МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Web-додаток підтримки роботи медичної сестри відділення фізичної реабілітації та медицини»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студентка групи ІТ-71 Токар Анастасія Сергіївна

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____«___» _____2021 р.

Науковий керівник _____ к.т.н., доц., Ващенко С. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ Шифрін Д. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2021

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«_____»_____ 2021 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Токар Анастасія Сергіївна

1 Тема роботи *Web-додаток підтримки роботи медичної сестри відділення фізичної реабілітації та медицини*

керівник роботи *Ващенко Світлана Михайлівна, к.т.н., доц.*,

затверджені наказом по університету від «*14*» *квітня* 2021 р. *№0181-VI*

2 Строк подання студентом роботи «*7*» *червня* 2021 р.

3 Вхідні дані до роботи *результати співбесіди з замовником, висунуті ним вимоги, встановлені форми звітної документації*

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) *Вступ. Аналіз предметної області: огляд проблеми та постановка задачі. Проєктування web-додатку: створення структурно-функціональної моделі основного процесу; розробка UML-моделі майбутнього додатку; модель бази даних. Практична реалізація додатку: розробка бази даних, реалізація програмних модулів, використання розробленого додатку. Висновки.*

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Актуальність роботи. Постановка задачі. Аналіз предметної області. Вимоги до web-додатку. Структурно-функціональна модель. Діаграма варіантів використання. Схема бази даних. Засоби реалізації . Демонстрація проекту. Висновки.

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання_____01.10.2020_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання	01.02.21 – 01.02.21	
2	Планування робіт	03.02.21 – 15.02.21	
3	Проектування інтерфейсу користувача	23.02.21 – 01.03 21	
4	Формування ТЗ на розробку ПЗ	19.02.21 – 23.02.21	
5	Технічний проект ПЗ	15.02.21 – 19.02.21	
6	Інструкції адміністратора та користувача	23.04.21 – 28.04.21	
7	Спринт 1 – реалізація критичних функцій	22.02.21 – 22.03.21	
8	Спринт 2 – реалізація додаткових функцій	25.03.21 – 23.04.21	
9	Написання тестових прикладів	23.02.21 – 23.03.21	
10	Модульне тестування	08.04.21 – 23.04.21	
11	Системне тестування	23.04.21 -.08.05.21	
12	Бета-тестування	10.05.21 – 31.05.21	
13	Перевірка документації	31.05.21 – 04.06.21	

Студент _____ Токар А.С.
(підпис)

Керівник роботи _____ к.т.н., доц. Ващенко С.М.
(підпис)

РЕФЕРАТ

Тема кваліфікаційної роботи бакалавра «Web-додаток підтримки роботи медичної сестри відділення фізичної реабілітації та медицини»

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 17 найменувань, 3 додатків. Загальний обсяг роботи − 112 сторінок, у тому числі 53 сторінки основного тексту, 2 сторінки списку використаних джерел, 59 сторінок додатків.

Кваліфікаційну роботу бакалавра присвячено розробці web-додатку, який би забезпечив підтримку роботи медичної сестри відділення фізичної реабілітації та медицини.

В роботі проведено аналіз рівня впровадження IT-технологій в медицині, виявлено проблему відсутності відповідних засобів, спрямованих забезпечення роботи медичних сестер фізіотерапевтичних відділів. За результатами аналізу предметної області складено технічне завдання на розробку.

У роботі виконано структурно-функціональне моделювання основного процесу забезпечення роботи сестер відділення по обробці щоденної інформації про пацієнтів. Роботу майбутнього програмного додатку описано у відповідності до вимог стандарту UML. Для збереження інформації спроектовано базу даних.

Результатом проведеної роботи є програмне забезпечення, реалізоване у вигляді відповідного web-додатку.

Практичне значення роботи полягає у тому, що розроблене програмне забезпечення спрощує та автоматизує роботу медичної сестри по обробці з внутрішньою інформацією при виконанні своїх посадових обов’язків щодо обробки інформації про виконані маніпуляції з пацієнтами відділення.

Ключові слова: WEB-ДОДАТОК, МЕДИЧНА СЕСТРА, БАЗА ДАНИХ, HTML, CSS, PHP, NODE.JS, САЙТ.

ЗМІСТ

ВСТУП

В будь-якій сфері діяльності людини в нинішній час відбувається впровадження інформаційних технологій (ІТ-технологій). В першу чергу це стосується виконання дій, спрямованих на обробку інформації. В медичній галузі також спостерігається значне зростання рівня використання інформаційних технологій.

Наразі ІТ-технології широко використовують при створенні систем діагностики, забезпеченні комунікації між ланками лікар-лікар, пацієнт-лікар. Останнім часом набувають широкого розповсюдження мобільні додатки, наприклад, довідники [1]. Також активно використовуються віртуальні програми-тренажери при навчанні студентів.

Проте наразі актуальним залишається питання підтримки діяльності медичних сестер відділень фізичної реабілітації, які окрім своїх обов’язків медичного характеру, мають ще й виконувати збір та обробку інформації про проведені маніпуляції з пацієнтами.

Тому темою даної роботи є розробка web-додатку підтримки роботи медичної сестри відділення фізичної реабілітації та медицини, який би дозволив спростити та автоматизувати роботу медичної сестри з внутрішньою інформацією при виконанні посадових інструкцій.

Для досягнення поставленої мети потрібно вирішити такі задачі.

1. Дослідити предметну область щодо використання ІТ-технологій та визначити ті види робіт, які можуть бути автоматизовані.
2. Скласти детальне технічне завдання на виконання робіт.
3. Виконати моделювання майбутнього програмного додатку.
4. Змоделювати та реалізувати базу даних для збереження потрібної інформації.
5. Виконати програмну реалізацію модулів згідно створеної моделі та технічного завдання.

Результати роботи доповідалися на міжнародній конференції «Інформатика. Математика. Автоматизація -2021» [2].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд проблеми

У зв’язку зі стрімким розповсюдженням інформаційних технологій та високою автоматизацією більшості систем управління, виникає необхідність впроваджувати передові технології в кожну сферу життя та безпосередньо полегшувати щоденну роботу робітника, студента, покупця, клієнта, школяра, та ін. З кожним роком все більше і більше комп’ютеризація, автоматизація, програмна підтримка охоплює нові аспекти людської діяльності та стало проникає в розуміння будь-якого робочого процесу [3 - 5].

Важко уявити сферу діяльності, де б ще й досі не використовували засоби та інструменти інформаційних технологій. Розглядаючи більш детально таку важливу сферу життя як медицину, можна зробити неоднозначні висновки.

В даний час в Україні в медичних закладах впроваджено інформаційну систему HELSI.ME. Користувачами системи можуть бути як пацієнти, так і весь персонал медичного закладу. Тому функціональні можливості системи досить широкі: від елементарної можливості запису пацієнта до конкретного лікаря (рис. 1.1), до комплексної автоматизації діяльності лікувального закладу [6].

В першу чергу автоматизація стосується роботи лікаря по веденню прийому пацієнтів та їх історій хвороби. Серед інших сервісів передбачено й можливість організації проведення відеоконсультацій. Для зручності пацієнтів розроблено мобільний додаток (рис. 1.2) [7].

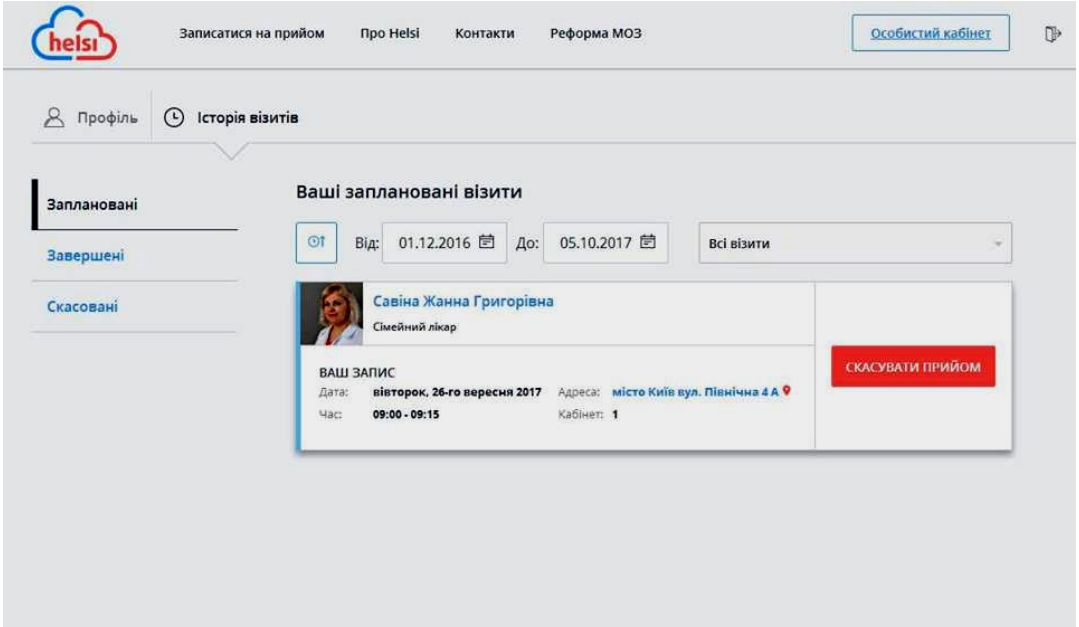


Рисунок 1.1 – Сервіс запису до лікаря пацієнтом

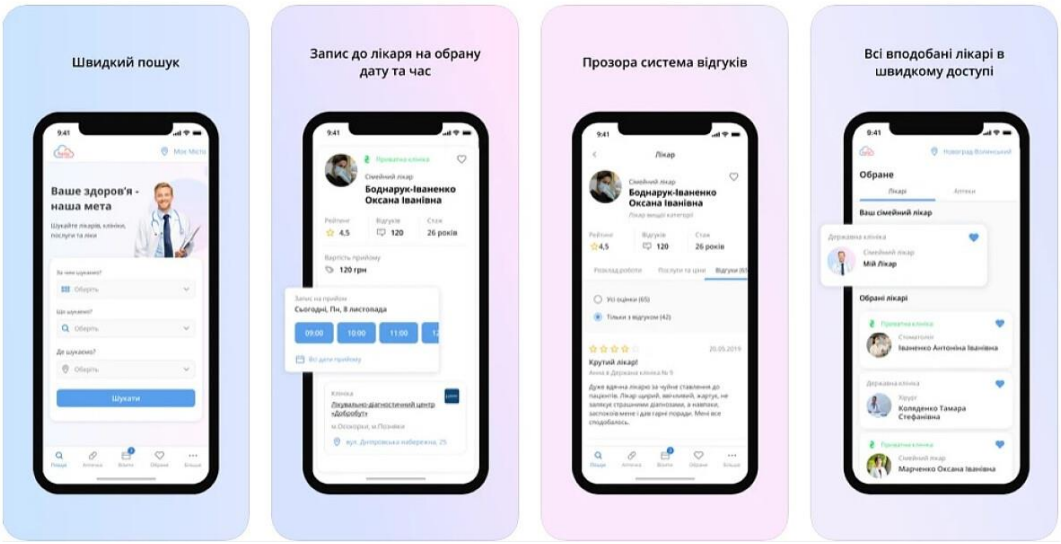


Рисунок 1.2 – Мобільний додаток для пацієнта

Не звертаючи увагу на те, що більшість трудових ресурсів при розробці цієї системи було вкладено в осучаснення ведення зв’язку між пацієнтом та записом на прийом до лікаря, сфера діяльності медичного персоналу у веденні внутрішніх облікових записів та обробки даних наразі ще залишилася без уваги. Якщо розглядати систему з точки зору виконання медичним сестрами своїх обов’язків, то функціонал не може в повній мірі охопити все різноманіття процедур, які можуть виконуватися. Тому при виконанні своїх посадових обов’язків медичні сестри, особливо в

фізіотерапевтичних відділеннях чи кабінетах, і досі опрацьовують значні обсяги поточної інформації та створюють звітну документацію в ручному режимі. В кращому випадку ступінь автоматизації цих процесів визначається використанням текстового редактора.

З огляду на це, можна зробити висновок, що на сьогодні не існує програмних продуктів аналогів, котрі б дійсно змогли автоматизувати обробку внутрішніх даних.

Наразі у відкритих джерелах не наведено інформації про існуючі інформаційної системи, що б дозволила відмовитися від застарілих, малоефективних методів обробки інформації медичними сестрами у відділеннях фізичної терапії. Досі співробітники фізичної реабілітації та медицини у реалізації задач, що вимагаються посадою, застосовують ручні записи, або ж ступінь автоматизації процесів завершується на використанні часу від часу текстового редактора.

Проаналізувавши детально об’єми робіт та щоденні задачі співробітника у повному його обсязі, було прийнято рішення розробити відповідний web-додаток підтримки діяльності сестри медичної фізичної реабілітації та медицини, котрий своїм функціоналом суттєво змінить ступінь навантаження співробітника по обробці щоденної інформації про пацієнтів, з якими проводилися маніпуляції.

1.2 Постановка задачі

Метою проєкту є розробка інформаційної системи – web-додатку підтримки сестри медичної фізичної реабілітації та медицини. Розроблювана система має максимально автоматизувати існуючі способи обліку та ведення внутрішніх задач співробітником у відділенні фізичної реабілітації та медицини.

Розроблюваний веб-додаток підтримки сестри медичної відділення фізичної реабілітації та медицини (ФРМ) має реалізовувати наступні функціональні вимоги:

- мають бути передбачені профілі для користувачів: сестра медична лазерної терапії, масажист, сестра медична процедур парафіном, старша сестра медична;

- для користувача сестра медична лазерної терапії реалізувати наступний функціонал: можливість зберігати інформацію за кожним пацієнтом (час лікування, кількість полів лікування, місце проживання, від якого відділення направлений); формування денного звіту (сумарна кількість пацієнтів за кожним районом, сумарна кількість пролікованих одиниць за кожним районом, сумарна кількість пацієнтів за кожним відділенням, сумарна кількість пролікованих одиниць за кожним відділенням);
- для користувача масажист реалізувати наступний функціонал: можливість зберігати інформацію за кожним пацієнтом (час проведення сеансу, область лікування тіла, місце проживання, від якого відділення направлений); формування денного звіту (сумарна кількість пацієнтів за кожним районом, сумарна кількість пацієнтів за кожним відділенням);
- для користувача сестра медична процедур з парафіном реалізувати наступний функціонал: можливість зберігати інформацію за кожним пацієнтом (кількість полів лікування, місце проживання, від якого відділення направлений); формування денного звіту (сумарна кількість пацієнтів за кожним районом, сумарна кількість пролікованих полів за кожним районом, сумарна кількість пацієнтів за кожним відділенням, сумарна кількість пролікованих полів за кожним відділенням);
- для користувача старша сестра медична реалізувати наступний функціонал: щомісячно формувати звіт із даними лазерної терапії (сумарна кількість пацієнтів за кожним районом, сумарна кількість пролікованих одиниць за кожним районом, сумарна кількість пацієнтів за кожним відділенням, сумарна кількість пролікованих одиниць за кожним відділенням); щомісячно формувати звіт із даними лікування парафіном (сумарна кількість пацієнтів за кожним районом, сумарна кількість пролікованих полів за кожним районом, сумарна кількість пацієнтів за кожним відділенням, сумарна кількість пролікованих полів за кожним відділенням); щомісячно формувати звіт із даними масажу (сумарна кількість пацієнтів за кожним районом, сумарна кількість часу лікування пацієнтів за кожним районом, сумарна кількість пацієнтів за кожним

відділенням, сумарна кількість часу лікування пацієнтів за кожним відділенням).

Технічне завдання на виконання вказаної розробки представлено у додатку А.

Проведено планування виконуваних робіт, результати якого наведено у додатку Б.

2 ПРОЄКТУВАННЯ WEB-ДОДАТКУ

2.1 Структурно-функціональне моделювання

Для побудови структурно-функціональної моделі було обрано нотацію стандарту IDEF0, яка дозволяє у графічному форматі точно представити всі процеси, які відбуваються в системі та відобразити ті потоки інформації, що при цьому циркулюють [8]. Перевагою використання саме цього стандарту є те, що всі об’єкти та діаграми вибудовуються в чітку ієрархію, яка точно та прозоро відображає відносини між окремими етапами роботи, а не їх виконанням у часі [9].

Для того, щоб ефективно виконати розробку web-додатку, треба чітко представляти процес його функціонування в майбутньому. Тому основним процесом для дослідження обрано «Інформаційна підтримка роботи сестри медичної фізичної реабілітації та медицини». Модель будується з точки зору медичного персоналу відділення (кабінету) фізіотерапії у форматі «to-be».

Згідно стандарту на початковому етапі виконано узагальнений опис процесу у вигляді контекстної діаграми (рис.2.1).

Вхідною інформацією є дані про пацієнта, який звернувся на лікування, а також дані користувача, щоб в подальшому можна було визначити рівень доступу працівника до функцій системи.

