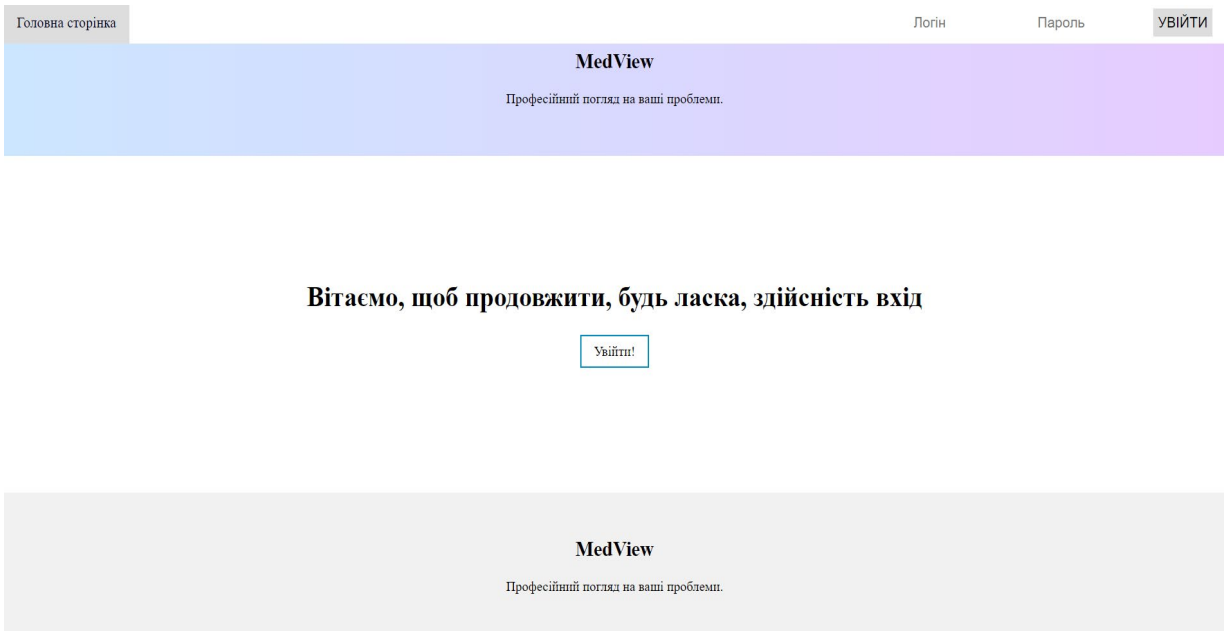

Рис. 2.7. Головна сторінка до автентифікації користувача

2.7.4 Опис інтерфейсу користувача

Подальші, користувач може увійти натиснувши кнопку «Увійти», що відкриє йому форму входу (рис. 2.8)

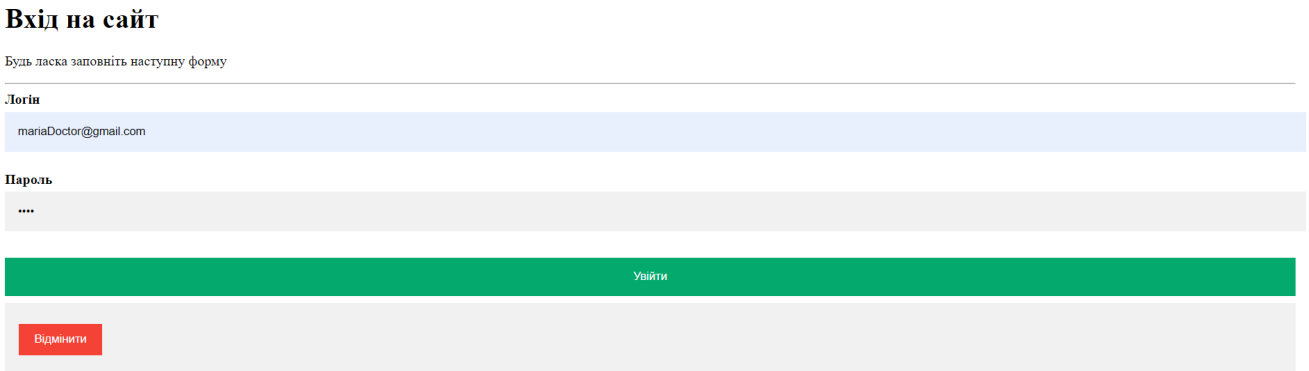

Рис. 2.8. Форма входу

Або заповнити форму у правому верхньому куті, що розташована на панелі навігації (рис. 2.9).

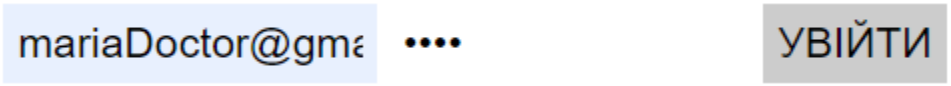

Рис. 2.9. Форма входу на панелі навігації

Далі буде представлено веб-інтерфейс для користувача з ролю адміністратора.

Отже, здійснивши вхід, на головному меню та в навігаційній панелі відображаються нові деталі. По-першу в навігаційному стають доступні три нові розділи це:

- Нова картка - створення нової картки для пацієнта;
- Картка пораненого ТССС - створення нової картки для пораненого військового;
- Зареєструвати користувача - зареєструвати нового користувача, доступно лише для користувача з правами адміністратора;
- Вихід - вихід з сайту.

Також, в шапці з’являється ім’я користувача, який здійснив автентифікації. Всі наведені пункти зображено на рисунку 2.10.

Рис. 2.10. Навігаційна панель та шапка веб-інтерфейса

На головній сторінці тепер відображається таблиця з пацієнтами, що вже були внесені в БД (рис. 2.11). Для більшої зручності пошуку конкретного пацієнта додано функції пошуку та очищення. Достатньо ввести потрібне ім’я,

прізвище або пошту та натиснути «пошук», якщо потрібно повернутись до попередньої таблиці натиснути «очистити».

У таблиці є п’ять колонок, це:

- Дія - що містить в собі «деталі», посилання на повну інформацію про пацієнта;
- Прізвище - прізвище пацієнта;
- Ім’я - ім’я пацієнта;
- Місто - місто проживання пацієнта;
- Електронна адреса - пошта за якою можна звернутись до пацієнта.

Введіть ключове слово.. | Шукати | Очистити

Дія	Прізвище	Ім'я	Місто	Електронна адреса
Деталі...	Залізняк	Максим	Дніпро	maksym@gmail.com
Деталі...	Феба	Коваль	Ужгород	feba@gmail.com
Деталі...	Батраченко	Василь	Запоріжжя	vasyl@kk
Деталі...	Щука	Олександр	Ужгород	shuka@kk
Деталі...	Бречко	Марія	Суми	maria@kk
Деталі...	Лутай	Євгеній	Кривий-Ріг	lutay@kk
Деталі...	Батраченко	Василь	Запоріжжя	vasyl@kk
Деталі...	Щука	Олександр	Ужгород	shuka@kk
Деталі...	Бречко	Марія	Суми	maria@kk
Деталі...	Лутай	Євгеній	Кривий-Ріг	lutay@kk
Деталі...	Батраченко	Василь	Запоріжжя	vasyl@kk
Деталі...	Щука	Олександр	Ужгород	shuka@kk
Деталі...	Бречко	Марія	Суми	maria@kk
Деталі...	Лутай	Євгеній	Кривий-Ріг	lutay@kk

Рис. 2.11. Таблиця пацієнтів, що вже містяться в БД

На наступному рисунку наведено приклад сортування за: конкретним ім’я або за конкретним місцем проживання. На прикладі з ім’ям, це єдине ім’я, що є у таблиці, тому виводиться тільки один результат (рис.2.12).

Феба | Шукати | Очистити

Дія	Прізвище	Ім'я	Місто	Електронна адреса
Деталі...	Феба	Коваль	Ужгород	feba@gmail.com

Рис. 2.12. Пошук в таблиці за конкретним ім’ям

На прикладі з сортуванням за місцем проживання виводиться три результати пацієнтів, що проживають в Запоріжжі (рис. 2.13). Такий пошук

може знадобитись у разі потреби дізнатись яка кількість пацієнтів проживає у конкретному місті.

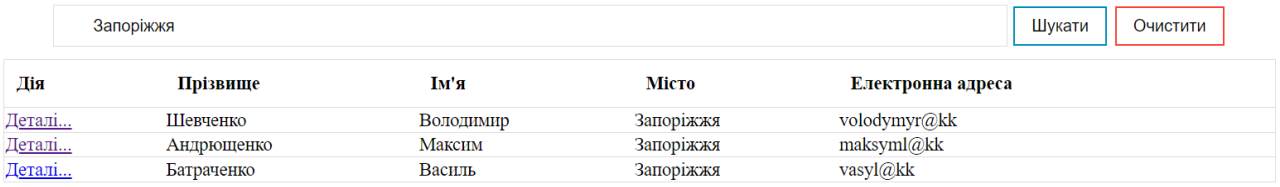

Рис. 2.13. Пошук в таблиці за конкретним містом проживання

Перейшовши за посиланням на детальну інформацію про пацієнта відкривається основна медична картка, що містить в собі таку особисту інформацію: електронна пошта, номер телефона, ім’я та фамілія пацієнта, стать, вік, місто проживання, лист скарг, в якому описуються основі жалоби пацієнта та назначенні медикаменти (рис. 2.14). Так само, у картці зображено дату, час створення або дату та час редагування, ким була написана картка. А втім, у ролі адміністратора є доступ до їх видалення, для користувача без прав, ця функція не відображаються.

Рис. 2.14. Медична картка пацієнта для користувача з роллю адміністратора

Перейшовши за посиланням для редагування даних, користувачу відкривається текстові форми з деталями пацієнта, але вже з доступом їх

змінити. Здійснивши потрібні зміни користувачу залишається натиснути кнопку розміщену унизу «редагувати».

На рисунку 2.15 зображено картку пацієнта, що була на попередньому рисунку, але вже з іншою інформацією.

Рис. 2.15. Вигляд попередньої картки, але вже з новою інформацією

Також, змінилась дата та час останнього редагування.

Після видалення картки, користувача повертає на головну сторінку, а інформація про цього пацієнта зникає у таблиці, так і з БД.

Для створення нової картки користувачу необхідно перейти за навігаційною панеллю за посиланням «нова картка». Тут користувачу доступні пусті текстові форми для створення нової картки (рис. 2.16), заповнення яких є обов’язковим. Після заповнення, як і з редагуванням, унизу треба натиснути кнопку «створити». Користувачу одразу відкриється нова картка та вона ж з’явиться у таблиці на головній сторінці та БД.

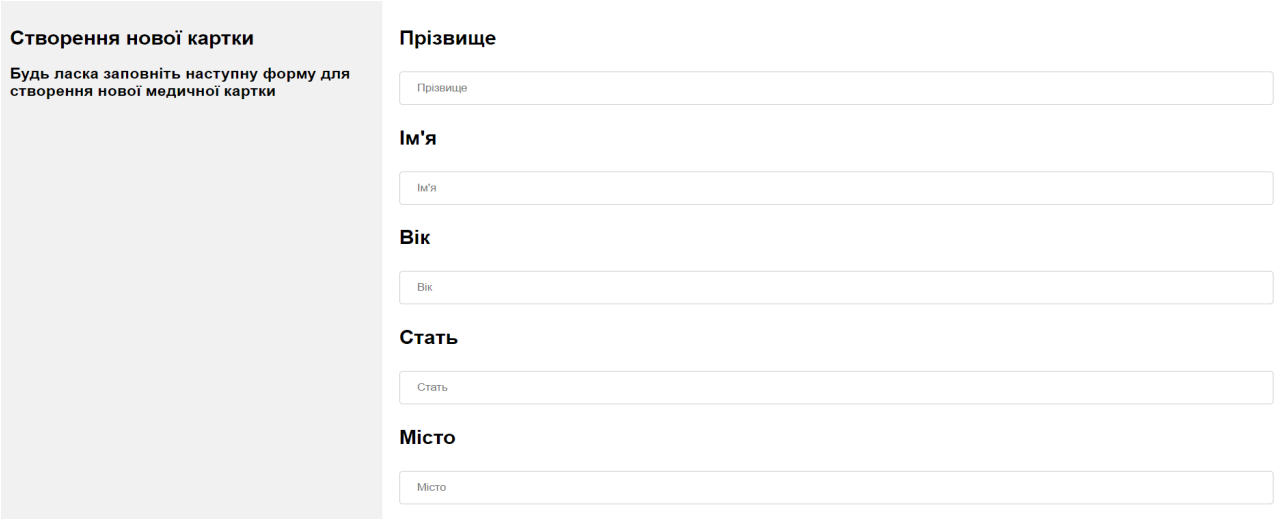

Рис. 2.16. Вигляд сторінки для створення нової картки пацієнта

Картка пораненого ТССС має схожі дані для введення, але є додаткова інформація така як: ідентифікаційний номер військового та механізм травми. Так само, інформація про створення картки, в особистій інформації, дає змогу зареєструвати о котрій було доставлено військового. Приклад вже існуючої картки пораненого наведено на рисунку 2.17.

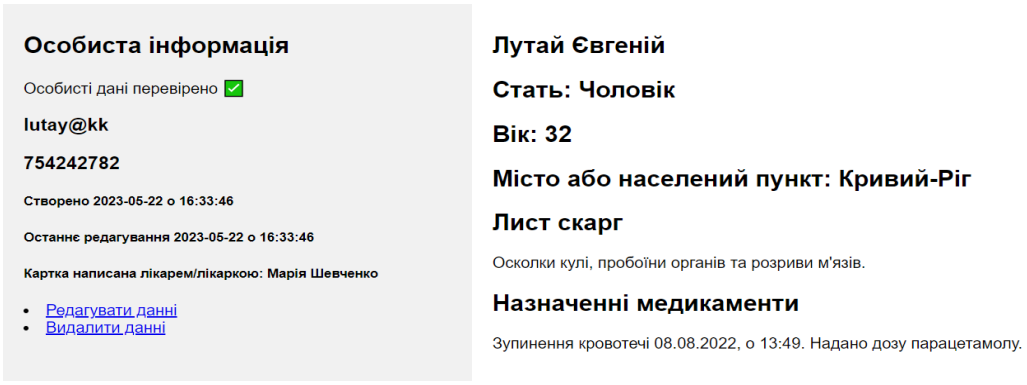

Рис. 2.17. Картка пораненого ТССС

Зареєструвати нового користувача може лише користувач з рівнем адміністратора. Реєстрація знаходиться на панелі навігації. Перейшови на сторінку реєстрації з’являється форма для заповнення. Дані що потрібно заповнити такі: ім’я, прізвище, електронна пошта та пароль. Всі дані є

обов’язковими для введення. Після заповнення натиснувши кнопку «зареєструвати» у базі даних з’явиться інформація про нового користувача.

Приклад заповнення форми наведено на наступному рисунку (рис. 2.18).

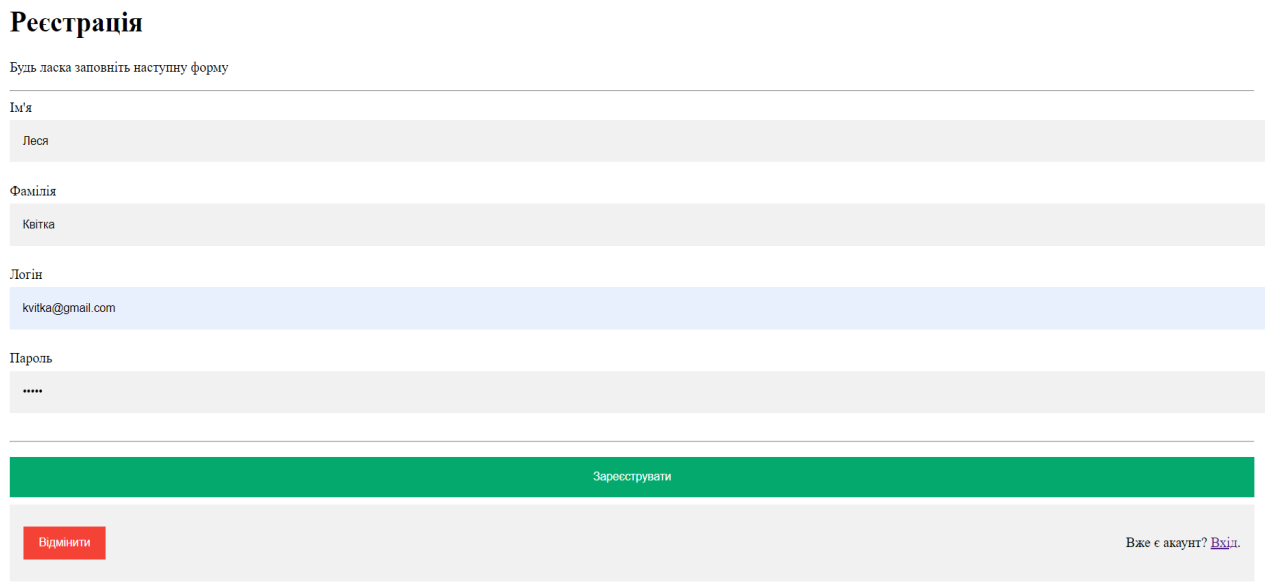

Рис. 2.18. Приклад заповнення форми реєстрації

Далі показано оновлену таблицю баз даних (рис.2.19).

SELECT * FROM ACCOUNT;

ID	EMAIL	FIRST_NAME	LAST_NAME	PASSWORD
1	olenaDoctor@gmail.com	Олена	Лутай	$2a$10$PifNDSiV.P4zdhg.MdwAZef3m.dvnBkvhp/pj/U0ki/IXtZM300ve
2	mariaDoctor@gmail.com	Марія	Шевченко	$2a$10$.Z5XSu5S5Z4zXGmINVOCxOr9tYh/rEdlRygK1DAQBUm.5Zcdz7C9a
3	kvitka@gmail.com	Леся	Квітка	$2a$10$MJPf8WazMvmRE9To.qMswOkt3xHww6ZeQCdnbMKHk2xhq3X4egDai

(3 rows, 5 ms)

Рис. 2.19. Оновлена інформація в БД

Також, перевіривши таблицю в БД можна дізнатись, що новий користувач був створений з правами адміністратора.

Здійснити вихід можна натиснувши кнопку на панелі навігацій у правому кутку. Після натиснення кнопку користувача поверне на сторінку входу.

Далі буде представлено веб-інтерфейс для користувача з ролю користувача. Ця роль відрізняється від адміністратора тим, що користувач не

може реєструвати нових користувачів, так само не може видалити картку пацієнта.

Головна сторінка залишилось такою самою, з таблицею даних про пацієнтів, але з навігаційної панелі зникло посилання на реєстрацію нового користувача. Навіть якщо в сторці ввести HTTP запит «http://localhost:3000/register», то це видасть помилку.

Переходячи на картку пацієнта, в лівій строці в розділі «Особиста інформація» зникла можливість видалити картку. Всі інші функції, такі як, редагування або ж створення нової картки залишаються можливими.

У разі того що сторінка не існує користувач отримує сторінку з помилкою (рис. 2.20).

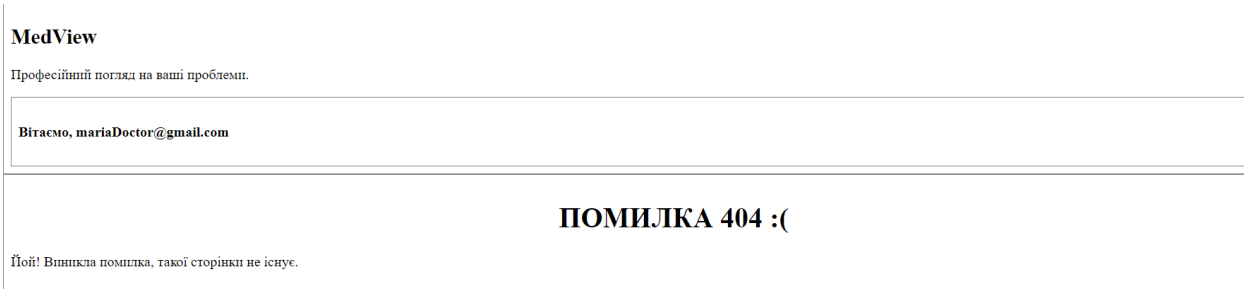

Рис. 2.20. Сторінка з помилкою

РОЗДІЛ 3
ЕКОНОМІЧНА ЧАСТИНА

3.1. Визначення трудомісткості та вартості розробки програмного продукту

Початкові дані:

1. передбачуване число операторів програми –2098;
2. коефіцієнт складності програми – 1,3;
3. коефіцієнт корекції програми в ході її розробки – 0,2;
4. годинна заробітна плата програміста– 90 грн/год;
5. коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2;
6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,08;
7. вартість машино-години ЕОМ – 6 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{, людино-годин,} \quad (3.1)$$

де t_o- витрати праці на підготовку й опис поставленої задачі (приймається 50 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a- витрати праці на розробку блок-схеми алгоритму;

t_n-витрати праці на програмування по готовій блок-схемі;

$t_{отл}$-витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмногму забезпеченні, яке розробляється.