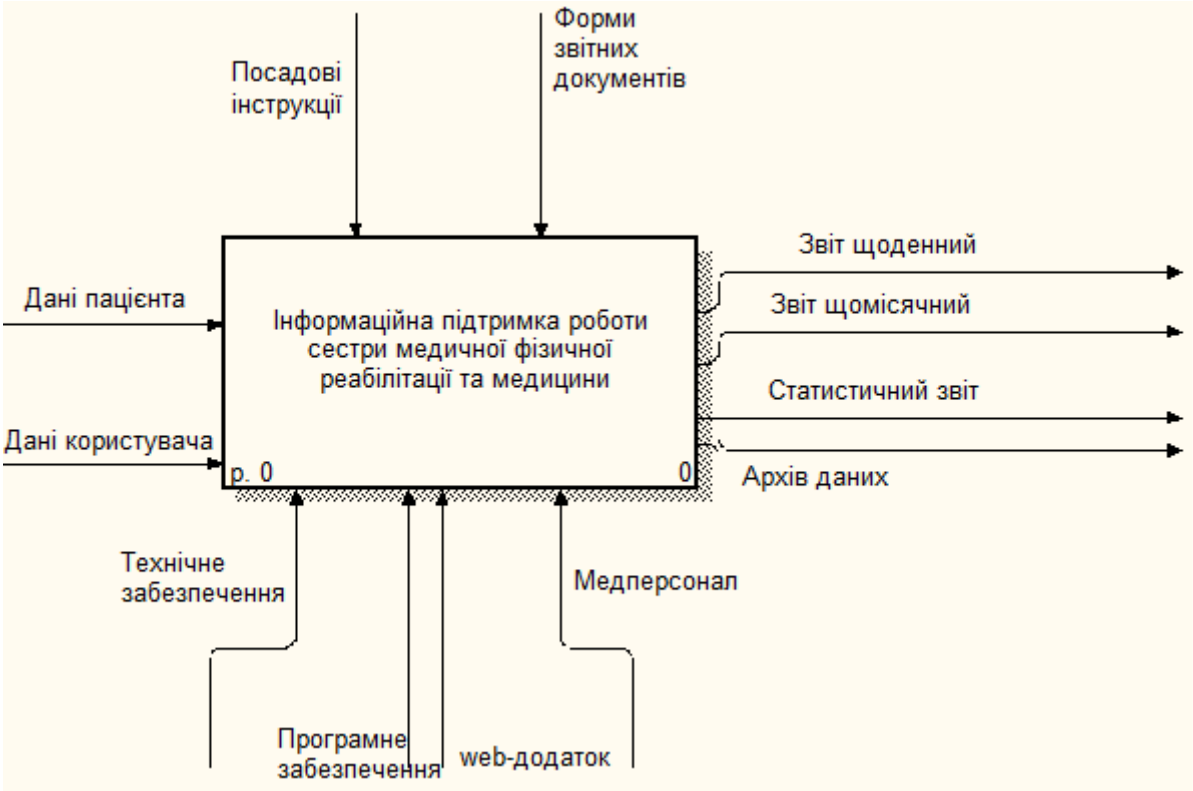


Рисунок 2.1 – Контекстна діаграма основного процесу

Механізми реалізації основного процесу – технічне та програмне забезпечення, розроблений Web-додаток та медичний персонал, який і буде виконувати всі необхідні функції.

Контролюючим фактором є посадові інструкції, згідно яких виконується робота медсестри з пацієнтом, та встановлені форми звітної документації, у відповідності до яких будуть формуватися документи звіту.

Результатом виконання основного процесу є звіти, які формує медперсонал за різні періоди, та архівна інформація, збережена у базі даних.

Далі виконано декомпозицію основного процесу. Отримана діаграма (рис. 2.2) більш детально описує його.

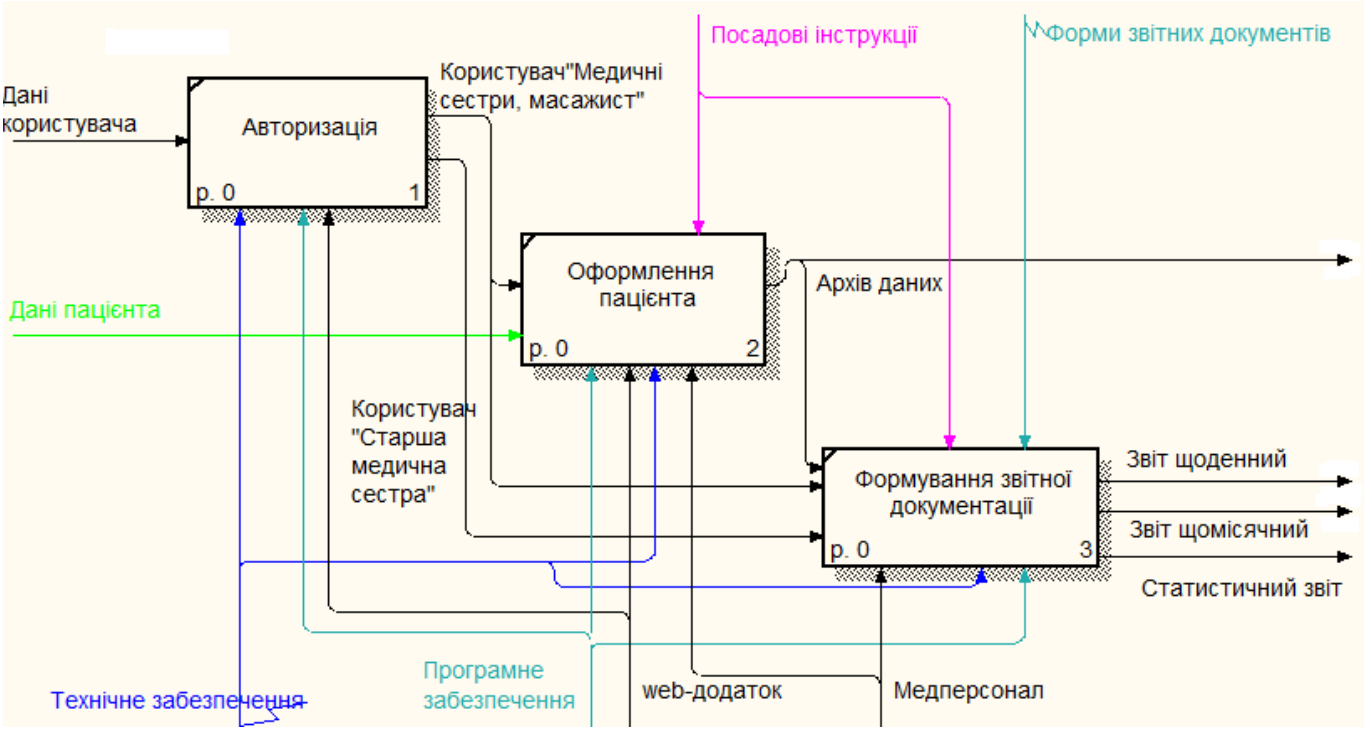


Рисунок 2.2 – Діаграма декомпозиції 1 рівня

2.2 UML - модель

За допомогою уніфікованої моделі моделювання UML існує можливість відобразити процес роботи будь-якої інформаційної системи за допомогою функціональних блоків. Дана модель широко використовується у візуалізації процесів, конструюванні та документуванні програмних систем [10, 11].

UML діаграма варіантів використання розроблюваної системи наведено на рисунку 2.3.

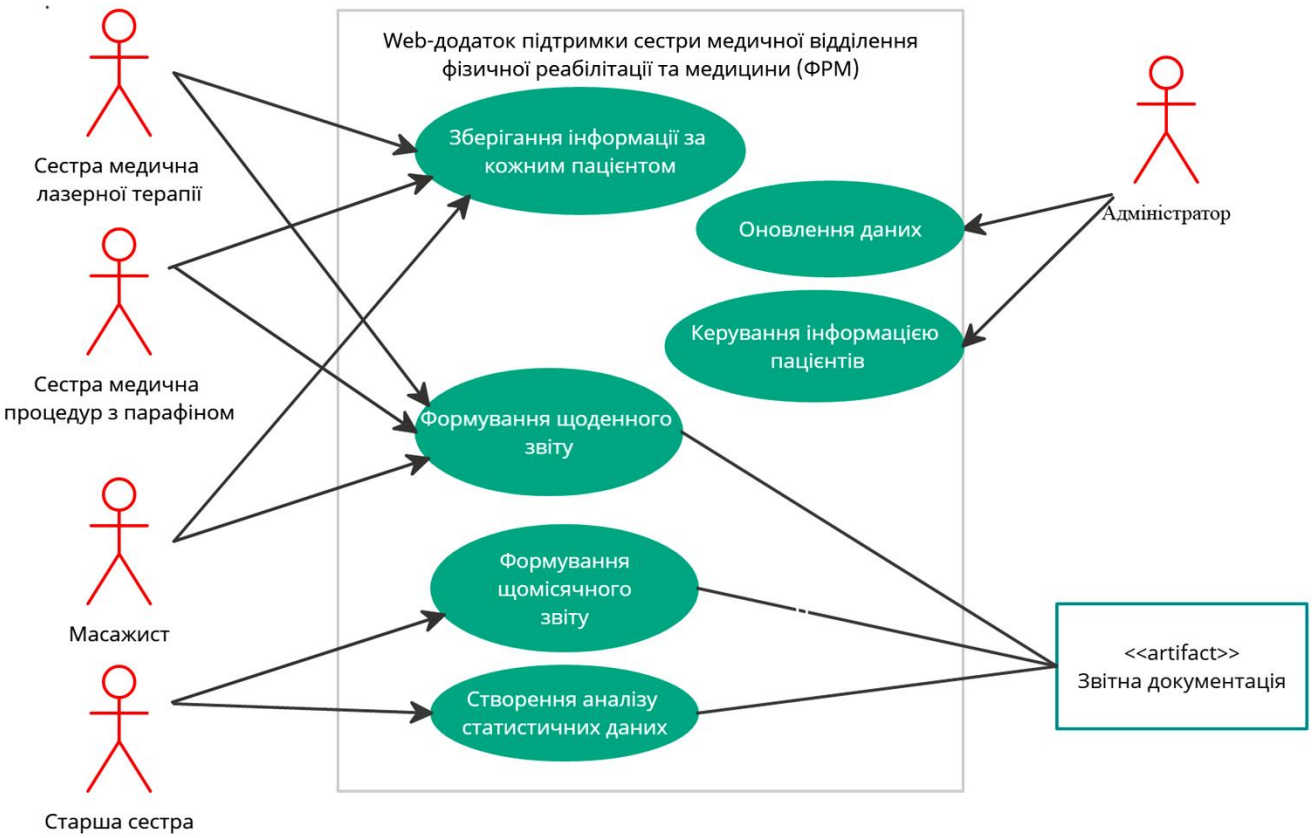


Рисунок 2.3 – Діаграма варіантів використання

Модель аналізу розроблюваного web-додатку підтримки сестри медичної фізичної реабілітації та медицини проілюстровано за допомогою діаграми класів аналізу інформаційної системи, що наведено на рисунку 2.4.

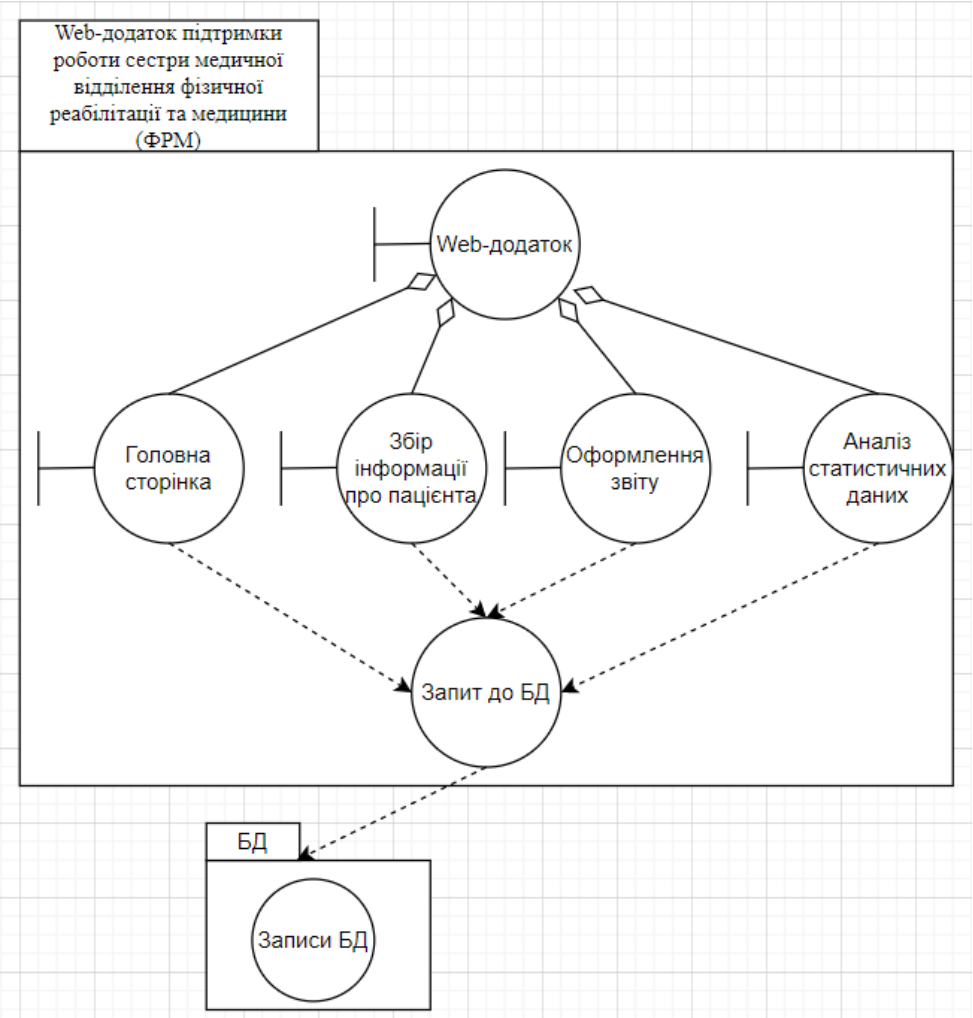


Рисунок 2.4 – Діаграма класів аналізу інформаційної системи

Діаграма комунікації відображає зв'язок між елементами під час їх взаємодії майже таким же способом, як діаграма послідовності. Однак якщо в діаграмі послідовності основна увага приділяється взаємодії об'єктів в часі, то діаграма комунікації відображає загальну взаємодію об'єктів моделі, де замість тимчасового показника використовуються впорядковані номери викликів [12].

Діаграма комунікацій web-додатку наведена на рисунку 2.5.

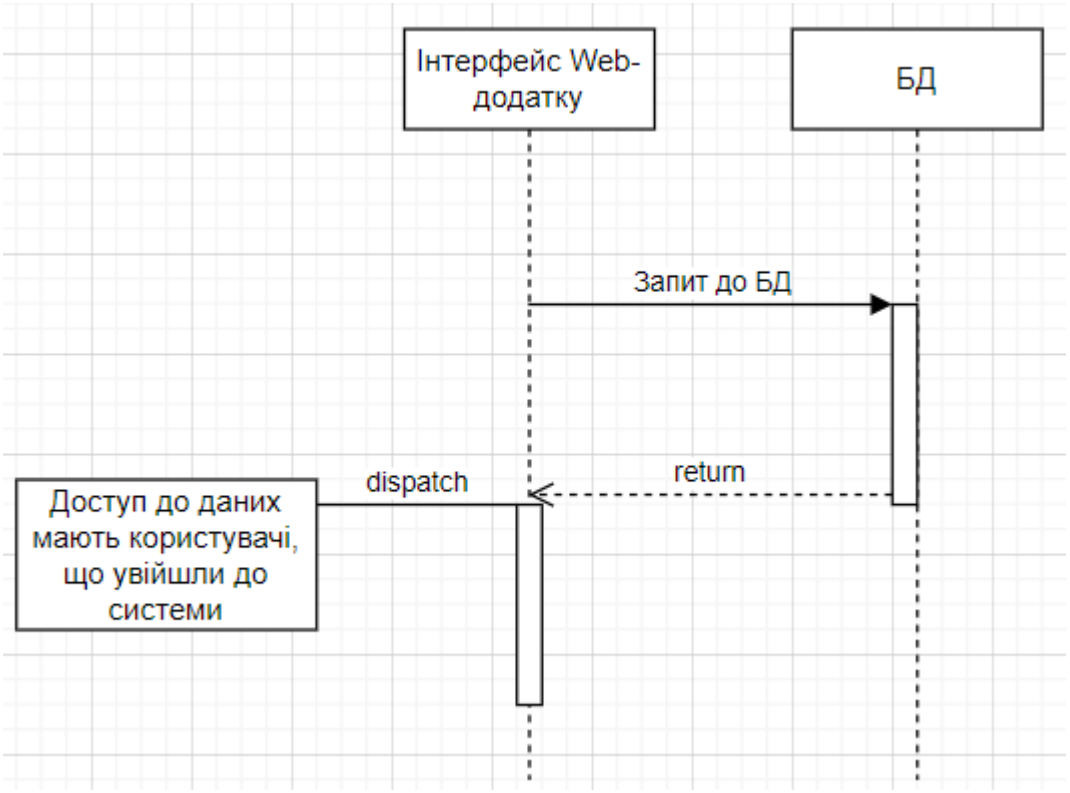


Рисунок 2.5 – Діаграма комунікацій web-додатку

2.3 Схема бази даних

Як вказано в структурно-функціональній моделі, в системі обробляються великі обсяги інформації про пацієнтів. Також потрібно організувати збереження цієї інформації в додатку. Тому для виконання цих задач потрібно передбачити в структурі додатку відповідну базу даних.