Умовне число операторів:

$$Q = q \cdot C \cdot (1 + p), \quad (3.2)$$

де q - передбачуване число операторів (2098);

C - коефіцієнт складності програми (1,3);

p - коефіцієнт корекції програми в ході її розробки (0,2).

$$Q = 2098 * 1{,}3 * (1 + 0.2) = 3272{,}88$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85)K}, \text{ людино-годин}, \quad (3.3)$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі, В=1.2 ... 1.5;

K – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. до 2 – 1,08;

$$t_u = \frac{3272{,}88 * 1{,}2}{85 * 1{,}08} = 42{,}8, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20..25)K} \tag{3.4}$$

$$t_{\text{а}} = \frac{3272,88}{20 * 1,08} = 151,5, \text{людино} - \text{годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)K} \tag{3.5}$$

$$t_n = \frac{3272,88}{22 * 1,08} = 137,7, \text{людино} - \text{годин.}$$

Витрати праці на налагодження програми на ЕОМ:

- за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)K} \tag{3.6}$$

$$t_{\text{отл}} = \frac{3272,88}{5 * 1,08} = 606,1, \text{людино} - \text{годин.}$$

- за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1,2 \cdot t_{отл} \tag{3.7}$$

$$t_{отл}^{\text{к}} = 1,2 * 606,1 = 727,32, \text{людино} - \text{годин.}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до} \quad (3.8)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису

(3.9)

$$t_{др} = \frac{Q}{(15..20)K}$$

$$t_{др} = \frac{3272,88}{20 * 1,08} = 151,5, \text{людино} - \text{годин}.$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0,75 \cdot t_{др} \quad (3.10)$$

$$t_{до} = 0,75 * 151,5 = 113,6, \text{людино-годин}.$$

$$t_д = 151,5 + 113,6 = 265,1, \text{людино-годин}.$$

Отримаємо трудомісткість розробки програмного забезпечення:

t = 50 + 42,8 + 151,5 + 137,7 + 727,32 + 265,1 = 1374,42, людино-годин.

У результаті ми розрахували, що в загальній складності необхідно 1374,421 людино-годин для розробки даного програмного забезпечення.

3.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $K_{по}$ включають витрати на заробітну плату виконавця програми $З_{зп}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$K_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн}, \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн}, \quad (3.12)$$

де t – загальна трудомісткість, людино-годин;

$C_{ПР}$ – середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 1374{,}42 * 40 = 123697{,}8, \text{грн}.$$

$З_{МВ}$– Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot C_{М}, \text{ грн}, \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$C_{МЧ}$ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 727{,}32 * 6 = 4363{,}92, \text{ грн}.$$

$$К_{ПО} = 123697{,}8 + 4363{,}92 = 128061{,}72, \text{ грн}.$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ мес}. \quad (3.14)$$

де B_k - число виконавців;

F_p - місячний фонд робочого часу (при 40 годинному робочому тижні Fp=176 годин).

$$T = \frac{1374{,}42}{1 * 176} = 7{,}8 \text{ міс.}$$

Висновки. Час розробки даного програмного забезпечення складає 1374,421 людино-годин. Таким чином, очікувана тривалість розробки складе 7,8 місяця при 40 годинному робочому тижні (місячний фонд робочого часу 176 годин), а витрати на створення програмного забезпечення складатимуть 128061,72 грн.

ВИСНОВКИ

Після виконання кваліфікаційної роботи та вивчення додаткової літератури, отримані наступні висновки:

У даний час з розвитком технологій, автоматизація стає невід’ємною частиною різних сфер людського життя, включаючи медицину. В Україні було запроваджено систему eHealth, що є центральною базою даних для зберігання та обробки медичної інформації. Доступ до цих даних має Міністерство охорони здоров’я, НСЗУ та медичні установи, які підключенні до системи. Лікарня або приватні медичні заклади мають змогу підключитися до ЦБД через медичну інформаційну систему (МІС).

МІС для медичного закладу виконує не лише функцію передачі даних до системи ЦБД eHealth, але й розв'язує локальні завдання управління медичним закладом. Впровадження МІС дає змогу медичному персоналу швидко знаходити потрібну інформацію, здійснювати аналіз даних, а також вносити оновлення та коригувати записи пацієнтів. Це сприяє збереженню часу та зменшенню помилок при обробці даних.

Забезпечення безпеки медичних даних, які використовуються в МІС, є надзвичайно важливим аспектом. Шляхом впровадження заходів безпеки таких як, шифрування, аутентифікація та авторизація, може гарантуватись конфіденційність та захист особистих медичних даних пацієнтів.

Під час розробки даного програмного забезпечення було зосереджено увагу на поліпшенні доступу та відображені інформації, за допомогою простого інтерфейсу та вбудованої системи пошуку даних. Наприклад, у МІС «Доктор Елекс» вміщує у собі різноманітні модулі, що робить інтерфейс перевантаженим та складним для ока.

В розробленому веб-інтерфейсі запроваджене розділення користувачів за ролями та доступ до інформації лише авторизованим користувачам. Всі дані зберігаються в СУБД H2, доступ до якої супроводжується логіном та паролем, а такі дані, як пароль користувача, зашифровані.

Інтерфейс продукту є достатньо простим для працювання з даними.

Створений у рамках кваліфікаційної роботи веб-інтерфейс представляє собою продукт, що знадобиться як лікарняним установам, так і приватним медичним закладам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Електронна система охорони здоров'я в Україні [Електронний ресурс]. URL: https://ehealth.gov.ua/ (дата звернення: 11.01.2023)

2. EMCIMED. Medical Information system [Електронний ресурс]. URL: https://emci.ua/iak-pidkliuchytysia-do-ehealth/ (дата звернення: 15.01.2023)

3. MedCenter [Електронний ресурс]. URL: https://medcentercrm.com/blog/kak-elektronnaya-mediczinskaya-karta-paczienta-uproshhaet-rabotu-vracha/ (дата звернення: 15.01.2023)

4. Міністерство та Комітет цифрової трансформації України [Електронний ресурс]. URL: https://web.archive.org/web/20220118183046/https://thedigital.gov.ua/news/zyavilisya-pershi-rezultati-rozsliduvannya-napadu-khakeriv-na-sayti-derzhustanov (дата звернення: 12.01.2023)

5. Dr Choong May Ling, Mimi, IMDRF Chair Principles and Practices for Medical Device Cybersecurity : технічний документ / IMDRF, 2020. 46 с.

6. InfoWorld [Електронний ресурс], URL: https://www.infoworld.com/article/3272244/what-is-the-jvm-introducing-the-java-virtual-machine.html (дата звернення: 14.02.2023)

7. Microsoft [Електронний ресурс]. URL: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-java-spring-boot/ (дата звернення: 18.05.2023)

8. Spring [Електронний ресурс]. URL: https://docs.spring.io/spring-security/site/docs/3.0.x/apidocs/org/springframework/security/core/context/SecurityContextHolder.html (дата звернення: 03.04.2023)

9. Spring [Електронний ресурс]. URL: https://docs.spring.io/spring-security/site/docs/3.0.x/apidocs/org/springframework/security/core/context/SecurityContext.html (дата звернення: 03.04.2023)

10. Spring [Електронний ресурс]. URL: https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/GrantedAuthority.html (дата звернення: 03.04.2023)

11. Spring [Електронний ресурс]. URL: https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/userdetails/UserDetails.html (дата звернення: 05.04.2023)

12. Semaphore [Електронний ресурс]. URL: https://semaphoreci.com/blog/api-layer-react (дата звернення: 20.05.2023)

13. Spring [Електронний ресурс]. URL: https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/bind/annotation/package-summary.html (дата звернення: 12.04.2023

14. Baeldung [Електронний ресурс]. URL: https://www.baeldung.com/spring-service-layer-validation (дата звернення: 20.05.2023)

15. Medium [Електронний ресурс]. URL: https://medium.com/towards-polyglot-architecture/design-patterns-for-the-database-layer-7b741b126036 (дата звернення: 21.05.2023)

16. Spring [Електронний ресурс]. URL: https://spring.io/guides/gs/accessing-data-jpa/ (дата звернення: 16.04.2023)

17. H2 Database Engine [Електронний ресурс]. URL: https://www.h2database.com/html/features.html (дата звернення: 21.05.2023)

18. Spring [Електронний ресурс]. URL: https://spring.io/guides/gs/securing-web/ (дата звернення: 20.04.2023)

19. Thymeleaf [Електронний ресурс]. URL: https://www.thymeleaf.org/ (дата звернення: 21.05.2023)

20. in28minutes [Електронний ресурс]. URL: https://www.springboottutorial.com/java-programmer-essentials-what-is-an-embedded-server (дата зверненн: 20.05.2023)

21. Т. О. Назірова, О. Б. Костенко Огляд моделей розвитку eHealth та наявних медичних інформаційних систем. Проблеми створення єдиного медико-інформаційного простору / Науковий вісник НЛТУ України, 2017. 5 с.

Додаток А

ЛІСТИНГ ПРОГРАМИ

Даний програмний код було розміщено на GitHub за посиланням: https://github.com/diana427/web-medical-application

Файл SeedData

```
package com.example.webmedicalapplication.config;

import com.example.webmedicalapplication.models.Account;
import com.example.webmedicalapplication.models.Authority;
import com.example.webmedicalapplication.models.Card;
import com.example.webmedicalapplication.repositories.AuthorityRepository;
import com.example.webmedicalapplication.services.AccountService;
import com.example.webmedicalapplication.services.CardService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;

import java.util.HashSet;
import java.util.List;
import java.util.Set;

@Component
public class SeedData implements CommandLineRunner {

    @Autowired
    private CardService cardService;
    @Autowired
    private AccountService accountService;
    @Autowired
    private AuthorityRepository authorityRepository;

    @Override
    public void run(String... args) throws Exception {
        List<Card> cards = cardService.getAll("keyword");

        if (cards.size() == 0) {

            Authority admin = new Authority();
            admin.setName("ROLE_ADMIN");
            authorityRepository.save(admin);

            Authority user = new Authority();
            user.setName("ROLE_USER");
            authorityRepository.save(user);

            Account account1 = new Account();
            Account account2 = new Account();

            account1.setFirstName("Олена");
            account1.setLastName("Лутай");
            account1.setEmail("olenaDoctor@gmail.com");
            account1.setPassword("4321");
            Set<Authority> authorities1 = new HashSet<>();
            authorityRepository.findById("ROLE_USER").ifPresent(authorities1::add);
            account1.setAuthorities(authorities1);
```

```
            account2.setFirstName("Марія");
            account2.setLastName("Шевченко");
            account2.setEmail("mariaDoctor@gmail.com");
            account2.setPassword("4321");
            Set<Authority> authorities2 = new HashSet<>();
            authorityRepository.findById("ROLE_ADMIN").ifPresent(authorities2::add);
            authorityRepository.findById("ROLE_USER").ifPresent(authorities2::add);
            account2.setAuthorities(authorities2);

            accountService.save(account1);
            accountService.save(account2);

            Card card1 = new Card();
            card1.setLastName("Залізняк");
            card1.setFirstName("Максим");
            card1.setCity("Дніпро");
            card1.setAge(41);
            card1.setPhoneNumber(125458351L);
            card1.setEmail("maksym@gmail.com");
            card1.setGender("Чоловік");
            card1.setProblemList("Сильний головний біль, високий артеріальний тиск,
запаморочення.");
            card1.setMedications("Бісопролол");
            card1.setAccount(account1);

            Card card2 = new Card();
            card2.setLastName("Феба");
            card2.setFirstName("Коваль");
            card2.setCity("Ужгород");
            card2.setAge(38);
            card2.setPhoneNumber(854665781L);
            card2.setEmail("feba@gmail.com");
            card2.setGender("Жіноча");
            card2.setProblemList("Ниюча біль у спині");
            card2.setMedications("Курс терапії лікувальної гімнастики");
            card2.setAccount(account1);

            Card card3 = new Card();
            card3.setLastName("Шевченко");
            card3.setFirstName("Володимир");
            card3.setCity("Запоріжжя");
            card3.setAge(25);
            card3.setPhoneNumber(845151552L);
            card3.setEmail("volodymyr@kk");
            card3.setGender("Чоловік");
            card3.setProblemList("Проблеми з дихальною системою, астма");
            card3.setMedications("Бронходилататори, кортикостероїди");
            card3.setAccount(account2);

            Card card4 = new Card();
            card4.setLastName("Вугор");
            card4.setFirstName("Олександр");
            card4.setCity("Ужгород");
            card4.setAge(56);
            card4.setPhoneNumber(424275782L);
            card4.setEmail("shuka@kk");
            card4.setGender("Чоловік");
            card4.setProblemList("Проблеми зі слухом, воспалення вуха");
            card4.setMedications("Антибіотики, аудіологічна реабілітація");
            card4.setAccount(account2);

            Card card5 = new Card();
```

```
            card5.setLastName("Бречко");
            card5.setFirstName("Марія");
            card5.setCity("Суми");
            card5.setAge(21);
            card5.setPhoneNumber(457527782L);
            card5.setEmail("maria@kk");
            card5.setGender("Жіноча");
            card5.setProblemList("Депресія, тривожність");
            card5.setMedications("Терапія, антидеприсанти");
            card5.setAccount(account1);

            Card card6 = new Card();
            card6.setLastName("Лутай");
            card6.setFirstName("Євгеній");
            card6.setCity("Кривий-Ріг");
            card6.setAge(32);
            card6.setPhoneNumber(754242782L);
            card6.setEmail("lutay@kk");
            card6.setGender("Чоловік");
            card6.setPrivateNumber(1864L);
            card6.setKindOfInjury("Поранення від осколків");
            card6.setProblemList("Осколки кулі, пробоїни органів та розриви м'язів.");
            card6.setMedications("Зупинення кровотечі 08.08.2022, о 13:49. Надано дозу
парацетамолу.");
            card6.setAccount(account2);

            Card card7 = new Card();
            card7.setLastName("Андрющенко");
            card7.setFirstName("Максим");
            card7.setCity("Запоріжжя");
            card7.setAge(25);
            card7.setPhoneNumber(845151552L);
            card7.setEmail("maksyml@kk");
            card7.setGender("Чоловік");
            card7.setProblemList("Проблеми з дихальною системою, астма");
            card7.setMedications("Бронходилататори, кортикостероїди");
            card7.setAccount(account1);

            Card card8 = new Card();
            card8.setLastName("Щука");
            card8.setFirstName("Олександр");
            card8.setCity("Ужгород");
            card8.setAge(56);
            card8.setPhoneNumber(424275782L);
            card8.setEmail("shuka@kk");
            card8.setGender("Чоловік");
            card8.setProblemList("Проблеми зі слухом, воспалення вуха");
            card8.setMedications("Антибіотики, аудіологічна реабілітація");
            card8.setAccount(account1);

            Card card9 = new Card();
            card9.setLastName("Бречко");
            card9.setFirstName("Марія");
            card9.setCity("Суми");
            card9.setAge(21);
            card9.setPhoneNumber(457527782L);
            card9.setEmail("maria@kk");
            card9.setGender("Жіноча");
            card9.setProblemList("Депресія, тривожність");
            card9.setMedications("Терапія, антидеприсанти");
            card9.setAccount(account1);
```

```
            Card card10 = new Card();
            card10.setLastName("Лутай");
            card10.setFirstName("Євгеній");
            card10.setCity("Кривий-Ріг");
            card10.setAge(32);
            card10.setPhoneNumber(754242782L);
            card10.setEmail("lutay@kk");
            card10.setGender("Чоловік");
            card10.setPrivateNumber(1864L);
            card10.setKindOfInjury("Поранення від осколків");
            card10.setProblemList("Осколки кулі, пробоїни органів та розриви м'язів.");
            card10.setMedications("Зупинення кровотечі 08.08.2022, о 13:49. Надано дозу
парацетамолу.");
            card10.setAccount(account2);

            Card card11 = new Card();
            card11.setLastName("Батраченко");
            card11.setFirstName("Василь");
            card11.setCity("Запоріжжя");
            card11.setAge(25);
            card11.setPhoneNumber(845151552L);
            card11.setEmail("vasyl@kk");
            card11.setGender("Чоловік");
            card11.setProblemList("Проблеми з дихальною системою, астма");
            card11.setMedications("Бронходилататори, кортикостероїди");
            card11.setAccount(account2);

            Card card12 = new Card();
            card12.setLastName("Щука");
            card12.setFirstName("Олександр");
            card12.setCity("Ужгород");
            card12.setAge(56);
            card12.setPhoneNumber(424275782L);
            card12.setEmail("shuka@kk");
            card12.setGender("Чоловік");
            card12.setProblemList("Проблеми зі слухом, воспалення вуха");
            card12.setMedications("Антибіотики, аудіологічна реабілітація");
            card12.setAccount(account1);

            Card card13 = new Card();
            card13.setLastName("Бречко");
            card13.setFirstName("Марія");
            card13.setCity("Суми");
            card13.setAge(21);
            card13.setPhoneNumber(457527782L);
            card13.setEmail("maria@kk");
            card13.setGender("Жіноча");
            card13.setProblemList("Депресія, тривожність");
            card13.setMedications("Терапія, антидепресанти");
            card13.setAccount(account1);

            Card card14 = new Card();
            card14.setLastName("Лутай");
            card14.setFirstName("Євгеній");
            card14.setCity("Кривий-Ріг");
            card14.setAge(32);
            card14.setPhoneNumber(754242782L);
            card14.setEmail("lutay@kk");
            card14.setGender("Чоловік");
            card14.setPrivateNumber(1864L);
            card14.setKindOfInjury("Поранення від осколків");
            card14.setProblemList("Осколки кулі, пробоїни органів та розриви м'язів.");
```

```
            card14.setMedications("Зупинення кровотечі 08.08.2022, о 13:49. Надано дозу
парацетамолу.");
            card14.setAccount(account2);


            cardService.save(card1);
            cardService.save(card2);
            cardService.save(card3);
            cardService.save(card4);
            cardService.save(card5);
            cardService.save(card6);
            cardService.save(card7);
            cardService.save(card8);
            cardService.save(card9);
            cardService.save(card10);
            cardService.save(card11);
            cardService.save(card12);
            cardService.save(card13);
            cardService.save(card14);
        }
    }
}
```

Файл WebSecurityConfig.java

```
package com.example.webmedicalapplication.config;

import lombok.RequiredArgsConstructor;
import org.springframework.boot.autoconfigure.security.servlet.PathRequest;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.method.configuration.EnableMethodSecurit
y;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
@EnableWebSecurity
@EnableMethodSecurity(prePostEnabled = true, securedEnabled = true)
@RequiredArgsConstructor
public class WebSecurityConfig {

    @Bean
    public static PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
    private static final String[] WHITELIST = {
            "/h2-console/*",
            "/"
    };

@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

    http
            .csrf(csrf -> csrf.disable())
```

```
            .headers(headers -> headers.frameOptions().disable())
            .authorizeHttpRequests(auth -> auth
                    .requestMatchers("/css/**", "/js/**", "/images/**", "/fonts/**",
"/webjars/**").permitAll()
                    .requestMatchers("/").permitAll()
                    .requestMatchers("/rss/**").permitAll()
                    .requestMatchers("/register/**").hasRole("ADMIN")
                    .requestMatchers("/cards/**").permitAll()
                    .requestMatchers(PathRequest.toH2Console()).permitAll()
                    .anyRequest().authenticated()
            )
            .formLogin(form -> form
                    .loginPage("/login")
                    .loginProcessingUrl("/login")
                    .usernameParameter("email")
                    .passwordParameter("password")
                    .defaultSuccessUrl("/", true)
                    .failureUrl("/login?error")
                    .permitAll()
            );

    return http.build();
}
}
```

Файл HomeController.java

```
package com.example.webmedicalapplication.controllers;

import com.example.webmedicalapplication.models.Card;
import com.example.webmedicalapplication.services.CardService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import java.util.List;

@Controller
public class HomeController {
    @Autowired
    private CardService cardService;

    // home page
    @GetMapping("/")
    // getting data by specific keyword
    public String home(Model model, @Param("keyword") String keyword) {
        List<Card> cards = cardService.getAll(keyword);
        model.addAttribute("cards", cards);
        model.addAttribute("keyword", keyword);

        return "home";
    }
}
```

Файл LoginController.java

```
package com.example.webmedicalapplication.controllers;

import org.springframework.stereotype.Controller;
```

```
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class LoginController {

    @GetMapping("/login")
    public String getLoginPage() {

        return "login";
    }
}
```