За результати аналізу предметної області виділено всі характеристик, які необхідно зберігати в базі. За результатами моделювання [17] складено концептуальну модель бази даних, котра наведена на рисунку 2.6.

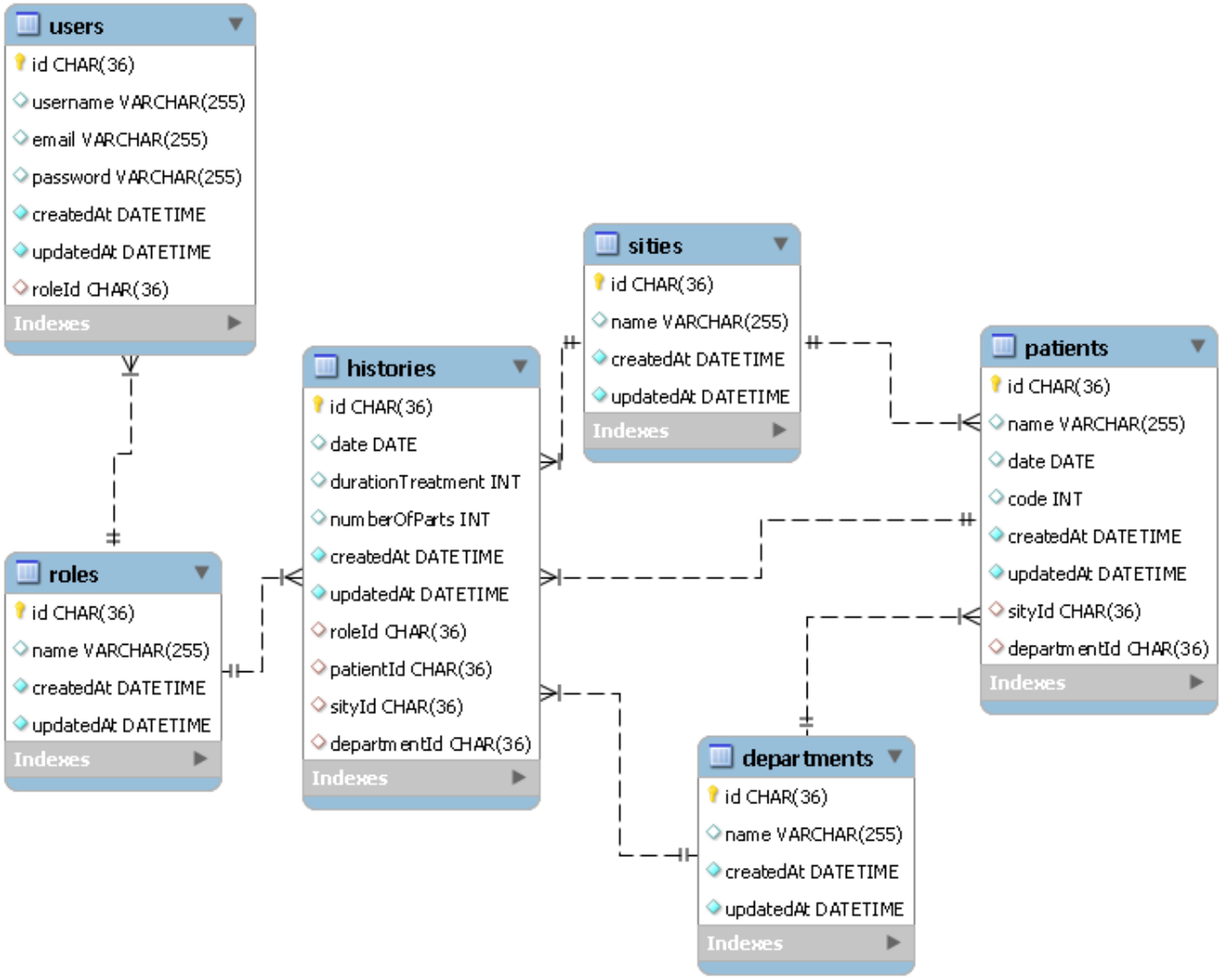


Рисунок 2.6 – Схема бази даних

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ДОДАТКУ

3.1 Архітектура програмного додатку

Архітектура web–додатку, що розробляється, побудована на співвідношенні клієнт-сервер та наведена на рисунку 3.1.

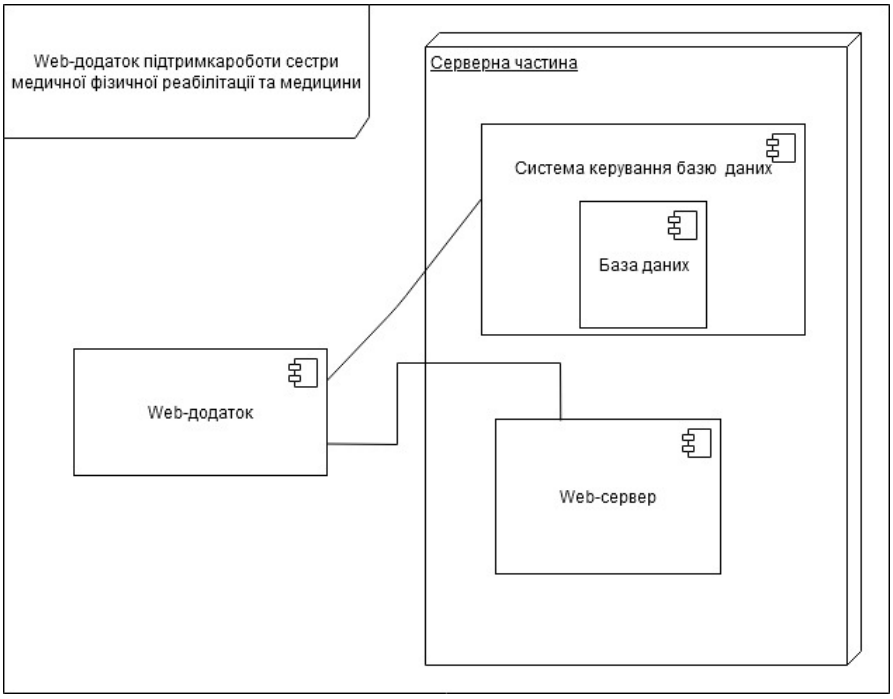


Рисунок 3.1 – Схема роботи додатку

У таблиці 3.1 описано логічне та функціональне навантаження кожного із необхідних файлів для роботи розроблюваного web-додатку [16].

Таблиця 3.1 – Структура програмного додатку

№ з/п	Назва файлу	Розташування	Опис
1	config.js	config	Зберігає усі необхідні дані для підключення до бази даних.

Продовження табл. 3.1

2	admin.controller.js	controllers	Виконує усі дії з базою даних, що належать до адміністратора.
3	auth.controller.js		Файл для авторизації та реєстрації користувача. Безпосередньо під час авторизації користувача створюється токен.
4	sister.controller.js		Зберігає усі необхідні методи для взаємодії із розробленою базою даних, що належать до сестер та масажиста.
5	dataBase.js	middleware	Виконує унікальні запити до бази даних, використовуючи декілька таблиць.
6	index.js		Об’єднання усіх файлів в папці middleware.
7	verifySignUp.js		Виконує перевірку на існуючі поштові адреси і ролі під час реєстрації користувача.
8	department.model.js	models	Шаблон таблиці department.
9	history.model.js		Шаблон таблиці history.
10	index.js		За допомогою бібліотеки sequelize встановлює підключення до бази даних, і використовуючи шаблони створює зв’язки бази даних.

Продовження табл. 3.1

11	patient.model.js		Шаблон таблиці patient.
12	role.model.js		Шаблон таблиці role.
13	sity.model.js		Шаблон таблиці sity.
14	user.model.js		Шаблон таблиці user.
15	auth.routes.js	routers	Запити на реєстрацію або авторизацію.
16	sister.routes.js		Усі запити користувачів – сестер.
17	admin.routes.js		Усі запити адміністратора.
18	admin\css	views	Зберігає стиль для сторінки адміністратора.
19	css		Стилі всіх користувачів, окрім адміністратора.
20	img		Фонове зображення сторінки авторизації.
21	js		Зберігає усі необхідні бібліотеки для використання на стороні клієнта.
22	layoutAdmin.hbs	layouts	Початковий шаблон сторінки адміністратора.
23	layoutLogin.hbs		Початковий шаблон сторінки авторизації.
24	layoutUser.hbs		Початковий шаблон сторінки користувача (сестер).
25	menuAdmin.hbs	partials	Зберігає опис меню адміністратора.
26	menuUser.hbs		Зберігає опис меню адміністратора.

Продовження табл. 3.1

27	createPatient.hbs		Зберігає шаблон тіла сторінки створення пацієнта.
28	createProcedure.hbs		Зберігає шаблон тіла сторінки створення процедури.
29	createUser.hbs		Зберігає шаблон тіла сторінки створення користувача.
30	help.hbs		Зберігає шаблон тіла сторінки створення повідомлення користувачів до адміністратора.
31	history.hbs		Зберігає шаблон тіла сторінки звіту.
32	singnin.hbs		Зберігає шаблон тіла сторінки авторизації.
33	technicaSupport.hbs		Зберігає шаблон тіла сторінки списку повідомлень у адміністратора.
34	app.js	views	Існує для налаштування і підключення усіх необхідних файлів.

Взаємозв’язки файлів створеного програмного додатку у середовищі розробки відображено на рисунку 3.2.

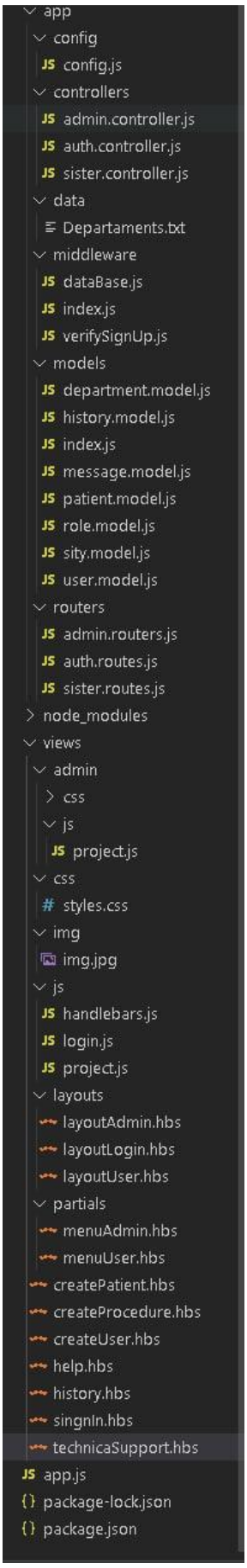


Рисунок 3.2 – Структура програмного додатку в середовищі розробки

3.2 Програмна реалізація

Оскільки основне призначення web-додатку – обробка інформації, то на першому етапі практичного виконання розробки було створено базу даних.

Системою керування базами даних, яка буде виконувати менеджмент роботи з відповідними таблицями, обрано MySQL Server. Розробка таблиць фізичної реалізації бази даних виконувалася у відповідності до розробленої моделі (рис. 2.6) з використанням програмного засобу MySQL Workbench. У додатку адміністрування бази даних було створено безпосередньо базу даних та додано відповідні таблиці.

Перелік створених таблиць наведено на рис. 3.3.

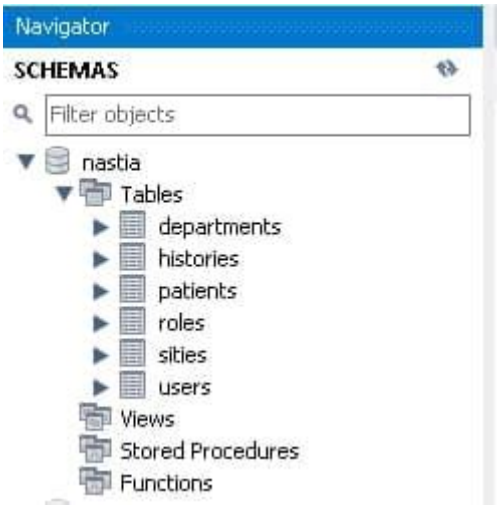


Рисунок 3.3 – Структура створеної бази даних у середовищі Workbench

Далі для кожної таблиці бази даних створювався набір полів, відповідно до моделі. При цьому вказувалися типи даних, які будуть в ньому зберігатися, та налаштовувалися відповідні властивості. На рис. 3.4 для прикладу наведено однієї структуру створеної таблиці patients.

Деякі довідкові таблиці (наприклад, з переліком існуючих у лікарні відділень), були заповнені безпосередньо засобами Workbench. Перевірка результату заповнення була проведена за рахунок виконання запиту на відображення всієї інформації таблиці (рис.3.5).

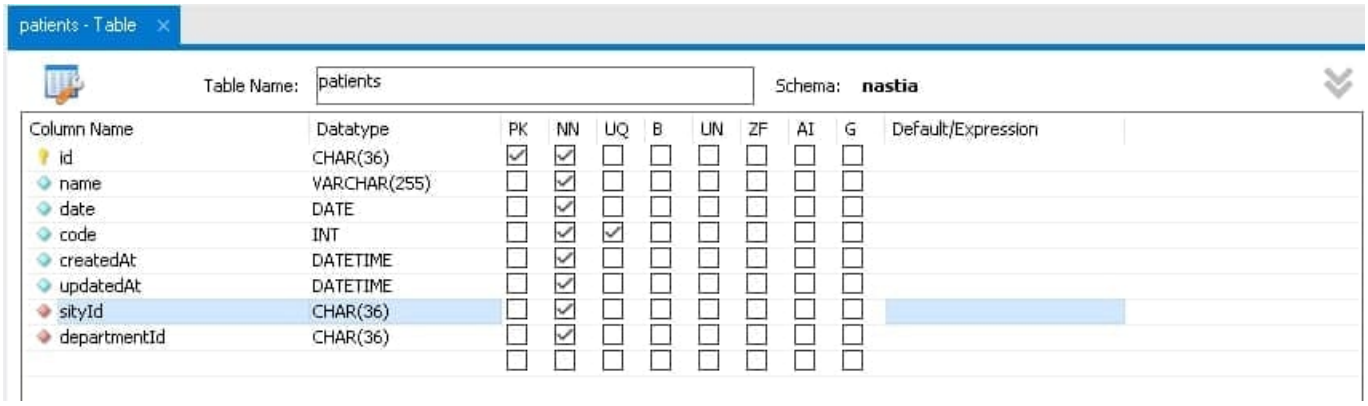


Рисунок 3.4 – Структура таблиці patients у Workbench

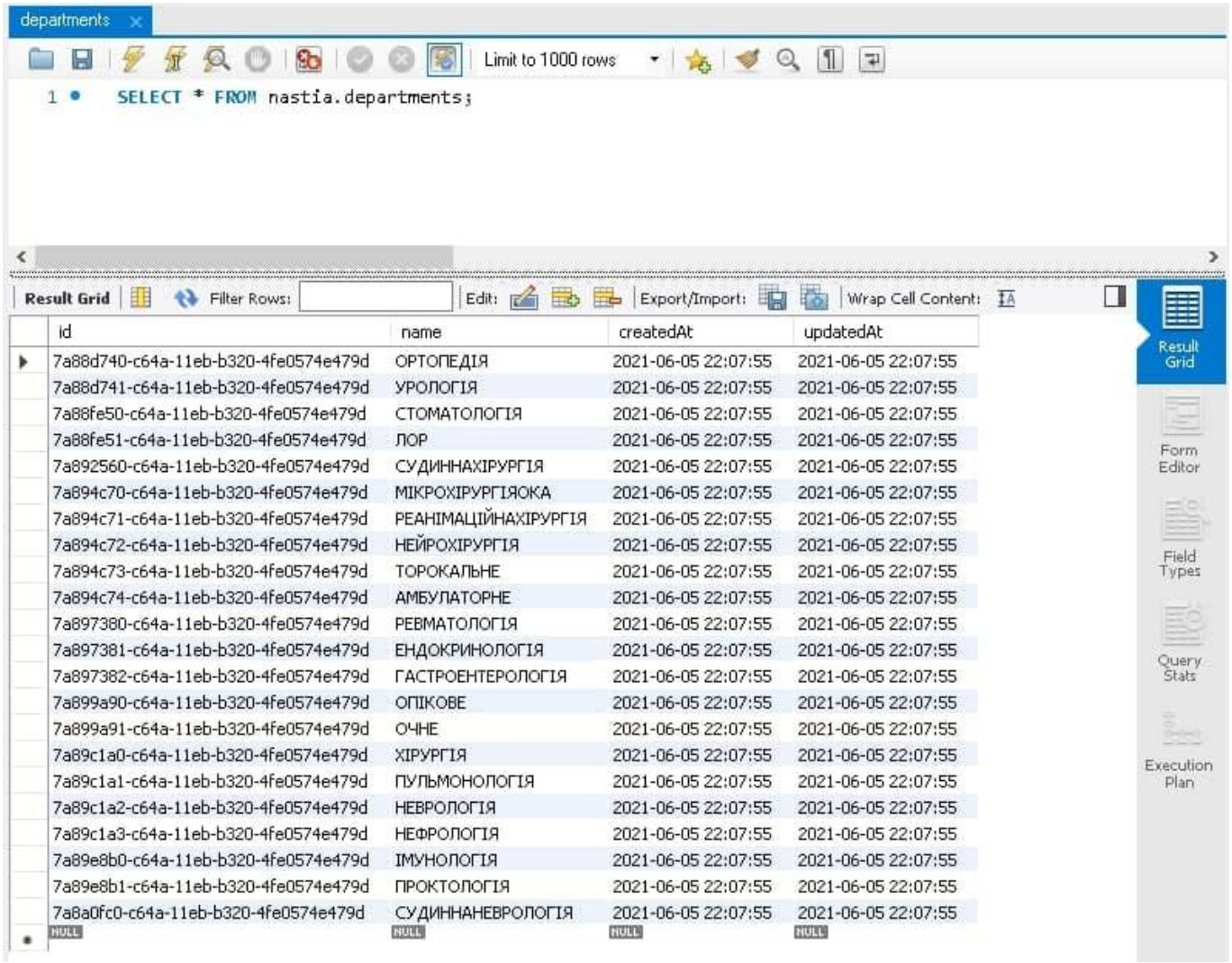


Рисунок 3.4 – Результат заповнення таблиці у Workbench

Наступний етап реалізації практичної частини дипломного проекту - написання програмного коду. Для його виконання було використано такі програмні продукти: Visual Studio Code – редактор вихідного коду, розроблений Microsoft для Windows,

Node.js – програмна платформа, котра здійснює трансляцію JavaScript у машинний код.

Також для Node.js було встановлено наступні бібліотеки:

- express – мінімалістичний і гнучкий веб-фреймворк для додатків Node.js, що надає великий набір функцій для реалізації веб-додатків [13];
- Jsonwebtoken – використовується для створення точену;
- sequelize – використовується для підключення до бази даних MySQL;
- hbs – це розширення за замовчуванням для представлень, які обробляються засобами Handlebars – це шаблонизатор для JavaScript, який допомагає знизити складність створення таких [14-15].

Так як вирішено використовувати бібліотеку Handlebars, то маємо можливість розділити сторінку на макети. У проєкті макети зберігаються в папці layouts (рис.3.5).

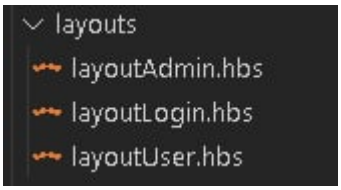


Рисунок 3.5 – Папка зберігання макетів проєкта

Також маємо можливість підключати інші шаблони сторінки. На рисунку 3.6 зображено налаштування макетів, де у 19 рядку буде взято файл макету menuUser.hbs з папки partials (рис.3.7).

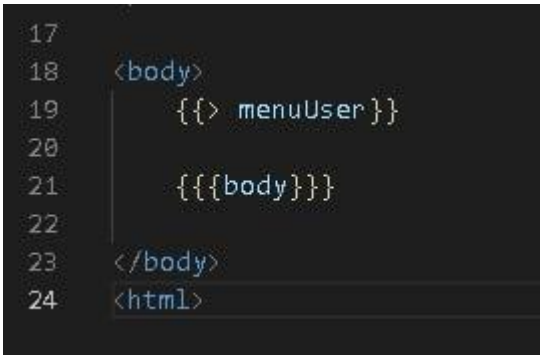


Рисунок 3.6 – Налаштування макетів

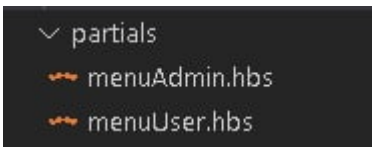


Рисунок 3.7 – Вміст папки partials

У рядку 21 рисунку 3.6 відбувається налаштування позиції основного шаблону тіла сторінки, що буде вказуватися сервером під час GET-запиту (рис.3.8).

```
13    app.get('/admin/createUser',
14        [verifySignUp.verifyToken,
15        verifySignUp.isAdmin],
16        (req, res) => {
17            dataBase.getRoles().then(result => {
18                res.render("createUser.hbs", {
19                    roles: result,
20                    layout: 'layoutAdmin.hbs'
21                });
22            });
23        });
24
```

Рисунок 3.8 – Налаштування позиції основного шаблону тіла сторінки

Основні підключення бібліотек hbs і налаштування на сервері зображені на рисунку 3.9.

```
7   const hbs = require("hbs");
8   app.engine("hbs", expressHbs(
9     {
10      layoutsDir: "views/layouts",
11      defaultLayout: "layoutUser",
12      extname: "hbs"
13    }
14  ));
15  app.set("view engine", "hbs");
16  app.use(express.static(__dirname + '/views'));
17  hbs.registerPartials(__dirname + "/views/partials");
```

Рисунок 3.9 – Основні підключення бібліотек hbs

Для підключення та взаємодії з базою даних, використовується бібліотека sequelize. Спочатку було встановлено всі необхідні конфігурації для підключення до бази даних (рис.3.10).

```
const sequelize = new Sequelize(
    config.DB,
    config.USER,
    config.PASSWORD,
    {
        host: config.HOST,
        dialect: config.dialect,
        operatorsAliases: false,

        pool: {
            max: config.pool.max,
            min: config.pool.min,
            acquire: config.pool.acquire,
            idle: config.pool.idle
        }
    }
);
```

Рисунок 3.10 – Встановлення всіх необхідних конфігурацій для підключення до бази даних

Запит на підключення до бази даних відбувається у файлі app.js та зображено на рисунку 3.11.

```
const db = require("./app/models");

db.sequelize.sync({ force: false }).then(() => {
});
```

Рисунок 3.11 – Запит на підключення до бази даних

Для створення таблиці у базі даних, на сервері потрібно створити шаблон моделі необхідної таблиці (рис.3.12).

```
module.exports = (sequelize, Sequelize) => {
    const User = sequelize.define("user", {
        id: {
            type: Sequelize.UUID,
            defaultValue: Sequelize.UUIDV1,
            primaryKey: true
        },
        username: {
            type: Sequelize.STRING
        },
        email: {
            type: Sequelize.STRING
        },
        password: {
            type: Sequelize.STRING
        }
    });

    return User;
};
```

Рисунок 3.12 – Створення шаблону моделі необхідної таблиці

Підключення всіх моделей для таблиці в базу даних відбувається в одному файлі index.js. Зразок підключення усіх необхідних таблиць зображено на рисунку 3.13.

```
db.user = require("../models/user.model.js")(sequelize, Sequelize);
db.role = require("../models/role.model.js")(sequelize, Sequelize);
db.history = require("../models/history.model.js")(sequelize, Sequelize);
db.department = require("../models/department.model.js")(sequelize, Sequelize);
db.patient = require("../models/patient.model.js")(sequelize, Sequelize);
db.sity = require("../models/sity.model.js")(sequelize, Sequelize);
db.message = require("../models/message.model.js")(sequelize, Sequelize);
```

Рисунок 3.13 – Зразок підключення усіх необхідних таблиць

Після чого відбувається створення зв’язків між таблицями.

Усі необхідні компоненти для переходу між сторінками для користувача знаходяться в меню (рис.3.14). Програмна реалізація наведена на рисунку 3.15.

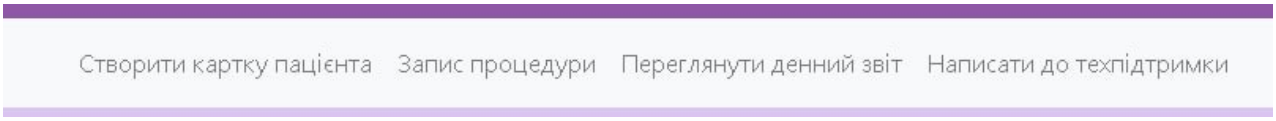


Рисунок 3.14 – Меню медичних сестер лазерної терапії, процедур з парафіном, масажиста

```
<div class="collapse navbar-collapse" id="navbarSupportedContent">
  <ul class="nav navbar-nav mx-auto">
    <li class="nav-item">
      <a class="nav-link" href="/createPatient">Створити картку пацієнта <span class="sr-only">(current)
    </li>
    <li class="nav-item">
      <a class="nav-link" href="/createProcedure">Запис процедури</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="/history">Переглянути денний звіт</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="/help">Написати до техпідтримки</a>
    </li>
  </ul>
</div>
```

Рисунок 3.15 – Програмна реалізація меню медичних сестер лазерної терапії, процедур з парафіном, масажиста

Для перехоплення усіх запитів було створено папку routers (рис.3.16).

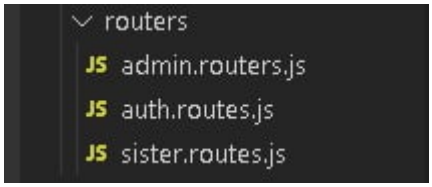


Рисунок 3.16 – Папка routers

Усі запити щодо категорій користувачів сестер відбуваються в папці sister.routes.js. Приклад запиту на відкриття сторінки технічної підтримки зображено на рисунку 3.17. Під час даного запиту першочергово відбувається перевірка на правильність токену, за що відповідає функція verifyToken, котра знаходиться в папці verifySignUp.js. Після чого відбувається перевірка чи є користувач сестрою лазерної терапії чи сестрою процедур з парафіном, чи масажистом, чи старшою сестрою. За це відповідає функція isUser, а для адміністратора функція isAdmin. А потім відбувається створення сторінки, де вказується ім’я файлу тіла сторінки.

```
84
85      app.get('/help',
86          [verifySignUp.verifyToken,
87          verifySignUp.isUser],
88          (req, res) => {
89              res.render("help.hbs");
90          });
91
```

Рисунок 3.17 – Приклад запиту на відкриття сторінки технічної підтримки

Для створення всіх нових даних у базі даних користувач застосовує POST запити. За допомогою створених шаблонів таблиць, відбуваються запити до бази даних, з метою створення чи знаходження необхідної інформації з метою модифікації цієї таблиці.

Програмний код створення пацієнта наведено на рисунку 3.18. За допомогою шаблона patient викликається метод create та всередині нього записується вся необхідна інформація для створення таблиць. Після цього, за допомогою метода then відбувається отримання створеного об’єкта , що дає змогу взаємодіяти з ним. Для перехоплення помилок використовується метод catch. За допомогою нього можна детально переглянути інформацію про помилку.

```
12      Patient.create({
13          name: req.body.username,
14          date: new Date(req.body.date),
15          code: req.body.code,
16          sityId: req.body.sityId,
17          departmentId: req.body.departmentId
18      }).then(user => {
19          if (user) {
20              res.send({ message: "Пацієнт зареєстрований!" });
21          }
22      }).catch(err => {
23          res.status(500).send({ message: err.message });
24      });
```

Рисунок 3.18 – Програмний код створення пацієнта

Однією із переваг бібліотеки sequelize є підключення моделей однієї таблиці під час взаємодії з іншою, яка має детальний зв’язок з попередньою (рис.3.19).

Відбувається отримання всіх повідомлень, також ім’я користувача з моделі User і взято назву ролі цього користувача з моделі Role. Результат даного запиту продемонстровано на рисунку 3.20.

```
93          Message.findAll({
94              attributes: ['id','description', 'priority'],
95              include:{
96                  model: User,
97                  attributes: ['username'],
98                  include:{
99                      model: Role,
100                     attributes: ['name'],
101                 }
102             },
103             raw: true
```

Рисунок 3.19 – Демонстрація підключення таблиць


```
[
    {
        "id": "93bc8870-c711-11eb-ad73-8d7ecfd3f7ed",
        "description": "знайшлм орфографічну помилку",
        "priority": 3,
        "user.username": "Токар Анастасія Сергіївна",
        "user.role.id": "54bba450-c6f6-11eb-b125-0981ced3e622",
        "user.role.name": "Сестра медична лазерної терапії"
    }
]
```


Рисунок 3.20 – Результат запиту зв’язку таблиць

Програмний код у повному його обсязі наведено у додатку В.

3.3 Використання програмного додатку

У даному розділі описано процес роботи програми, її головних навігаційних схем, а також процес використання з відображенням головних функціональних можливостей.

Співробітники відділення фізичної реабілітації та медицини мають відкрити браузер, що встановлений на комп’ютері та перейти за посиланням задля отримання доступу до веб-додатку. Для успішності виконання попередньо вказаних дій, необхідно і достатньо мати стабільне підключення в цей момент до всесвітньої мережі Інтернет.

Після вдалого переходу користувача за посиланням, йому доступна сторінка з авторизацією до системи (рис. 3.20).

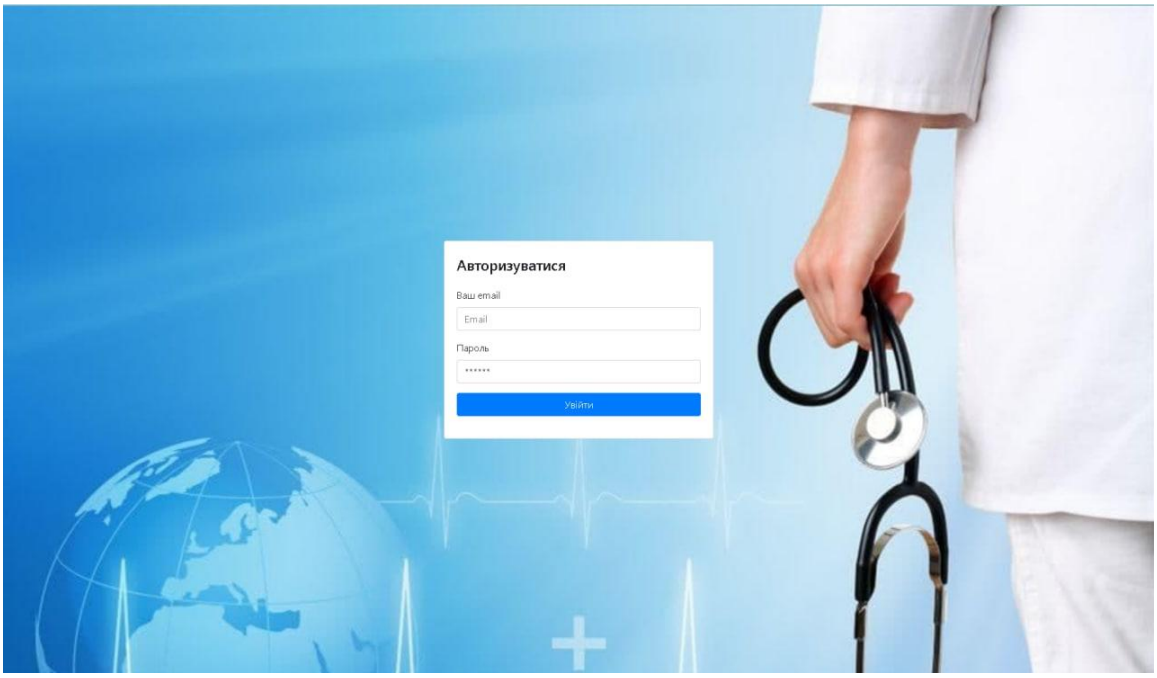


Рисунок 3.20 – Сторінка авторизації до системи

У разі неправильно введеного поля пошти чи паролю, користувач буде проінформований відповідним повідомленням (рис. 3.21-3.22).