Файл CardConroller.java

```
package com.example.webmedicalapplication.controllers;

import com.example.webmedicalapplication.models.Account;
import com.example.webmedicalapplication.models.Card;
import com.example.webmedicalapplication.services.AccountService;
import com.example.webmedicalapplication.services.CardService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;

import java.util.Optional;

@Controller
public class CardController {

    @Autowired
    private CardService cardService;

    @Autowired
    private AccountService accountService;

    // showing data into the page
    @GetMapping("/cards/{id}")
    public String getCard(@PathVariable Long id, Model model) {
        Optional<Card> optionalCard = cardService.getById(id);
        if (optionalCard.isPresent()) {
            Card card = optionalCard.get();
            model.addAttribute("card", card);
            return "card";
        } else { // if post isn`t found
            return "404";
        }
    }

    // create new card
    @GetMapping("/cards/new")
    public String creatNewCard(Model model) {
        Optional<Account> optionalAccount =
accountService.findUserByEmail("mariaDoctor@gmail.com");
        if (optionalAccount.isPresent()) {
            Card card = new Card();
```

```
            card.setAccount(optionalAccount.get());
            model.addAttribute("card", card);
            return "card_new";
        } else {
            return "404";
        }
    }

    // save new card
    @PostMapping("/cards/new")
    public String saveNewPost(@ModelAttribute Card card) {
        cardService.save(card);
        return "redirect:/cards/" + card.getId();
    }

    // new card for TCCC
    @GetMapping("/cards/new/tacmed")
    public String creatNewCardTacMed(Model model) {
        Optional<Account> optionalAccount =
accountService.findUserByEmail("mariaDoctor@gmail.com");
        if (optionalAccount.isPresent()) {
            Card card = new Card();
            card.setAccount(optionalAccount.get());
            model.addAttribute("card", card);
            return "card_new_tacmed";
        } else {
            return "404";
        }
    }

    // save card
    @PostMapping("/cards/new/tacmed")
    public String saveNewPostTacMed(@ModelAttribute Card card) {
        cardService.save(card);
        return "redirect:/cards/" + card.getId();
    }

    // editing specific card
    @GetMapping("/cards/{id}/edit")
    @PreAuthorize("isAuthenticated()")
    public String getCardForEdit(@PathVariable Long id, Model model) {

        // find post by id
        Optional<Card> optionalCard = cardService.getById(id);
        // if post exist put it in model
        if (optionalCard.isPresent()) {
            Card card = optionalCard.get();
            model.addAttribute("card", card);
            return "card_edit";
        } else {
            return "404";
        }
    }

    // save changes
    @PostMapping("/cards/{id}")
    @PreAuthorize("isAuthenticated()")
    public String updateCard(@PathVariable Long id, Card card, BindingResult result,
Model model) {
        Optional<Card> optionalCard = cardService.getById(id);
         if (optionalCard.isPresent()) {
             Card existingCard = optionalCard.get();
```

```
            existingCard.setLastName(card.getLastName());
            existingCard.setFirstName(card.getFirstName());
            existingCard.setAge(card.getAge());
            existingCard.setCity(card.getCity());
            existingCard.setEmail(card.getEmail());
            existingCard.setPhoneNumber(card.getPhoneNumber());
            existingCard.setProblemList(card.getProblemList());
            existingCard.setMedications(card.getMedications());

            cardService.save(existingCard);
        }

        return "redirect:/cards/" + card.getId();
    }

    // delete existing card
    @GetMapping("/cards/{id}/delete")
    @PreAuthorize("hasRole('ROLE_ADMIN')")
    public String deleteCard(@PathVariable Long id) {

        // find post by id
        Optional<Card> optionalCard = cardService.getById(id);
        if (optionalCard.isPresent()) {
            Card card = optionalCard.get();

            cardService.delete(card);
            return "redirect:/";
        } else {
            return "404";
        }
    }
}
```

Файл RegisterController.java

```
package com.example.webmedicalapplication.controllers;

import com.example.webmedicalapplication.models.Account;
import com.example.webmedicalapplication.services.AccountService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;


@Controller
public class RegisterController {
    @Autowired
    private AccountService accountService;

    // register new account
    @GetMapping("/register")
    public String getRegisterPage(Model model) {
        Account account = new Account();
        model.addAttribute("account", account);
        return "register";
    }
```

```
    // saving new account
    @PostMapping("/register")
    public String registerNewUser(@ModelAttribute Account account) {
        accountService.save(account);

        return "redirect:/";
    }
}
```

Файл Account.java

```
package com.example.webmedicalapplication.models;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.util.HashSet;
import java.util.List;
import java.util.Set;

@Entity
@Getter
@Setter
@NoArgsConstructor
public class Account {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long id;

    private String password;
    private String firstName;
    private String lastName;
    private String email;


    // one account can have many cards
    @OneToMany(mappedBy = "account")
    private List<Card> cards;

    // many account can have many authorities
    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(name = "user_authority",
    joinColumns = {@JoinColumn(name = "account_id", referencedColumnName = "id")},
    inverseJoinColumns = {@JoinColumn(name = "authority_name", referencedColumnName =
"name")})
    private Set<Authority> authorities = new HashSet<>();

    @Override
    public String toString() {
        return "Account{" +
                ", password='" + password + '\'' +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                ", email='" + email + '\'' +
                ", authorities=" + authorities +
                '}';
    }
}
```

Файл Authority.java

```
package com.example.webmedicalapplication.models;

import jakarta.persistence.*;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.io.Serializable;

@Entity
@Getter
@Setter
@NoArgsConstructor
public class Authority implements Serializable {

    @Id
    @Column(length = 16)
    private String name;

 }
```

Файл Card.java

```
package com.example.webmedicalapplication.models;

import jakarta.persistence.*;
import jakarta.validation.constraints.NotNull;
import lombok.Getter;
import lombok.Setter;

import java.time.LocalDate;
import java.time.LocalTime;

@Entity
@Getter
@Setter
public class Card {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long id;
    private String lastName;
    private String firstName;
    private Integer age;
    private Long privateNumber;
    private String kindOfInjury;
    private String gender;
    private Long phoneNumber;
    private String email;
    private String city;
    @Lob
    @Column(name="Problem List", columnDefinition = "CLOB")
    private String problemList;
    @Lob
    @Column(name="Medications", columnDefinition = "CLOB")
```

```
    private String medications;
    private LocalDate createdAtDate;
    private LocalTime createdAtTime;
    private LocalDate updatedAtDate;
    private LocalTime updatedAtTime;

    @NotNull
    // many cards can be connected only to one account
    @ManyToOne
    @JoinColumn(name = "account_id", referencedColumnName = "id", nullable = false)
    private Account account;

    @Override
    public String toString() {
        return "Card{" +
                "id=" + id +
                ", lastName='" + lastName + '\'' +
                ", firstName='" + firstName + '\'' +
                ", age=" + age +
                ", IdNumber=" + privateNumber +
                ", kindOfInjury='" + kindOfInjury + '\'' +
                ", gender='" + gender + '\'' +
                ", phoneNumber=" + phoneNumber +
                ", email='" + email + '\'' +
                ", city='" + city + '\'' +
                ", problemList='" + problemList + '\'' +
                ", medications='" + medications + '\'' +
                ", createdAtDate=" + createdAtDate +
                ", createdAtTime=" + createdAtTime +
                ", updatedAt=" + updatedAtDate +
                ", updatedAtTime=" + updatedAtTime +
                '}';
    }
}
```

Файл AccountRepository.java

```
package com.example.webmedicalapplication.repositories;

import com.example.webmedicalapplication.models.Account;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository
public interface AccountRepository extends JpaRepository<Account, Long> {
    Optional<Account> findUserByEmail(String email);
}
```

Файл AuthorityRepository.java

```
package com.example.webmedicalapplication.repositories;

import com.example.webmedicalapplication.models.Authority;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface AuthorityRepository extends JpaRepository<Authority, String> {
}
```

Файл CardRepository.java

```
package com.example.webmedicalapplication.repositories;

import com.example.webmedicalapplication.models.Card;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface CardRepository extends JpaRepository<Card, Long> {

    // sorting data using keyword
    @Query("SELECT p FROM Card p WHERE p.lastName LIKE %?1%"
    + "OR p.firstName LIKE %?1%"
    + "OR p.city LIKE %?1%"
    + "OR p.email LIKE %?1%"
    + "OR p.gender LIKE %?1%"
    + "OR CONCAT(p.age, '') LIKE %?1%")
    public List<Card> findAll(String keyword);

}
```

Файл AccountService.java

```
package com.example.webmedicalapplication.services;
import com.example.webmedicalapplication.models.Account;
import com.example.webmedicalapplication.models.Authority;
import com.example.webmedicalapplication.repositories.AccountRepository;
import com.example.webmedicalapplication.repositories.AuthorityRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.HashSet;
import java.util.Optional;
import java.util.Set;

@Service
public class AccountService {

    @Autowired
    private PasswordEncoder passwordEncoder;
    @Autowired
    private AccountRepository accountRepository;
    @Autowired
    private AuthorityRepository authorityRepository;

    public Account save(Account account) {

        if (account.getId() == null) {
            if (account.getAuthorities().isEmpty()) {
                Set<Authority> authorities = new HashSet<>();
                authorityRepository.findById("ROLE_ADMIN").ifPresent(authorities::add);
                account.setAuthorities(authorities);
```

```
            }
        }

        // encryption of password
        account.setPassword(passwordEncoder.encode(account.getPassword()));
        return accountRepository.save(account);
    }

    public Optional<Account> findUserByEmail(String email) {
        return accountRepository.findUserByEmail(email);
    }
}
```

Файл UserDetailsService.java

```
package com.example.webmedicalapplication.services;

import com.example.webmedicalapplication.models.Account;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Component;

import java.util.List;
import java.util.stream.Collectors;

@Component("userDetailService")
public class UserDetailsService implements
org.springframework.security.core.userdetails.UserDetailsService {

    @Autowired
    private AccountService accountService;

    @Override
    public UserDetails loadUserByUsername(String email) throws
UsernameNotFoundException {
        Account account = accountService.findUserByEmail(email)
                .orElseThrow(() -> new UsernameNotFoundException("Account not found"));

        List<GrantedAuthority> grantedAuthorities = account
                .getAuthorities()
                .stream()
                .map(authority -> new SimpleGrantedAuthority(authority.getName()))
                .collect(Collectors.toList());

         return new
org.springframework.security.core.userdetails.User(account.getEmail(),
account.getPassword(), grantedAuthorities);
    }
}
```

Файл CardService.java

```
package com.example.webmedicalapplication.services;

import com.example.webmedicalapplication.models.Card;
import com.example.webmedicalapplication.repositories.CardRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Optional;
import java.util.List;

@Service
public class CardService {

    @Autowired
    private CardRepository cardRepository;

    public Optional<Card> getById(Long id) {

        return cardRepository.findById(id);
    }

    // sorting data with keyword
    public List<Card> getAll(String keyword) {
        if(keyword != null) {
            return cardRepository.findAll(keyword);
        }

        return cardRepository.findAll();
    }

    // saving card with current data and time
    public Card save(Card card) {
        if (card.getId() == null) {
            card.setCreatedAtDate(LocalDate.now());
            card.setCreatedAtTime(LocalTime.now());
        }
        card.setUpdatedAtDate(LocalDate.now());
        card.setUpdatedAtTime(LocalTime.now());
        return cardRepository.save(card);
    }

    public void delete(Card card) {
        cardRepository.delete(card);
    }
}
```