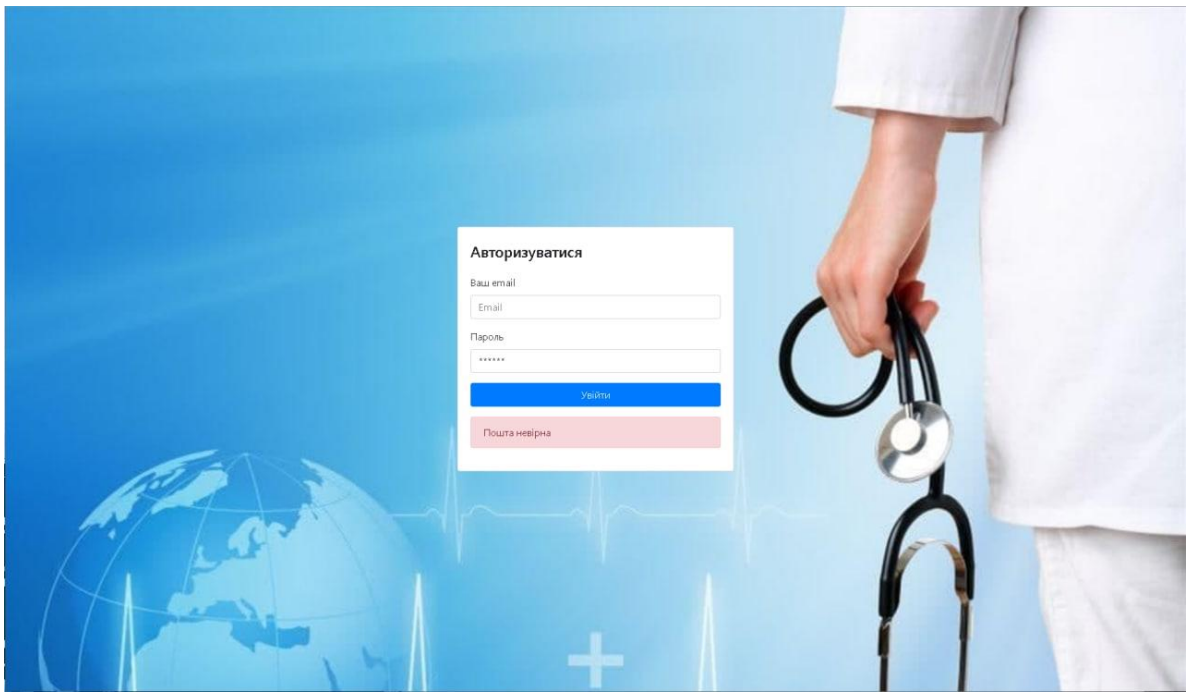


Рисунок 3.21 – Повідомлення користувача про неправильно введену пошту

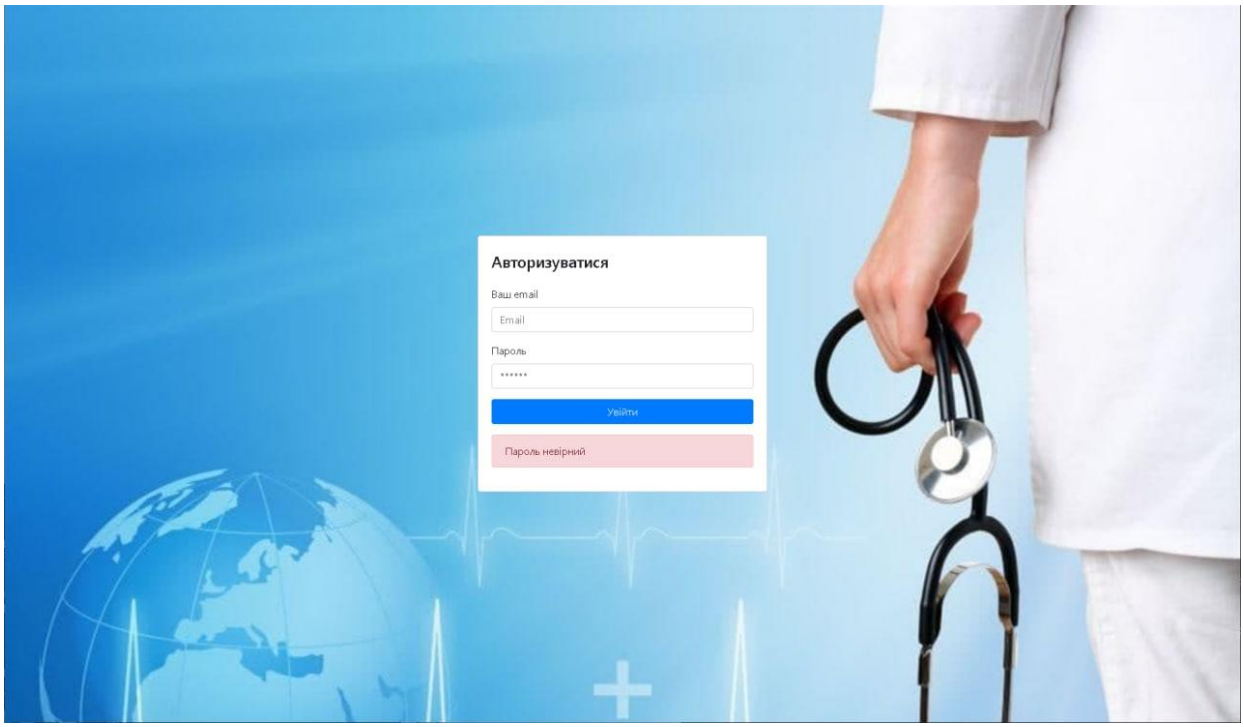


Рисунок 3.22 – Повідомлення користувача про неправильно введений пароль

Після успішної авторизації під профілем медичної сестри лазерної терапії стає доступна наступна сторінка функціональних можливостей (рис. 3.23). На даній сторінці можна переглянути саме під яким профілем було виконано авторизацію.

Поміж іншим, доступне до перегляду меню, що дублюється на кожній із сторінок, в рамках функціональних обов’язків авторизованого користувача, а саме: сестри лазерної терапії в даний момент.

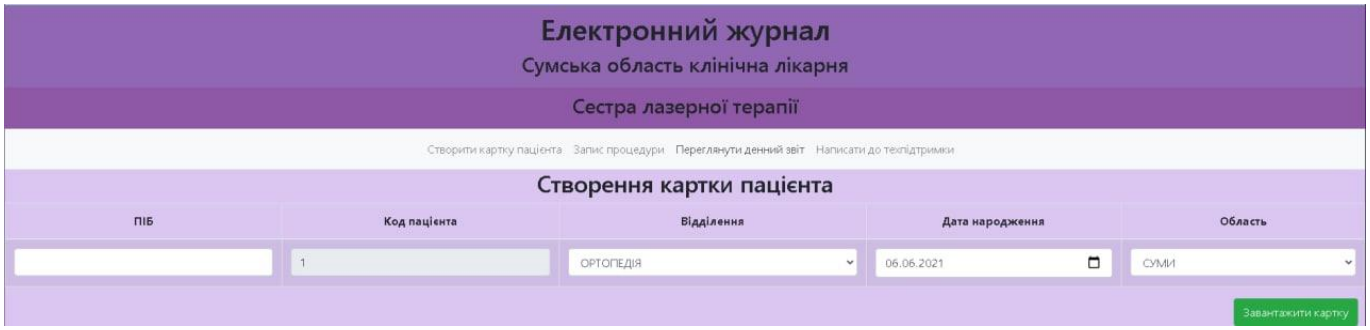


Рисунок 3.23 – Сторінка функціональних можливостей сестри лазерної терапії

Дана медична сестра лазерної терапії, як і інші (сестра медична процедур з парафіном, масажист, старша сестра), має можливість створити картку пацієнта. Унікальними даними кожного пацієнта є «код пацієнта». Під час звернення хворого до відділення фізичної реабілітації на нього створюється картка із зазначенням прізвища, ім’я, по батькові; коду пацієнта, що за замовчуванням для кожної картки ітеративно збільшується; відділення, від якого було направлено хворого на лікування до фізичної реабілітації та медицини; дату народження хворого; його адресу (рис. 3.23).

Відділення можна обрати із випадаючого списку за доступними відділеннями, котрі переглядаються скролінгом (рис. 3.24). Обрати область проживання пацієнта також можна обрати із випадаючого списку (рис. 3.25).

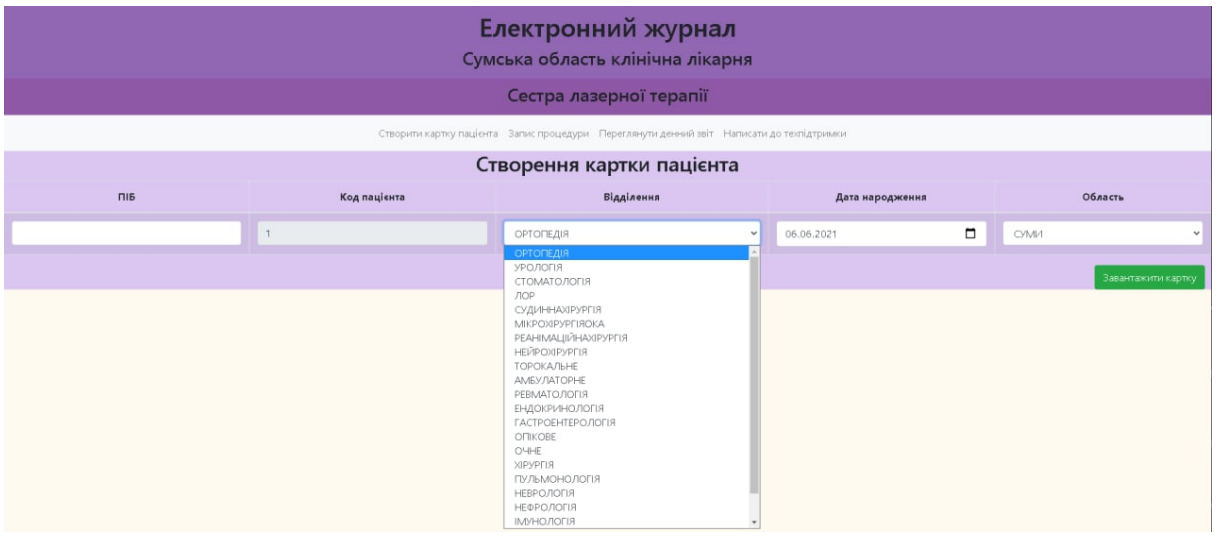


Рисунок 3.24 – Можливість вибору відділення

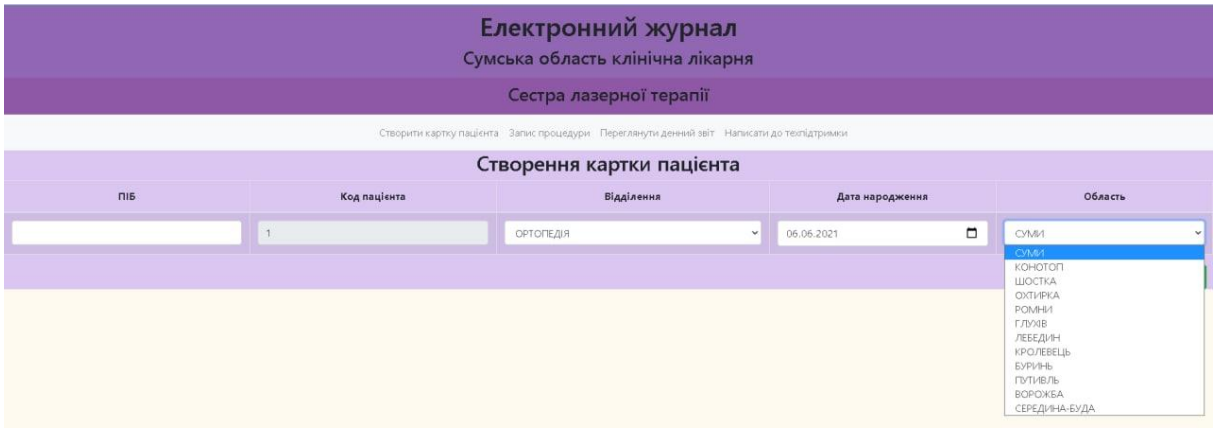


Рисунок 3.25 – Можливість вибору району

Також є можливість обрати дату народження із перегляду календаря (рис. 3.26).

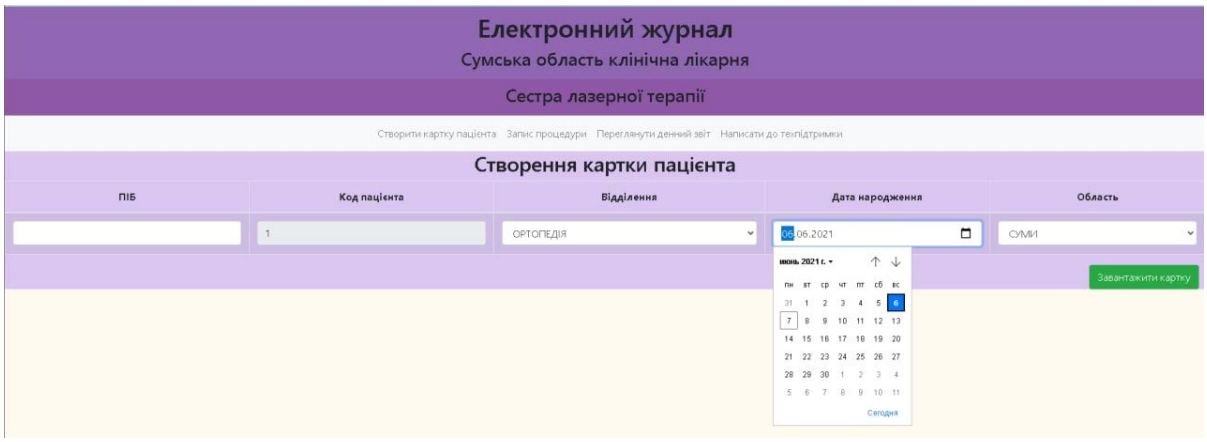


Рисунок 3.26 – Можливість вибору дати народження пацієнта

У випадку, якщо даному пацієнту вже створювали раніше карту та він знає свій код пацієнта, то сестра може перевірити чи існує така картка за унікальними даними - за кодом пацієнта.

Після заповнення усіх даних про пацієнта є можливість завантаження картки до бази даних, для цього необхідно натиснути кнопку «Завантажити картку». Якщо будь-яке поле було випадково залишено пустим та не заповненим, то користувач буде проінформованим, якщо буде намагатися завантажити картку до бази даних (рис. 3.27).

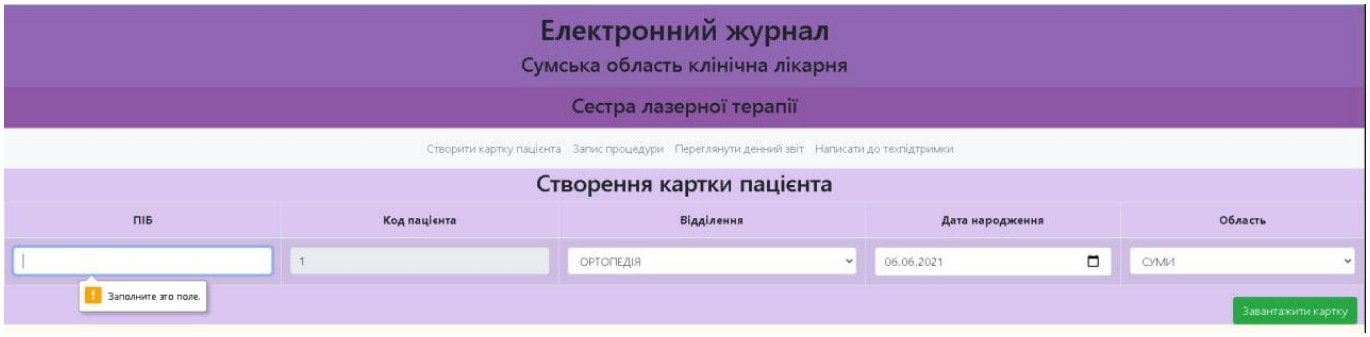


Рисунок 3.27 – Повідомлення про залишення пустого обов’язкового поля

Також сестра лазерної терапії має можливість записати проведену процедуру на пацієнта, що прийшов на лікування упродовж робочого дня (рис. 3.28). По введенню коду пацієнта у відповідному полі, з’являється інформація про ПІБ пацієнта та область його проживання.

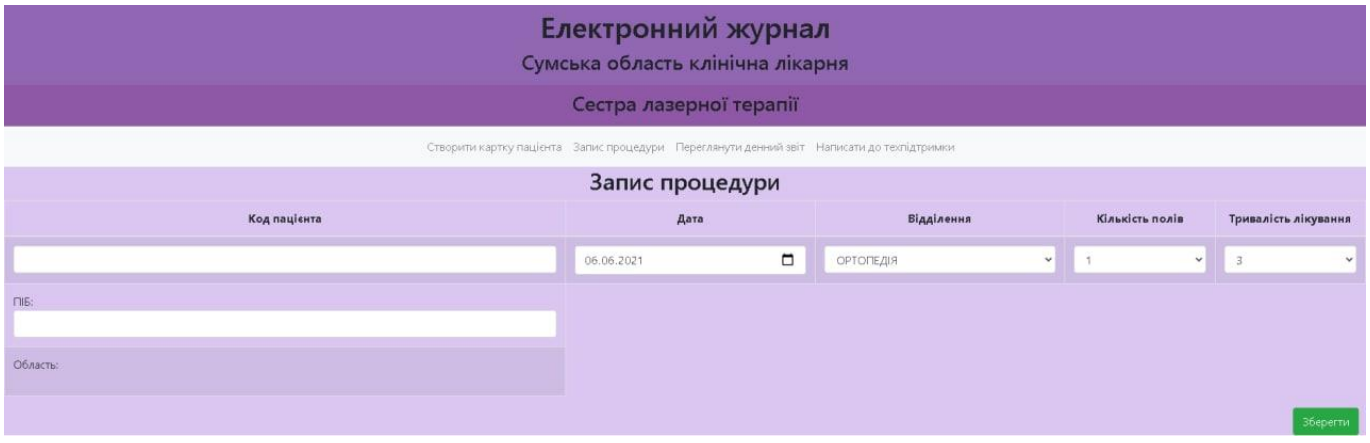


Рисунок 3.28 – Можливість запису процедури сестрою лазерної терапії

Кількість пролікованих полів для одного пацієнта можна обрати від 1 до 10 із випадаючого відповідного списку (рис. 3.29).

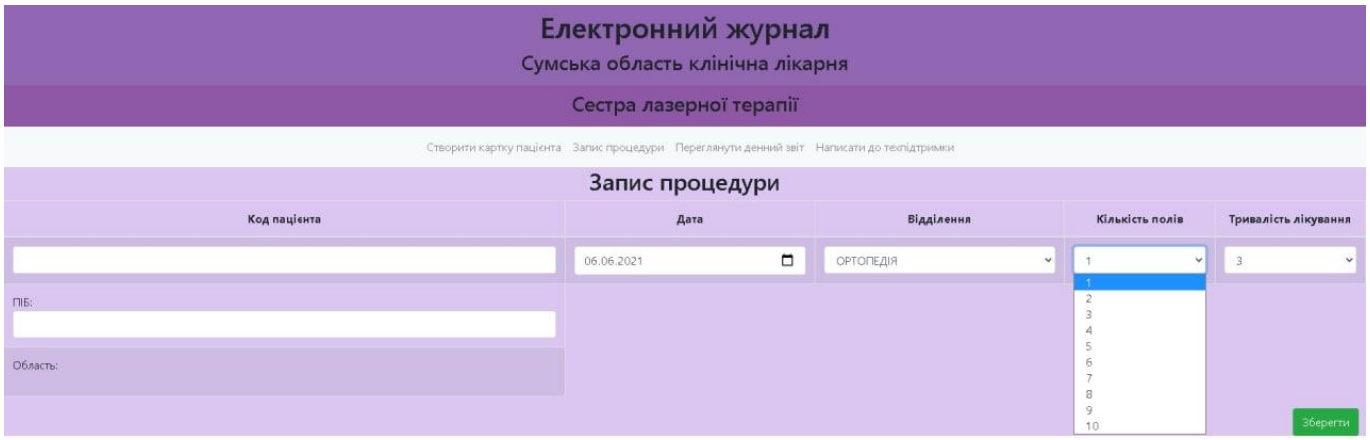


Рисунок 3.29 – Можливість обрати кількість пролікованих полів

Тривалість лікування можна обрати із значенням в діапазоні від 3 до 30 хв із випадаючого списку (рис 3.30).

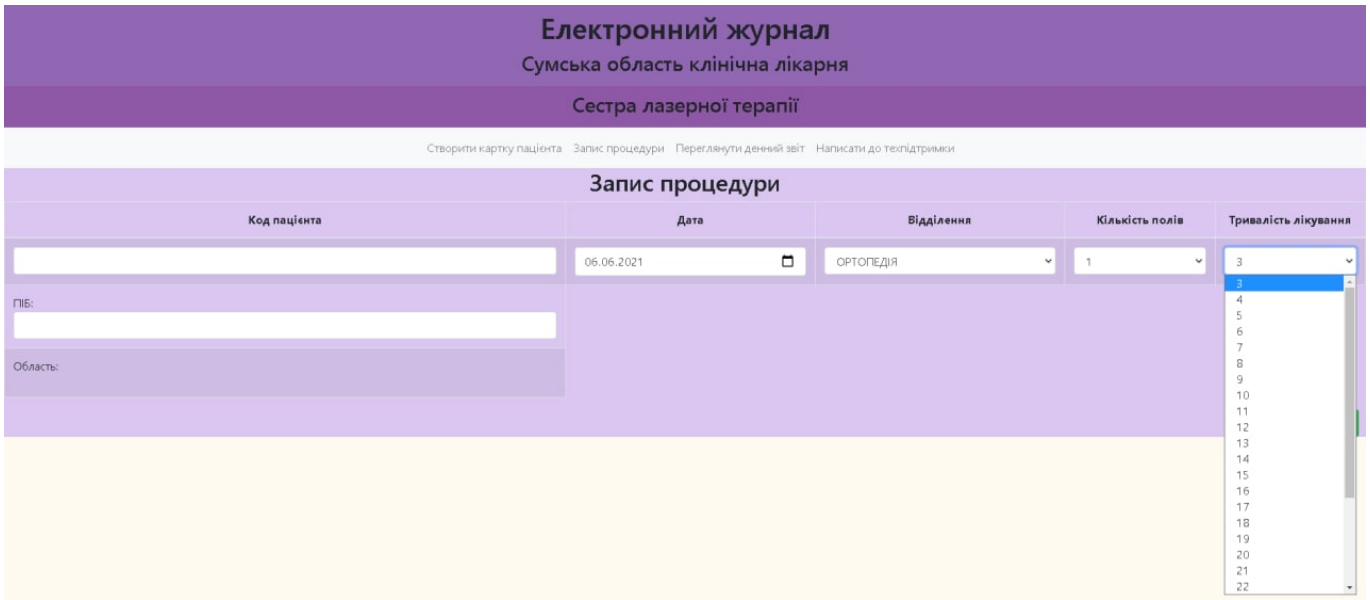


Рисунок 3.30 – Можливість обрати тривалість лікування

У випадку, якщо поле коду пацієнта залишились порожнім, користувач буде проінформованим про необхідність заповнення поля, якщо забажає зберегти процедуру до бази даних (рис. 3.31). Після заповнення необхідних даних, користувач

натискає кнопку «Зберегти» та відбувається запис процедури до бази даних (рис.3.32).

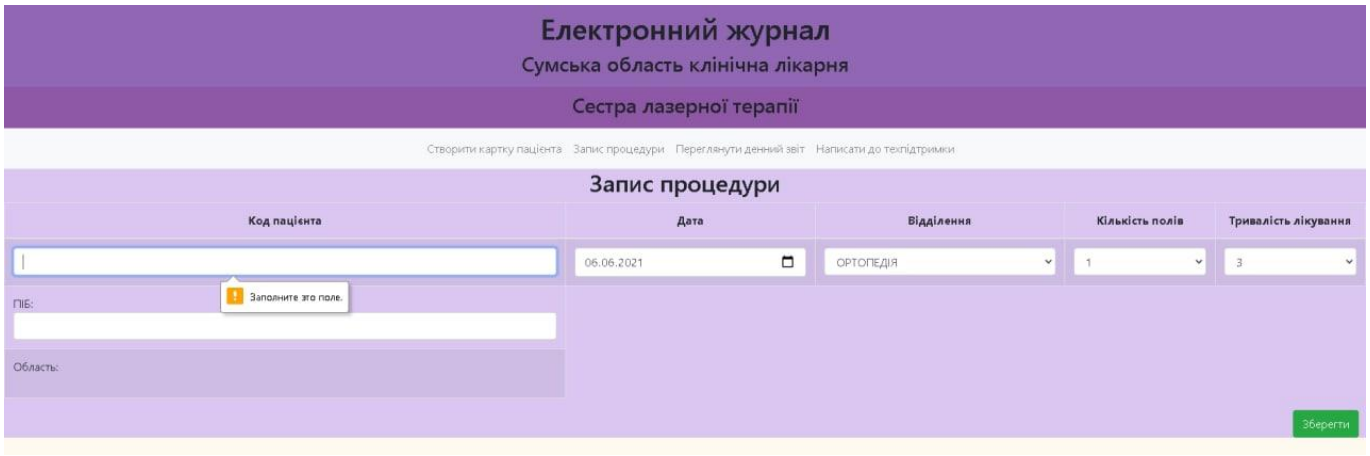


Рисунок 3.31 – Повідомлення про залишене порожнє поле

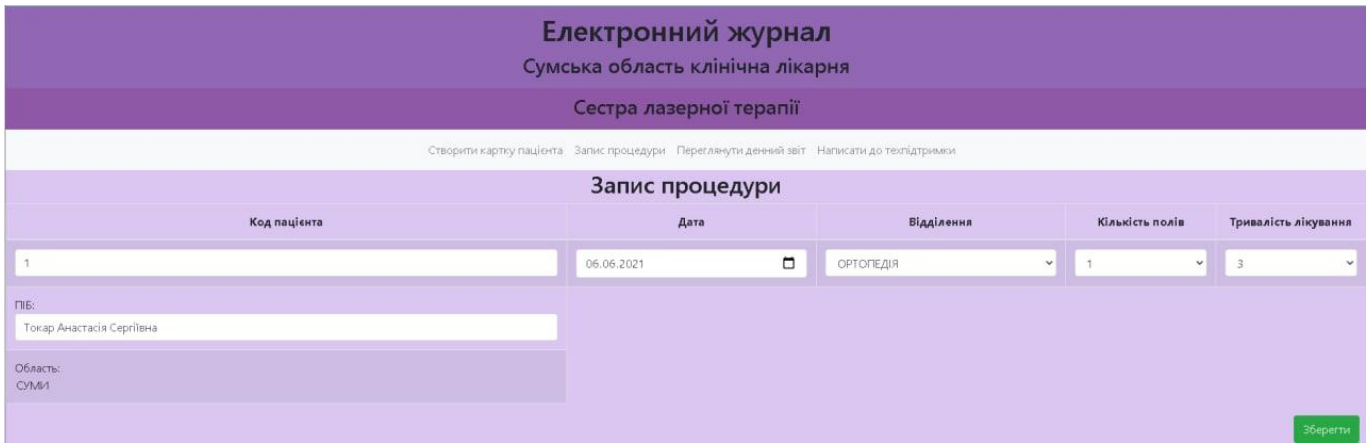


Рисунок 3.32 – Форма із заповненими даними

Якщо користувач був авторизований як сестра процедур з парафіном чи масажист, то функціональні особливості запису процедури залишаються майже тими ж, але змінюються логічні навантаження на головні колонки даних, що будуть зберігатися (рис. 3.33 - 3.34).

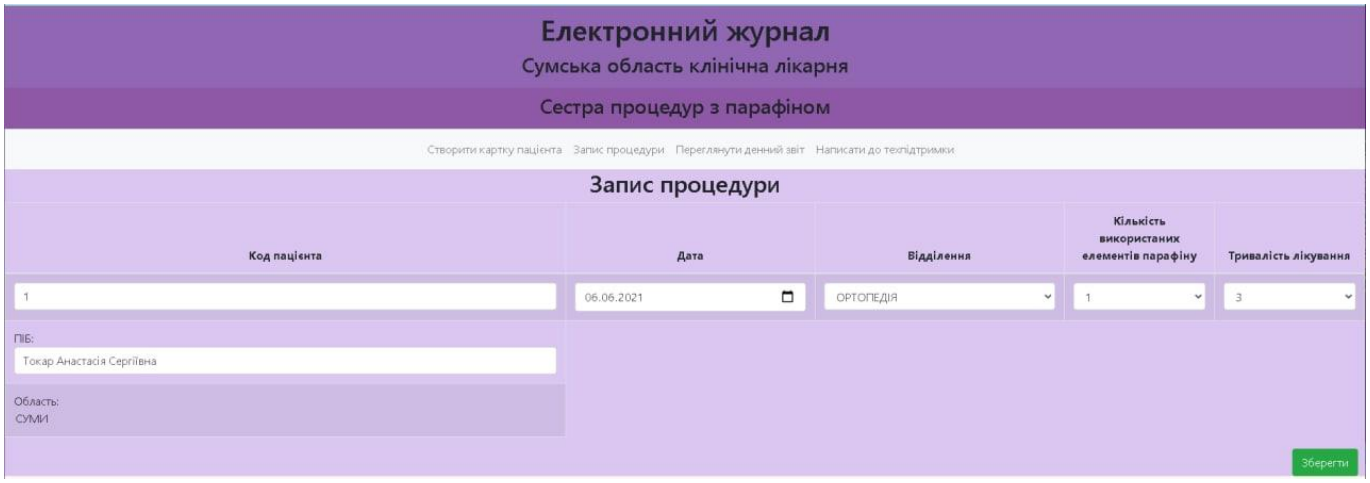


Рисунок 3.33 – Можливість запису процедури сестрою процедур з парафіном

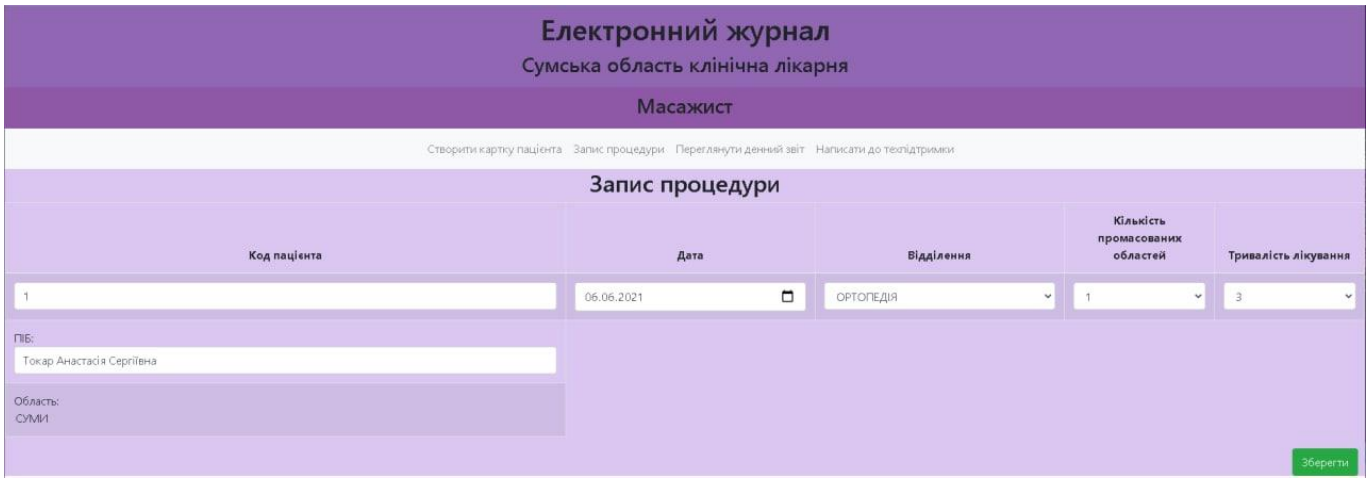


Рисунок 3.34 – Можливість запису процедури масажистом

Також для сестри лазерної терапії, сестри процедур з парафіном та масажисту доступна можливість переглянути денний звіт (рис. 3.35 – 3.37).