Файл application.properties

```
# connecting server to port 3000
server.port=3000

spring.datasource.url=jdbc:h2:file:./data/medapp
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=admin
spring.datasource.password=password
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true
spring.h2.console.path=/h2-console

spring.jpa.hibernate.ddl-auto=create-drop
```

Файл StyleHome.css

```
body {
   margin: 0;
   font-family: Arial, Helvetica, sans-serif;
}

* {
   box-sizing: border-box;
}

.header {
   background-image: linear-gradient(to right,      rgb(204, 230, 255), rgb(230, 204,
255));
   padding: 40px;
   text-align: center;
}

.basement {
   background-color: #f1f1f1;
   padding: 30px;
   text-align: center;
}

.not-logout-container {
   text-align: center;
   padding: 120px;
}

.login-button {
   background-color: white;
   color: black;
   font-size: 16px;
   margin: 4px 2px;
   border: 2px solid #e1e1e1;
   padding: 8px 16px;
   text-align: center;
   text-decoration: none;
   display: inline-block;
   cursor: pointer;
}

#navbar {
   overflow: hidden;
   background-color: #ffffff;
}

#navbar a {
   float: left;
   display: block;
   color: #00001a;
   text-align: center;
   padding: 14px 16px;
   text-decoration: none;
   font-size: 17px;
}

#navbar a:hover {
   background-color: #c7c7c7;
   color: #00001a;
}

#navbar a.active {
   background-color: #e5e5e5;
```

```
   color: #00001a;
}

.content {
   padding: 16px;
}

.hold {
   position: fixed;
   top: 0;
   width: 100%;
}

.hold + .content {
   padding-top: 60px;
}

#navbar .log-content button {
   float: right;
   margin-top: 9px;
   margin-right: 15px;
   padding: 6px;
   background: #e5e5e5;
   font-size: 17px;
   border: 1px;
   cursor: pointer;
}
.logout-content button {
   float: right;
   margin-top: 9px;
   margin-right: 15px;
   padding: 6px;
   background: #e5e5e5;
   font-size: 17px;
   border: 1px;
   cursor: pointer;
}
#navbar .log-content {
   float: right;
}

#navbar input[type=text], input[type=password] {
   padding: 6px;
   margin-top: 8px;
   font-size: 18px;
   border: 2px;
   width: 150px;
}

#navbar .log-content button:hover {
   background: #c7c7c7;
}

#navbar .logout-content button:hover {
   background: #c7c7c7;
}

#keyInput {
   background-position: 10px 10px;
   background-repeat: no-repeat;
   width: 60%;
   font-size: 16px;
```

```
    padding: 10px 18px 10px 38px;
    border: 1px solid #e5e5e5;
    margin-bottom: 12px;
}

#cards-table {
    border-collapse: collapse;
    width: 80%;
    border: 1px solid #e5e5e5;
    font-size: 18px;
    margin-left: auto;
    margin-right: auto;
}

#cards-table th, #myTable td {
    text-align: left;
    padding: 12px;
}

#cards-table tr {
    border-bottom: 1px solid #e5e5e5;
}

#cards-table tr.header, #myTable tr:hover {
    background-color: #f1f1f1;
}

.buttonSearch {
    background-color: white;
    color: black;
    border: 2px solid #008CBA;
    padding: 8px 16px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    transition-duration: 0.4s;
    cursor: pointer;
}

.buttonSearch:hover {
    background-color: #80bfff;
    color: white;
}

.buttonClear {
    background-color: white;
    color: black;
    border: 2px solid #f44336;
    padding: 8px 16px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    transition-duration: 0.4s;
    cursor: pointer;
}

.buttonClear:hover {
    background-color: #ff8080;
```

```
   color: white;
}
```

Файл styleLogReg.css

```
body {
   margin: 0;
   font-family: Arial, Helvetica, sans-serif;
}

* {
   box-sizing: border-box;
}

.header {
   background-image: linear-gradient(to right, rgb(204, 230, 255), rgb(230, 204,
255));
   padding: 40px;
   text-align: center;
}

.basement {
   background-color: #f1f1f1;
   padding: 30px;
   text-align: center;
}

#navbar {
   overflow: hidden;
   background-color: #ffffff;
}

#navbar a {
   float: left;
   display: block;
   color: #00001a;
   text-align: center;
   padding: 14px 16px;
   text-decoration: none;
   font-size: 17px;
}

#navbar a:hover {
   background-color: #c7c7c7;
   color: #00001a;
}

#navbar a.active {
   background-color: #e5e5e5;
   color: #00001a;
}

.content {
   padding: 16px;
}

.sticky {
   position: fixed;
   top: 0;
   width: 100%;
}
```

```
.sticky + .content {
   padding-top: 60px;
}

.logout-content button {
   float: right;
   padding: 6px;
   margin-top: 8px;
   margin-right: 16px;
   background: #ddd;
   font-size: 17px;
   border: 1px;
   cursor: pointer;
}

#navbar .logout-content button:hover {
   background: #c7c7c7;
}

input[type=text], input[type=password] {
   width: 100%;
   padding: 15px;
   margin: 5px 0 22px 0;
   display: inline-block;
   border: none;
   background: #f1f1f1;
}

button {
   background-color: #80bfff;
   color: white;
   padding: 14px 20px;
   margin: 8px 0;
   border: none;
   cursor: pointer;
   width: 100%;
}

button:hover {
   opacity: 0.8;
}

.cancelbtn {
   width: auto;
   padding: 10px 18px;
   background-color: #f44336;
}

.container {
   padding: 16px;
}
```

Файл styleCard.css

```
body {
   margin: 0;
   font-family: Arial, Helvetica, sans-serif;
}

* {
```

```
   box-sizing: border-box;
}

.header {
   background-image: linear-gradient(to right, rgb(204, 230, 255), rgb(230, 204,
255));
   padding: 40px;
   text-align: center;
}

.basement {
   background-color: #f1f1f1;
   padding: 30px;
   text-align: center;
}

#navbar {
   overflow: hidden;
   background-color: #ffffff;
}

#navbar a {
   float: left;
   display: block;
   color: #00001a;
   text-align: center;
   padding: 14px 16px;
   text-decoration: none;
   font-size: 17px;
}

#navbar a:hover {
   background-color: #c7c7c7;
   color: #00001a;
}

#navbar a.active {
   background-color: #e5e5e5;
   color: #00001a;
}

.content {
   padding: 16px;
}

.sticky {
   position: fixed;
   top: 0;
   width: 100%;
}

.sticky + .content {
   padding-top: 60px;
}

.logout-content button {
   float: right;
   padding: 6px;
   margin-top: 8px;
   margin-right: 16px;
   background: #ddd;
   font-size: 17px;
```

```
   border: 1px;
   cursor: pointer;
}

#navbar .logout-content button:hover {
   background: #c7c7c7;
}


.row {
   display: -ms-flexbox;
   display: flex;
   -ms-flex-wrap: wrap;
   flex-wrap: wrap;
}

.side {
   -ms-flex: 30%;
   flex: 30%;
   background-color: #f1f1f1;
   padding: 20px;
}

.main {
   -ms-flex: 70%;
   flex: 70%;
   background-color: white;
   padding: 20px;
}
```

Файл styleCardNew.css

```
body {
   margin: 0;
   font-family: Arial, Helvetica, sans-serif;
}

* {
   box-sizing: border-box;
}

.header {
   background-image: linear-gradient(to right,    rgb(204, 230, 255), rgb(230, 204,
255));
   padding: 40px;
   text-align: center;
}

.basement {
   background-color: #f1f1f1;
   padding: 30px;
   text-align: center;
}

#navbar {
   overflow: hidden;
   background-color: #ffffff;
}

#navbar a {
```

```
   float: left;
   display: block;
   color: #00001a;
   text-align: center;
   padding: 14px 16px;
   text-decoration: none;
   font-size: 17px;
}

#navbar a:hover {
   background-color: #c7c7c7;
   color: #00001a;
}

#navbar a.active {
   background-color: #e5e5e5;
   color: #00001a;
}

.content {
   padding: 16px;
}

.sticky {
   position: fixed;
   top: 0;
   width: 100%;
}

.sticky + .content {
   padding-top: 60px;
}

.logout-content button {
   float: right;
   padding: 6px;
   margin-top: 8px;
   margin-right: 16px;
   background: #ddd;
   font-size: 17px;
   border: 1px;
   cursor: pointer;
}

#navbar .logout-content button:hover {
   background: #c7c7c7;
}

.row {
   display: -ms-flexbox;
   display: flex;
   -ms-flex-wrap: wrap;
   flex-wrap: wrap;
}

.side {
   -ms-flex: 30%;
   flex: 30%;
   background-color: #f1f1f1;
   padding: 20px;
}
```

```
.main {
   -ms-flex: 70%;
   flex: 70%;
   background-color: white;
   padding: 20px;
}


input[type=text], select {
   width: 100%;
   padding: 12px 20px;
   margin: 8px 0;
   display: inline-block;
   border: 1px solid #ccc;
   border-radius: 4px;
   box-sizing: border-box;
}

input[type=submit] {
   width: 100%;
   background-color: #4CAF50;
   color: white;
   padding: 14px 20px;
   margin: 8px 0;
   border: none;
   border-radius: 4px;
   cursor: pointer;
}

input[type=submit]:hover {
   background-color: #45a049;
}

.buttonCreate {
   background-color: white;
   color: black;
   border: 2px solid #008CBA;
   padding: 8px 16px;
   text-align: center;
   text-decoration: none;
   display: inline-block;
   font-size: 16px;
   margin: 4px 2px;
   transition-duration: 0.4s;
   cursor: pointer;
}

.buttonCreat:hover {
   background-color: #80bfff;
   color: white;
}
```

Файл 404.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org"
     xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
   <meta name="viewport" content="width=device-width,initial-scale=1">
   <title> MedView. 404 </title>
   <link   th:href="@{css/styleHome.css}"   href="/css/styleHome.css"   rel="stylesheet"
type="text/css" media="screen"/>
   <style>
       div {
           border: 1px solid gray;
           padding: 8px;
       }

       h1 {
           text-align: center;
           text-transform: uppercase;
       }
   </style>
</head>
<body>

<!--navbar-->

<div id="navbar">
   <a class="active" th:href="@{/}">Головна сторінка</a>
       <a th:href="@{/cards/new}">Нова картка</a>
       <a th:href="@{/cards/new/tacmed}">Картка пораненого ТССС</a>
   <div sec:authorize="hasRole('ADMIN')">
       <a th:href="@{/register}">Зареєструвати користувача</a>
   </div>
   <div class="log-content" sec:authorize="!isAuthenticated()">
       <form action="#"
             th:action="@{/login}"
             method="POST">
           <input id="email" type="text" placeholder="Логін" name="email">
           <input id="password" type="password" placeholder="Пароль" name="password">
           <button type="submit">УВІЙТИ</button>
       </form>
   </div>
   <div class="logout-content" sec:authorize="isAuthenticated()">
       <form action="#"
             th:action="@{/logout}"
             method="POST">
           <button type="submit">ВИХІД</button>
       </form>
   </div>
</div>