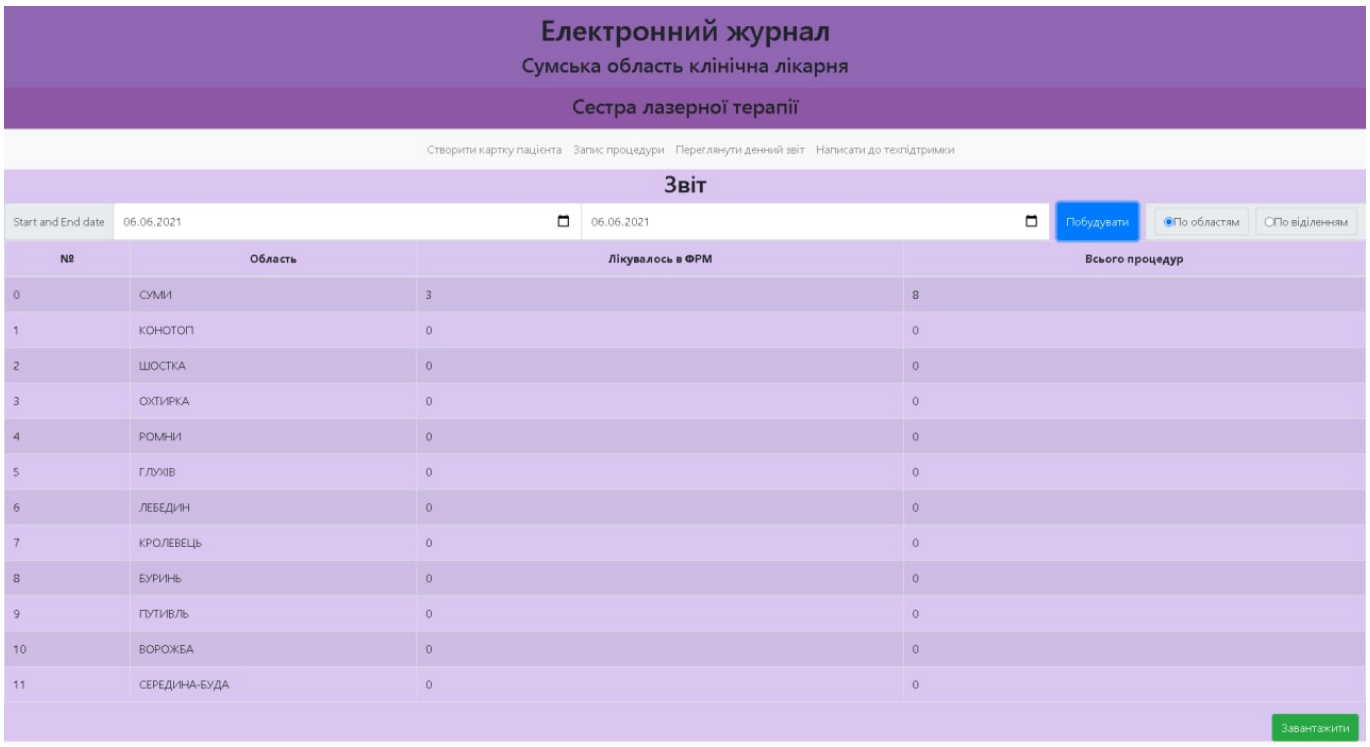


Рисунок 3.35 – Можливість перегляду денного звіту сестрою лазерної терапії

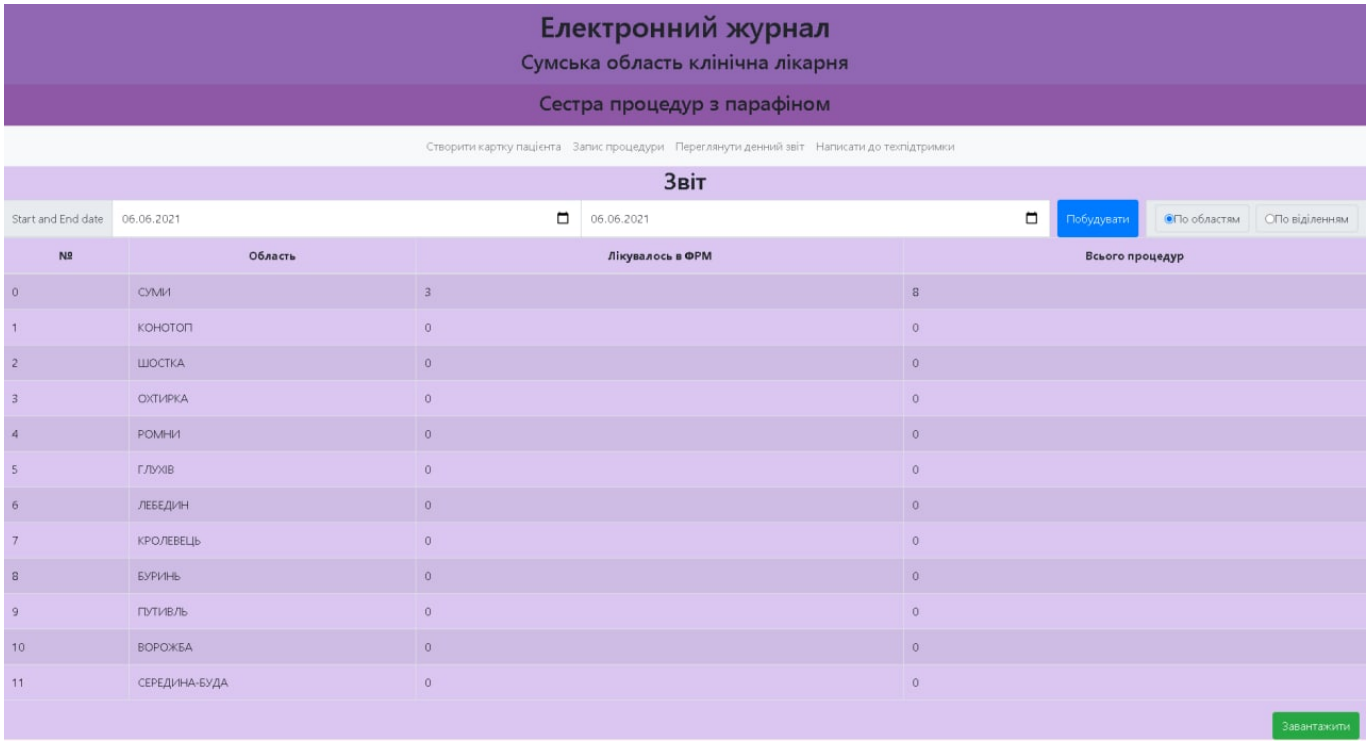


Рисунок 3.36 – Можливість перегляду денного звіту сестрою процедур з парафіном

Рисунок 3.37 – Можливість перегляду денного звіту масажистом

Для створення денного звіту є можливість обрати за яким критерієм можна виконати підрахунки: по областям, по відділенням. Також реалізована перевірка вибору початкової дати для підрахунку та кінцевої. Для користувачів сестра лазерної терапії, сестра процедур з парафіном та масажист надана можливість обрати одну дату. За замовчуванням реалізовано проставлення фінальної дати такої ж, як і початкова (рис. 3.38).

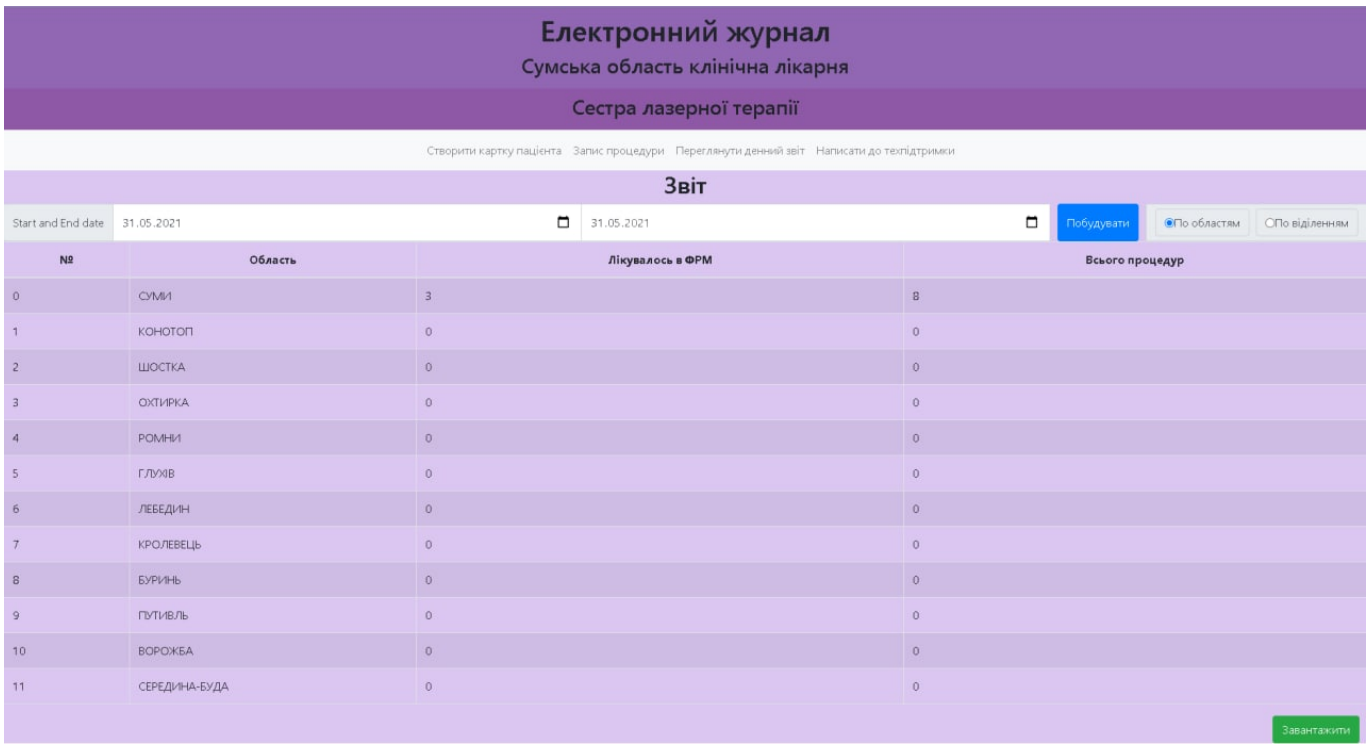


Рисунок 3.38 – Проставлення початкової та кінцевої дати

Після натискання кнопки «Побудувати», є можливість переглянути денний звіт у варіанті таблиці нижче форми у рамках даної сторінки. Також можна завантажити денний звіт у вигляді таблиці Microsoft Excel (рис.3.39).

ЗВІТ			
ПО МЕТОДАМ ЛІКУВАННЯ			
ФРМ КНП СОР "Сумська обласна клінічна лікарня"			
за період:	05.06.2021		
№ з\п	Відділення	Лікувалось в ФРМ	Всього процедур
1	ОРТОПЕДІЯ	5	20
2	УРОЛОГІЯ	0	0
3	СТОМАТОЛОГІЯ	3	6
4	ЛОР	12	25
5	СУДИННА ХІРУРГІЯ	6	23
6	МІКРОХІРУРГІЯ ОКА	14	35
7	РЕАНІМАЦІЙНА ХІРУРГІЯ	2	6
8	НЕЙРОХІРУРГІЯ	0	0
9	ТОРОКАЛЬНЕ	3	13
10	АМБУЛАТОРНЕ	6	18
11	РЕВМАТОЛОГІЯ	5	19
12	ЕНДОКРИНОЛОГІЯ	1	3
13	ГАСТРОЕНТЕРОЛОГІЯ	2	2
14	ОПІКОВЕ	4	16
15	ОЧНЕ	0	0
16	ХІРУРГІЯ	0	0
17	ПУЛЬМОНОЛОГІЯ	4	14
18	НЕВРОЛОГІЯ	5	26
19	НЕФРОЛОГІЯ	0	0
20	ІМУНОЛОГІЯ	0	0
21	ПРОКТОЛОГІЯ	0	0
22	СУДИННА НЕВРОЛОГІЯ	5	12
Всього			

Рисунок 3.39 – Завантажений на ПК денний звіт

Поміж користувачів сестра медична лазерної терапії, сестра процедур з парафіном, масажист, існує можливість авторизуватися під користувачем старшої сестри. У цьому випадку старша сестра може сформувати тижневий, місяцевий, річний звіти по кожному з направлень (лазерна терапія, процедури з парафіном, масаж). Також має можливість завантажити оформлені звіти (рис. 3.40).

Рисунок 3.40 – Можливість створювати старшій сестрі звітів

Окрім можливості створювати більш об’ємні по часовим проміжкам звіти, старша сестра має доступ створювати статистичні дані та завантажувати на персональний комп’ютер. Статистичні дані можна підрахувати по кожному із направлень: лазерна терапія, процедури з парафіном, масажист. Також є можливість обрати за який період підраховувати статистичні дані. Статистика реалізує підрахунки в відсотковому відношенні кількості пролікованих пацієнтів від даної області/відділення до загальної кількості пролікованих пацієнтів за обраний період (рис. 3.41). Статистичні дані оформлюються у вигляді таблиці Microsoft Excel із зазначенням періоду, за який вони підраховані, поряд можна переглянути відповідний звіт та безпосередньо підраховані статистичні дані, що оформлені у вигляді кругової діаграми (рис. 3.42).

Рисунок 3.41 – Можливість створювати статистичні дані

ЗВІТ			
ПО МЕТОДАМ ЛІКУВАННЯ			
ФРМ КНП СОР "Сумська обласна клінічна лікарня"			
за період:	05.06.2021		
№ з\п	Відділення	Лікувалось в ФРМ	Всього процедур
1	ОРТОПЕДІЯ	5	20
2	УРОЛОГІЯ	0	0
3	СТОМАТОЛОГІЯ	3	6
4	ЛОР	12	25
5	СУДИННА ХІРУРГІЯ	6	23
6	МІКРОХІРУРГІЯ ОКА	14	35
7	РЕАНІМАЦІЙНА ХІРУРГІЯ	2	6
8	НЕЙРОХІРУРГІЯ	0	0
9	ТОРОКАЛЬНЕ	3	13
10	АМБУЛАТОРНЕ	6	18
11	РЕВМАТОЛОГІЯ	5	19
12	ЕНДОКРИНОЛОГІЯ	1	3
13	ГАСТРОЕНТЕРОЛОГІЯ	2	2
14	ОПІКОВЕ	4	16
15	ОЧНЕ	0	0
16	ХІРУРГІЯ	0	0
17	ПУЛЬМОНОЛОГІЯ	4	14
18	НЕВРОЛОГІЯ	5	26
19	НЕФРОЛОГІЯ	0	0
20	ІМУНОЛОГІЯ	0	0
21	ПРОКТОЛОГІЯ	0	0
22	СУДИННА НЕВРОЛОГІЯ	5	12
Всього			

Рисунок 3.42 – Приклад завантаженого файлу із статистичними даними

У кожного з профілів користувачів є можливість звернутися до технічної підтримки, для цього є відповідний пункт меню (рис. 3.43).

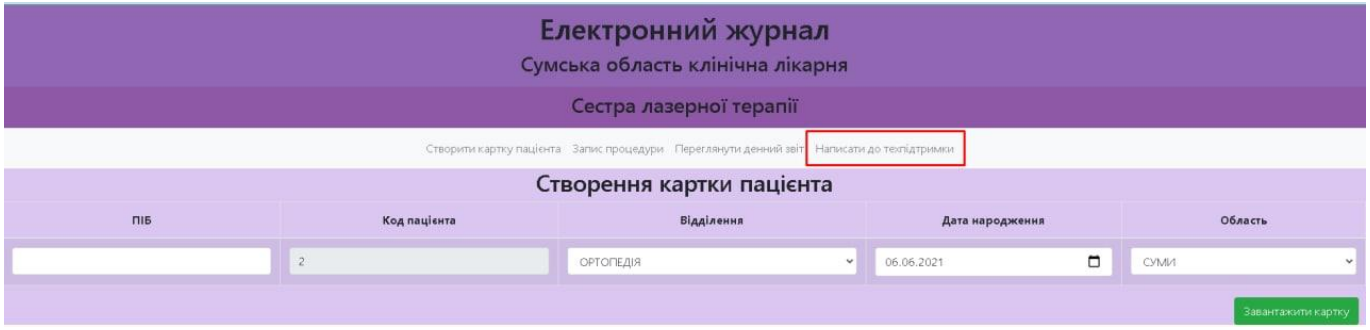


Рисунок 3.43 – Вкладка меню звернення до технічної підтримки

Форма звернення до техпідтримки наведена на рисунку 3.44.

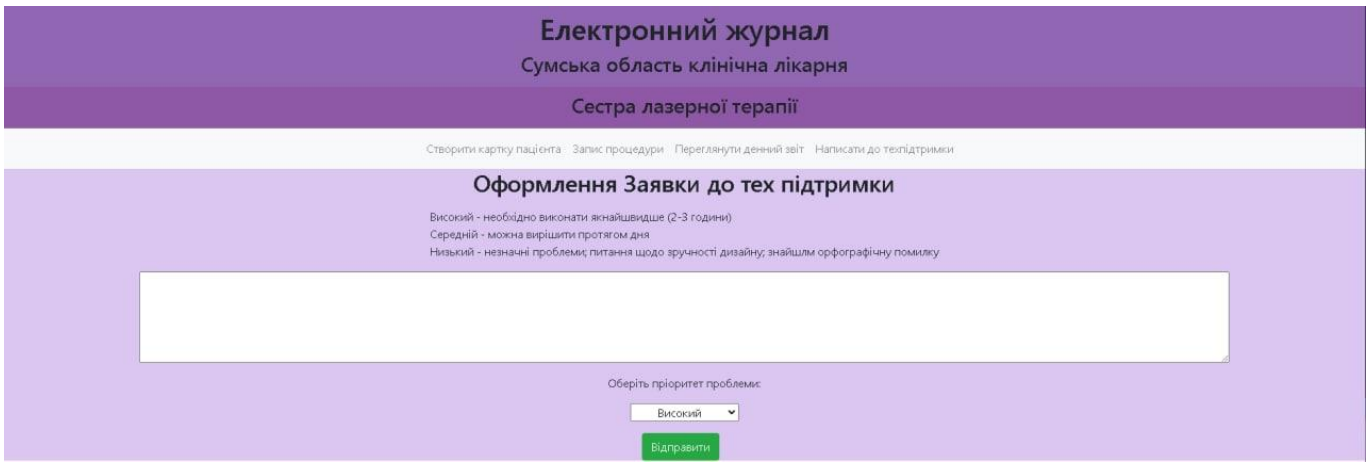


Рисунок 3.44 – Форма звернення до техпідтримки

Із кожного профілю авторизації в додатку є можливість вийти та повернутися до початкової форми авторизації (рис. 3.45). Також упродовж 30 хвилин відбувається автоматичний вихід із профілю до форми авторизації.