<!--head-->

<div class="header">
   <h2>MedView</h2>
   <p>Професійний погляд на ваші проблеми.</p>
   <div class="login-container" sec:authorize="isAuthenticated()">
       <h4>                              <label>Вітаємо,                              <span
sec:authentication="name">Username</span></label></h4>
   </div>
</div>

<div>
   <h1>помилка 404 :( </h1>
   <p>Йой! Виникла помилка, такої сторінки не існує.</p>
</div>

<!--basement-->
```

```
<div class="basement">
   <h2>MedView</h2>
   <p>Професійний погляд на ваші проблеми.</p>
</div>

</body>
</html>
```

Файл home.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org"
xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width,initial-scale=1">
   <title> MedView. Головна сторінка </title>
   <link   th:href="@{css/styleHome.css}"   href="/css/styleHome.css"   rel="stylesheet"
type="text/css" media="screen"/>
</head>
<body>

<!--navbar-->

<div id="navbar">
   <a class="active" th:href="@{/}">Головна сторінка</a>
   <a th:href="@{/cards/new}">Нова картка</a>
   <a th:href="@{/cards/new/tacmed}">Картка пораненого ТССС</a>
   <div sec:authorize="hasRole('ADMIN')">
   <a th:href="@{/register}">Зареєструвати користувача</a>
   </div>
<div class="log-content" sec:authorize="!isAuthenticated()">
   <form action="#"
         th:action="@{/login}"
         method="POST">
       <input id="email" type="text" placeholder="Логін" name="email">
       <input id="password" type="password" placeholder="Пароль" name="password">
       <button type="submit">УВІЙТИ</button>
   </form>
</div>
   <div class="logout-content" sec:authorize="isAuthenticated()">
       <form action="#"
             th:action="@{/logout}"
             method="POST">
          <button type="submit">ВИХІД</button>
       </form>
   </div>
</div>

<!--head-->

<div class="header">
   <h2>MedView</h2>
   <p>Професійний погляд на ваші проблеми.</p>
   <div class="login-container" sec:authorize="isAuthenticated()">
       <h4>                              <label>Вітаємо,                              <span
sec:authentication="name">Username</span></label></h4>
   </div>
```

```
</div>

<!--body-->

<div sec:authorize="!isAuthenticated()" class="not-logout-container">
   <h1>Вітаємо, щоб продовжити, будь ласка, здійсність вхід</h1>
   <a th:href="@{/login}" class="login-button">Увійти!</a>
</div>

<div sec:authorize="isAuthenticated()">
<form th:action="@{/}">
  <center>  <input  type="text"  name="keyword"  id="keyInput"  onkeyup="myFunction()"
placeholder="Введіть     ключове     слово.."     title="Введіть     ключове     слово"
th:value="${keyword}" required />

   <input class="buttonSearch" type="submit" value="Шукати" />

   <input    class="buttonClear"    type="button"    value="Очистити"    id="btnClear"
onclick="clearSearch()" /> </center>
</form>

<table class="cards-table" id="cards-table">
   <thead>
       <tr>
           <th>Дія</th>
           <th>Прізвище</th>
           <th>Ім'я</th>
           <th>Місто</th>
           <th>Електронна адреса</th>
       </tr>
   </thead>
   <tbody>
   <tr class="card" th:each="card : ${cards}">
       <td><a th:href="@{'/cards/' + ${card.id}}">Деталі...</a></td>
       <td th:text="${card.lastName}">Last Name</td>
       <td th:text="${card.firstName}">First Name</td>
       <td th:text="${card.city}">City</td>
       <td th:text="${card.email}">Email</td>
   </tr>
   </tbody>
</table>
</div>
<br>

<!--basement-->

<div class="basement">
   <h2>MedView</h2>
   <p>Професійний погляд на ваші проблеми.</p>
</div>

<!--sticky navbar script-->

<script>
   window.onscroll = function() {myFunction()};

   var navbar = document.getElementById("navbar");
   var hold = navbar.offsetTop;

   function myFunction() {
       if (window.pageYOffset >= hold) {
           navbar.classList.add("hold")
```

```
        } else {
            navbar.classList.remove("hold");
        }
    }

    function clearSearch() {
        window.location = "[[@{/}]]";
    }
</script>
</body>
</html>
```

Файл login.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org"
xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title> MedView. Увійти </title>
    <link  th:href="@{css/styleLogReg.css}"  href="/css/styleLogReg.css"  rel="stylesheet"
type="text/css" media="screen"/>
</head>
<body>
<!--navbar-->

<div id="navbar">
    <a class="active" th:href="@{/}">Головна сторінка</a>
        <a th:href="@{/cards/new}">Нова картка</a>
        <a th:href="@{/cards/new/tacmed}">Картка пораненого ТССС</a>
    <div sec:authorize="hasRole('ADMIN')">
        <a th:href="@{/register}">Зареєструвати користувача</a>
    </div>
</div>

<!--header-->

<div class="header">
    <h2>MedView</h2>
    <p>Професійний погляд на ваші проблеми.</p>
</div>

<!--body-->

<div class="container">
    <h1>Вхід на сайт</h1>
    <p>Будь ласка заповніть наступну форму</p>
    <hr>
    <form action="#"
        th:action="@{/login}"
        method="POST">

                <div>
                    <label for="email"><b>Логін</b></label>
                    <input   id="email"   type="text"   placeholder="Логін"   name="email"
required>

                    <label for="password"><b>Пароль</b></label>
```

```
                    <input     id="password"     type="password"     placeholder="Пароль"
name="password" required>

                    <button type="submit" class="buttonLogin">Увійти</button>
                </div>

                <div class="container" style="background-color:#f1f1f1">
                    <button                                                 class="cancelbtn"
onclick="window.location='http://localhost:3000/';return false;">Відмінити</button>
                </div>
            </form>
</div>

<!--basement-->

<div class="basement">
   <h2>MedView</h2>
   <p>Професійний погляд на ваші проблеми.</p>
</div>

</body>
</html>
```

Файл card.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org"
     xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width,initial-scale=1">
   <title> MedView. Деталі </title>
   <link   th:href="@{/css/styleCard.css}"   href="/css/styleCard.css"   rel="stylesheet"
type="text/css" media="screen"/>
</head>
<body>

<!--navbar-->

<div id="navbar">
   <a class="active" th:href="@{/}">Головна сторінка</a>
       <a th:href="@{/cards/new}">Нова картка</a>
       <a th:href="@{/cards/new/tacmed}">Картка пораненого ТССС</a>
   <div sec:authorize="hasRole('ADMIN')">
       <a th:href="@{/register}">Зареєструвати користувача</a>
   </div>
   <div class="logout-content" sec:authorize="isAuthenticated()">
       <form action="#"
             th:action="@{/logout}"
             method="POST">
           <button type="submit">ВИХІД</button>
       </form>
   </div>
</div>

<!--head-->

<div class="header">
   <h2>MedView</h2>
```

```
   <p>Професійний погляд на ваші проблеми.</p>
   <div class="login-container" sec:authorize="isAuthenticated()">
       <h4> <label>Hi, <span sec:authentication="name">Username</span></label></h4>
   </div>
</div>

<!--body-->

<div class="row">
   <div class="side">
       <h2>Особиста інформація</h2>
       <p>Особисті дані перевірено &#9989;</p>
       <h3 th:text="${card.email}">Email</h3>
       <h3 th:text="${card.phoneNumber}">Phone Number</h3>
       <h5   th:text="'Створено   '   +   ${card.createdAtDate}   +   '   o   '   +
${card.createdAtTime}">Created At</h5>
       <h5  th:text="'Останнє  редагування  '  +  ${card.updatedAtDate}  +  '  o  '  +
${card.updatedAtTime}">Updated At</h5>
       <h5 th:text="'Картка написана лікарем/лікаркою: ' + ${card.account.firstName} +
' '+ ${card.account.lastName}">Account Name</h5>
       <li><a th:href="@{'/cards/' + ${card.id} + '/edit'}">Редагувати данні</a></li>
       <div sec:authorize="hasRole('ROLE_ADMIN')">
       <li><a th:href="@{'/cards/' + ${card.id} + '/delete'}">Видалити данні</a></li>
       </div>
   </div>
   <div class="main">
       <h2  th:text="${card.lastName}  +  '  '  +  ${card.firstName}">Last  Name  and  First
Name</h2>
       <h2 th:text="'Стать: ' + ${card.gender}">Gender</h2>
       <h2 th:text="'Вік: ' + ${card.age}">Age</h2>
       <h2 th:text="'Місто або населений пункт: ' + ${card.city}">City</h2>
       <h2>Лист скарг</h2>
       <p th:text="${card.problemList}">Problem List</p>
       <h2>Назначенні медикаменти</h2>
       <p th:text="${card.medications}">Medications</p>
   </div>
</div>

<!--basement-->

<div class="basement">
   <h2>MedView</h2>
   <p>Професійний погляд на ваші проблеми.</p>
</div>

</body>
</html>
```

Файл card_edit.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
     xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <title> MedView. Редагувати </title>
   <link  th:href="@{/css/styleCardNew.css}"  href="/css/styleCard.css"  rel="stylesheet"
type="text/css" media="screen"/>
```

```
</head>
<body>

<!--navbar-->

<div id="navbar">
   <a class="active" th:href="@{/}">Головна сторінка</a>
       <a th:href="@{/cards/new}">Нова картка</a>
       <a th:href="@{/cards/new/tacmed}">Картка пораненого ТССС</a>
   <div sec:authorize="hasRole('ADMIN')">
       <a th:href="@{/register}">Зареєструвати користувача</a>
   </div>
   <div class="logout-content" sec:authorize="isAuthenticated()">
       <form action="#"
             th:action="@{/logout}"
             method="POST">
           <button type="submit">ВИХІД</button>
       </form>
   </div>
</div>
<!--head-->

<div class="header">
   <h2>MedView</h2>
   <p>Професійний погляд на ваші проблеми.</p>
   <div class="login-container" sec:authorize="isAuthenticated()">
       <h4>                              <label>Вітаємо,                              <span
sec:authentication="name">Username</span></label></h4>
   </div>
</div>

<!--body-->

<div class="row">
   <div class="side">
       <h2>Редагувати данні</h2>
       <h3>Будь ласка, заповність наступну форму</h3>
   </div>
   <div class="main">
       <form action="#"
             th:action="@{'/cards/{id}'(id=${card.id})}"
             th:object="${card}"
             method="POST">
           <input type="hidden" th:field="*{account}" />
           <input type="hidden" th:field="*{createdAtDate}" />
           <input type="hidden" th:field="*{createdAtTime}" />
           <h2>Прізвище</h2>
           <input         id="lName"         type="text"         th:field="*{lastName}"
placeholder="Прізвище"/>

           <h2>Ім'я</h2>
           <input id="fName" type="text" th:field="*{firstName}" placeholder="Ім'я"/>

           <h2>Вік</h2>
           <input id="age" type="text" th:field="*{age}" placeholder="Вік"/>

           <h2>Місто</h2>
           <input id="City" type="text" th:field="*{city}" placeholder="Місто"/>

           <h2>Електронна адреса</h2>
           <input id="email" type="text" th:field="*{email}" placeholder="Email"/>
```

```
            <h2>Номер телефона</h2>
            <input      id="phoneNumber"      type="text"      th:field="*{phoneNumber}"
placeholder="Номер телефона"/>

        <h2>Лист скарг</h2>
            <textarea      id="problemList"      th:field="*{problemList}"      rows="10"
cols="50"></textarea>
        <h2>Назначенні медикаменти</h2>
            <textarea      id="medications"      th:field="*{medications}"      rows="10"
cols="50"></textarea>
            <br>
            <button type="submit" class="buttonCreat">Редагувати</button>
        </form>
</div>
</div>

<!--basement-->

<div class="basement">
   <h2>MedView</h2>
   <p>Професійний погляд на ваші проблеми.</p>
</div>
</body>
</html>
```

Файл card_new.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org"
     xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width,initial-scale=1">
   <title> MedView. Створити </title>
   <link  th:href="@{/css/styleCardNew.css}"  href="/css/styleCard.css"  rel="stylesheet"
type="text/css" media="screen"/>
</head>
<body>

<!--navbar-->

<div id="navbar">
   <a th:href="@{/}">Головна сторінка</a>
       <a class="active" th:href="@{/cards/new}">Нова картка</a>
       <a th:href="@{/cards/new/tacmed}">Картка пораненого ТССС</a>
   <div sec:authorize="hasRole('ADMIN')">
       <a th:href="@{/register}">Зареєструвати користувача</a>
   </div>
   <div class="logout-content" sec:authorize="isAuthenticated()">
       <form action="#"
             th:action="@{/logout}"
             method="POST">
           <button type="submit">ВИХІД</button>
       </form>
   </div>
</div>

<!--head-->
```

```
<div class="header">
   <h2>MedView</h2>
   <p>Професійний погляд на ваші проблеми.</p>
   <div class="login-container" sec:authorize="isAuthenticated()">
       <h4>                              <label>Вітаємо,                              <span
sec:authentication="name">Username</span></label></h4>
   </div>
</div>

<!--body-->

<div class="row">
   <div class="side">
       <h2>Створення нової картки</h2>
       <h3>Будь ласка заповніть наступну форму для створення нової медичної картки</h3>
   </div>
   <div class="main">
       <form action="#"
             th:action="@{'/cards/new'}"
             th:object="${card}"
             method="POST">
           <input type="hidden" th:field="*{account}" />
           <input type="hidden" th:field="*{createdAtDate}" />
           <input type="hidden" th:field="*{createdAtTime}" />
           <h2>Прізвище</h2>
           <input id="lName" type="text" th:field="*{lastName}" placeholder="Прізвище"
required/>

           <h2>Ім'я</h2>
           <input id="fName" type="text" th:field="*{firstName}" placeholder="Ім'я"
required/>

           <h2>Вік</h2>
           <input id="age" type="text" th:field="*{age}" placeholder="Вік" required/>

           <h2>Стать</h2>
           <input id="gender" type="text" th:field="*{gender}" placeholder="Стать"
required/>

           <h2>Місто</h2>
           <input id="City" type="text" th:field="*{city}" placeholder="Місто"
required/>

           <h2>Електронна адреса</h2>
           <input id="email" type="text" th:field="*{email}" placeholder="Email"
required/>

           <h2>Номер телефона</h2>
           <input id="phoneNumber" type="text" th:field="*{phoneNumber}"
placeholder="Номер телефона" required/>

           <h2>Лист скарг</h2>
           <textarea id="problemList" th:field="*{problemList}" rows="10" cols="50"
required></textarea>
           <h2>Назначенні медикаменти</h2>
           <textarea id="medications" th:field="*{medications}" rows="10" cols="50"
required></textarea>
           <br>
           <button type="submit" class="buttonCreate">Створити</button>
       </form>
   </div>
</div>
```

```
<!--basement-->

<div class="basement">
   <h2>MedView</h2>
   <p>Професійний погляд на ваші проблеми.</p>
</div>

</body>
</html>
```

Файл card_new_tacmed.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org"
     xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
 <meta charset="utf-8">
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
 <meta name="viewport" content="width=device-width,initial-scale=1">
 <title> MedView. Створити ТССС </title>
 <link th:href="@{/css/styleCardNew.css}" href="/css/styleCardNew.css" rel="stylesheet"
type="text/css" media="screen"/>
</head>
<body>

<!--navbar-->

<div id="navbar">
 <a th:href="@{/}">Головна сторінка</a>
   <a th:href="@{/cards/new}">Нова картка</a>
   <a class="active" th:href="@{/cards/new/tacmed}">Картка пораненого ТССС</a>
 <div sec:authorize="hasRole('ADMIN')">
   <a th:href="@{/register}">Зареєструвати користувача</a>
 </div>
 <div class="logout-content" sec:authorize="isAuthenticated()">
   <form action="#"
         th:action="@{/logout}"
         method="POST">
     <button type="submit">ВИХІД</button>
   </form>
 </div>
</div>

<!--head-->

<div class="header">
 <h2>MedView</h2>
 <p>Професійний погляд на ваші проблеми.</p>
 <div class="login-container" sec:authorize="isAuthenticated()">
   <h4> <label>Вітаємо, <span sec:authentication="name">Username</span></label></h4>
 </div>
</div>

<!--body-->

<div class="row">
 <div class="side">
   <h2>Картка пораненого ТССС</h2>
   <h3>Заповніть інформацію про пораненого, запишіть усі ліки, надані постраждалому.
```

```
    Вкажіть назву, дозування, шлях та час введення анальгетику, антибіотику тощо.</h3>
 </div>
 <div class="main">
   <form action="#"
         th:action="@{'/cards/new'}"
         th:object="${card}"
         method="POST">
     <input type="hidden" th:field="*{account}" />
     <input type="hidden" th:field="*{createdAtDate}" />
     <input type="hidden" th:field="*{createdAtTime}" />
     <h2>Прізвище</h2>
     <input   id="lName"   type="text"   th:field="*{lastName}"   placeholder="Прізвище"
required/>

     <h2>Ім'я</h2>
     <input    id="fName"    type="text"    th:field="*{firstName}"    placeholder="Ім'я"
required/>

     <h2>Вік</h2>
     <input id="age" type="text" th:field="*{age}" placeholder="Вік" required/>

     <h2>Стать</h2>
     <input id="gender" type="text" th:field="*{gender}" placeholder="Стать" required/>

     <h2>Ідентифікаційний номер</h2>
     <input         id="idNumber"         type="text"         th:field="*{privateNumber}"
placeholder="Ідентифікаційний номер" required/>

     <h2>Місто</h2>
     <input id="City" type="text" th:field="*{city}" placeholder="Місто" required/>

     <h2>Електронна адреса</h2>
     <input id="email" type="text" th:field="*{email}" placeholder="Email" required/>

     <h2>Номер телефона</h2>
     <input  id="phoneNumber"  type="text"  th:field="*{phoneNumber}"  placeholder="Номер
телефона" required/>

     <h2>Механізм травми</h2>
     <input        id="kindOfInjury"        type="text"        th:field="*{kindOfInjury}"
placeholder="Механізм травми" required/>
     <h2>Лист скарг</h2>
     <textarea    id="problemList"    th:field="*{problemList}"    rows="10"    cols="50"
required></textarea>
     <h2>Назначенні медикаменти</h2>
     <textarea    id="medications"    th:field="*{medications}"    rows="10"    cols="50"
required></textarea>
     <br>
     <button type="submit" class="buttonCreat">Створити</button>
   </form>
 </div>
</div>

<!--basement-->

<div class="basement">
 <h2>MedView</h2>
 <p>Професійний погляд на ваші проблеми.</p>
</div>

</body>
</html>
```

Файл register.html

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org"
     xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <title> MedView. Реєстрація </title>
   <link  th:href="@{css/styleLogReg.css}"  href="/css/styleLogReg.css"  rel="stylesheet"
type="text/css" media="screen" />
</head>
<body>

<!--navbar-->

<div id="navbar">
   <a th:href="@{/}">Головна сторінка</a>
       <a th:href="@{/cards/new}">Нова картка</a>
       <a th:href="@{/cards/new/tacmed}">Картка пораненого TCCC</a>
   <div sec:authorize="hasRole('ADMIN')">
       <a class="active" th:href="@{/register}">Зареєструвати користувача</a>
   </div>
</div>

<!--head-->

<div class="header">
   <h2>MedView</h2>
   <p>Професійний погляд на ваші проблеми.</p>
   <div class="login-container" sec:authorize="isAuthenticated()">
       <h4>                              <label>Вітаємо,                              <span
sec:authentication="name">Username</span></label></h4>
   </div>
</div>

<!--body-->

<div class="container">
<form action="#"
     th:action="@{/register}"
     th:object="${account}"
     method="POST">
   <div class="container">
       <h1>Реєстрація</h1>
       <p>Будь ласка заповніть наступну форму</p>
       <hr>
        <label for="firstname">Ім'я</label>
        <input  id="firstname"  type="text"  th:field="*{firstName}"  placeholder="Ім'я"
required/>

       <label for="lastname">Прізвище</label>
       <input  id="lastname"  type="text"  th:field="*{lastName}"  placeholder="Прізвище"
required/>

       <label for="email">Логін</label>
       <input id="email" type="text" th:field="*{email}" placeholder="Email" required/>

       <label for="password">Пароль</label>
```

```
       <input id="password" type="password" th:field="*{password}" placeholder="Пароль"
required/>

       <hr>
       <button type="submit" class="buttonRgstr">Зареєструвати</button>
       <div class="container" style="background-color:#f1f1f1">
           <button                                                    class="cancelbtn"
onclick="window.location='http://localhost:3000/';return false;">Відмінити</button>
           <span     class="low">Вже     є     акаунт?     <a     th:href="@{/login}"
href="login">Вхід</a>.</span>
       </div>
   </div>
</form>
</div>

<!--basement-->

<div class="basement">
   <h2>MedView</h2>
   <p>Професійний погляд на ваші проблеми.</p>
</div>

</body>
</html>
```

Додаток Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	