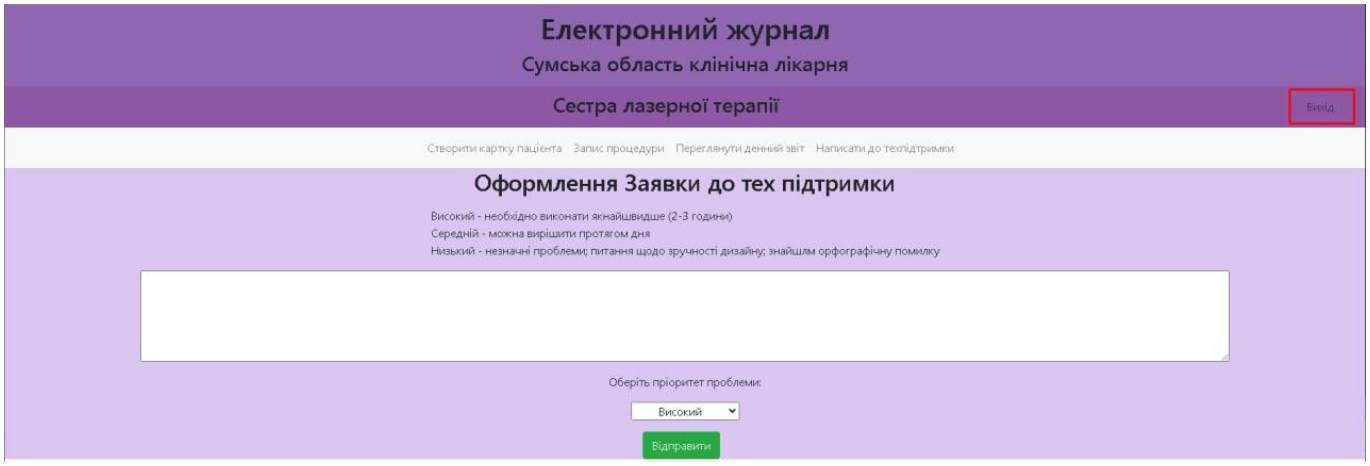


Рисунок 3.45 – Кнопка виходу з авторизованого профілю

Також доступний профіль адміністратора, функціонал якого наведено на рисунках 3.46 – 3.47.

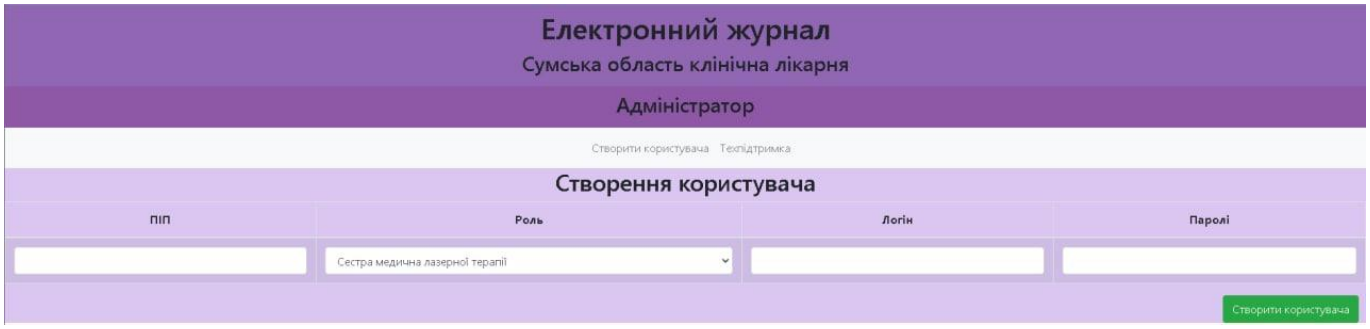


Рисунок 3.46 – Функціонал профілю адміністратора

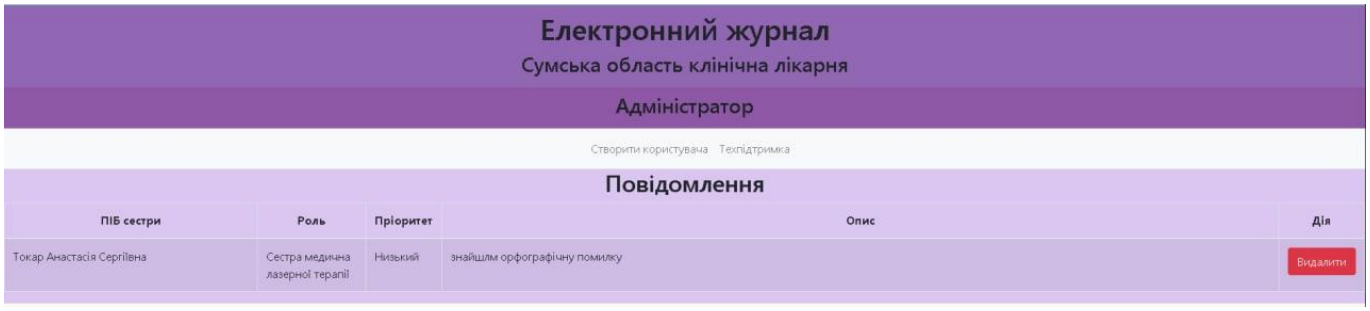


Рисунок 3.47 – Функціонал профілю адміністратора

ВИСНОВКИ

У результаті виконання дипломного проектування була сформульована мета та задачі, виконаний аналіз предметної області, визначено функціональні вимоги, описано архітектуру програмного додатку та виконано його програмну реалізацію.

Під час проведеного аналізу стану використання IT-технологій в медицині виявлено, що операції по обробці щоденних поточних даних про пацієнтів, які проходили лікування у фізіотерапевтичних відділеннях, до цього часу виконуються в ручному режимі. На підставі цього було визначено, що потрібно розробити програмний засіб, який би дозволив автоматизувати всі рутинні операції.

Тому метою дипломного проектування було визначено розробку web-додатку, який би спростив та пришвидшив ведення документації, збір статистичних даних та подальшу її обробку робітниками відділу фізичної реабілітації та медицини. Користувачу не потрібно буде вручну підраховувати дані, збирати статистичні дані, витрачати багато часу, який можна приділити для виконання інших професійних задач. Використання програмного продукту, який допомагає у веденні щоденних та місячних облікових звітів, значно зменшує навантаження робітника.

Було складено технічне завдання на виконання проекту. Прийнято рішення, що реалізувати розробку краще у форматі web-додатку. Визначено структуру додатку, функціональні вимоги до нього. Проведено планування робіт.

Також за результатами аналізу складено структурно-функціональну модель, яка описує процес інформаційної підтримки роботи медичної сестри відділу фізичної реабілітації та медицини. Моделювання проводилося у відповідності до вимог стандарту IDEF0. Складені UML-моделі дають представленні про майбутній програмний продукт. Вся інформація буде зберігатися у базі даних, модель якої складено за результатами аналізу предметної області, а точніше інформації, яка в ній циркулює.

На наступному етапі розроблену модель бази даних було втілено у вигляді фізичних реалізацій таблиць. Для забезпечення виконання додатком всіх висунутих функціональних вимог розроблено відповідне програмне забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ИТ в медицине: как технологии меняют одну из старейших отраслей [Електронний ресурс] – Режим доступу до ресурсу: https://www.cnews.ru/articles/2017-12-28_it_v_meditsine_kak_tehnologii_menyayut_odnu_iz_starejshih_otraslej
2. Токар А.С., Ващенко С.М. Web-додаток підтримки роботи медичної сестри фізичної реабілітації та медицини. // «Інформатика, математика, автоматика»: матеріали та програма науково-технічної конференції, м. Суми, 19 – 20 квітня 2021 р. – Суми: Сумський державний університет, 2021. – с. 100.
3. IT-технологии и машиностроение [Електронний ресурс] – Режим доступу до ресурсу: http://www.bogorodskmash.ru/it-texnologii-i-mashinostroenie/
4. Jubileen Aguilar. ROLE OF INFORMATION TECHNOLOGY IN EDUCATION SECTOR [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@iamaguilarjubileen1999/role-of-information-technology-in-education-sector-baef822a001e
5. Konsbruck Robert. Impacts of Information Technologyon Society in the new Century [Електронний ресурс] – Режим доступу до ресурсу: https://www.zurich.ibm.com/pdf/news/Konsbruck.pdf
6. Про HELSI [Електронний ресурс] – Режим доступу до ресурсу: https://helsi.me/about
7. Запис до лікаря, пошук ліків і декларації: Helsi.me запустив мобільний застосунок [Електронний ресурс] – Режим доступу до ресурсу: https://thepage.ua/ua/news/medichnij-servis-helsime-zapustiv-mobilnij-zastosunok

8. Анисимов В. В. Проектирование информационных систем: Разработка функциональной модели [Електронний ресурс] – Режим доступу до ресурсу: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6
9. Знакомство с нотацией IDEF0 и пример использования [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/trinion/blog/322832/
10. Леоненков. Самоучитель по UML. – http://khpi-iip.mipk.kharkiv.edu/library/case/leon/index.html
11. Справочник UML. Объектно-ориентированное проектирование. – https://openu.ru/Books/UML/Artifact.asp
12. Диаграмма коммуникации (Communication Diagram) [Електронний ресурс] – Режим доступу до ресурсу: https://studref.com/670299/menedzhment/diagramma_kommunikatsii_communication_diagram.
13. Express - фреймворк веб-приложений Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://expressjs.com/ru/.
14. Node.JS - Представления и движок представлений Handlebars [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/web/nodejs/4.7.php.
15. Введение в Handlebars, шаблонизатор JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://ru.code-maven.com/introduction-to-handlebars-javascript-templating-system.
16. Тузовський А.Ф. Проектування і розробка web-додатків – https://stud.com.ua/97571/informatika/proektuvannya_i_rozrobka_web-dodatkiv
17. Normalization of Database – https://www.studytonight.com/dbms/database-normalization.php#

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інформаційної системи
«Web-додаток підтримки роботи медичної сестри фізичної реабілітації та медицини»

ПОГОДЖЕНО:

Доцент кафедри комп’ютерних наук

_____ Ващенко С.М.

Студент групи ІТ-71

_____ Токар А.С.

Суми 2021

1. Призначення й мета створення web-додатку

1.1 Призначення web-додатку

Розроблювана інформаційна система має повністю замінити ведення щоденних записів про кожного пацієнта та його лікування, підраховувати статистичні дані по проведеним об’ємам роботи, формувати щоденні, щотижневі, місяцеві та річні звіти.

1.2 Мета створення web-додатку

Надання полегшення роботи сестри медичної відділення фізичної реабілітації та медицини, розвантаження співробітника для додаткових не менш важливих задач, що вимагаються посадою.

1.3 Цільова аудиторія

До цільової аудиторії, розроблюваної інформаційної системи, належать сестра медична фізичної реабілітації та медицини з процедур лазерної терапії, з процедур лікування парафіном, масажист та старша сестра. Також зацікавленою стороною є керівництво статистичних даних.

2 Вимоги до web-додатку

2.1 Вимоги до web-додатку в цілому

2.1.1 Вимоги до структури й функціонування web-додатку

Web-додаток повинен бути доступним в мережі Інтернет під доменним іменем, а також має складатися із взаємопов’язаних та залежних між собою розділів та чітко окреслених функцій, що виконують логічні, лаконічні дії, котрі зрозумілі на інтуїтивному рівні.

2.1.2 Вимоги до персоналу

Від персоналу, що буде експлуатувати розроблюваний web-додаток, не вимагається специфічних технічних навичок роботи з системою. Необхідно мати базові знання використання персонального комп'ютера та вміння користуватися всесвітньою мережею Інтернет. Усі дії у web-додатку будуть інтуїтивно зрозумілими та нескладними.

2.1.3 Вимоги до збереження інформації

Уся інформація, що буде використана у розроблюваній інформаційній системі, буде зберігатися у базі даних, котра буде реалізована засобами системи управління базами даних MySQL.

2.1.4 Вимоги до розмежування доступу

Розроблюваний web-додаток передбачено створити загальнодоступним.

Безпосередньо за правами доступу користувачів буде розділено за профілями сестра медична лазерної терапії, сестра медична лікування парафіном, масажист, старша сестра, адміністратор.

Відвідувачі сторінки можуть ознайомитися з короткою інформацією про відділення фізичної реабілітації та медицини.

За унікальним логіном та паролем буде надано доступ до профілів сестри медичної лазерної терапії, сестри медичної лікування парафіном, масажиста відповідно. У кожному з профілів співробітник, в залежності від займаної посади, зможе вносити дані лікування пацієнту щодня.

За унікальним логіном та паролем також буде надано можливість входу старшій сестрі для відповідно формування звітів щотижневих, місяцевих, річних, а також буде надана можливість формування статистичних даних.

У профіль адміністратора також буде надано доступ за логіном та паролем. Користувач під профілем адміністратора зможе редагувати внесені дані щоденні, крім того, буде надана можливість внести зміни до сформованих звітів. Адміністратор

також буде мати можливість додати блоки зовнішнього вигляду на наповнення профілів новими даними.

Відвідувачі можуть переглядати усі сторінки web-додатку, які доступні згідно їх статусу в системі після авторизації.

2.2 Структура web-додатку

2.2.1 Загальна інформація про структуру web-додатку

Структура web-додатку являє собою головну сторінку з загальною інформацією про відділення фізичної реабілітації та медицини та можливими профілями з наступною авторизацією.

Такими профілями є:

Профіль сестри медичної фізичної реабілітації лазерної терапії – на сторінці доступна форма із занесенням інформації про пацієнта, область його проживання, кількість одиниць процедур за один раз, що призначена йому на лікування. Також є можливість сформувати денний звіт та зробити його доступним старшій сестрі для подальшого формування звітності.

Профіль сестри медичної фізичної реабілітації лікування парафіном – на сторінці доступна форма із занесенням інформації про пацієнта, область його проживання, область тіла, яка призначена на лікування та тривалість процедури. Також є можливість сформувати денний звіт та зробити його доступним старшій сестрі для подальшого формування звітності.

Профіль масажиста фізичної реабілітації – на сторінці доступна форма із занесенням інформації про пацієнта, область його проживання, різновид масажу та тривалість проведення процедури. Також є можливість сформувати денний звіт та зробити його доступним старшій сестрі для подальшого формування звітності.

Профіль старшої сестри медичної фізичної реабілітації – сторінці доступна форма із формуванням звітів щотижневих, місяцевих, річних. Також доступна можливість сформувати аналіз статистичних даних.

Профіль адміністратора – на сторінці доступна форма із додаванням нових пунктів функціоналу попередньо вказаних профілів та редагування вже занесених даних або ж звітів.

2.2.2 Навігація

Після спілкування із замовником було прийнято рішення, що в кожному профілі буде доступне навігаційне меню, котре має бути розділене за логічними окремими задачами, задля кращого розуміння структури функціоналу та пришвидшення безпосередньо роботи користувачами-співробітниками відділення фізичної реабілітації та медицини.

2.2.3 Наповнення web-додатку (контент)

Для управління контентом web-додатку буде застосовано програмний редактор Brackets.

Для оформлення форм буде використано привабливо оформлений варіант дизайну форм, а саме програмний код с Bootstrap.

Уся інформація, котра буде оброблюватися в процесі взаємодії користувача з програмою, буде зберігатися в базі даних MySQL.

Заповнення та редагування контенту інформаційної системи буде реалізовано через програмний код, використовуючи інформацію з бази даних.

Усю інформацію для логічного та прикладного наповнення додатку буде надавати відділення фізичної реабілітації та медицини КНП СОР «Сумської обласної клінічної лікарні».

2.2.4 Дизайн та структура додатку

Дизайн та стиль розроблюваного web-додатку має відповідати сучасним оформленням, бути досить приємним та лаконічним для сприйняття. Щодо кольорової гамми було обрано пастельні тони, а саме: блідо-фіалковий.

Розташування елементів на головній сторінці кожного із профілів користувачів web-додатку схематично показано на рисунку А.1.

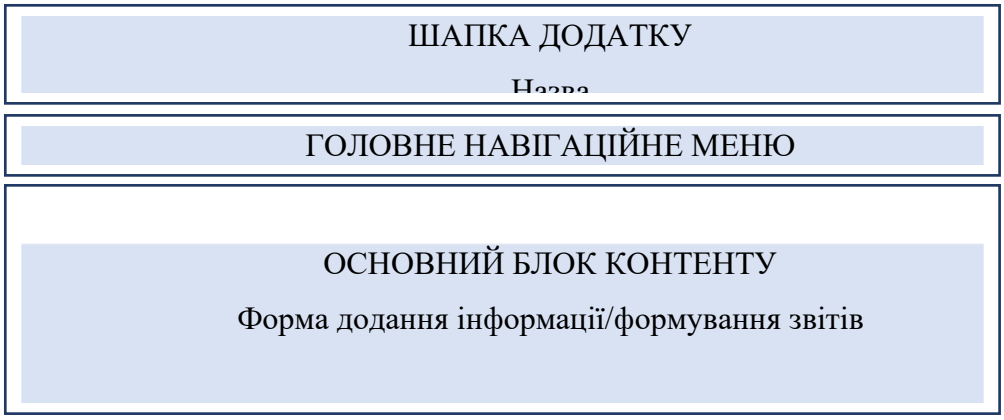


Рисунок А.1 – Схема головної сторінки

2.3 Вимоги до функціонування системи

2.3.1 Функціональні вимоги

На основі потреб користувача були визначено наступні функціональні вимоги:

- реєстрація та авторизація користувачів-співробітників;
- можливість занести персональні дані про пацієнта до бази даних;
- можливість занести до сховища область від якої направлено пацієнта;
-можливість формування та перегляду денного звіту/щотижневого/місяцевого/щорічного;
- формування звіту статистичних даних;
-адміністрування інформації, видалення, зміну занесених даних та створених звітів, зміна логіну та паролю користувачів.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація web-додатку відбувається з використанням:

- Brackets

- Bootstrap
- PHP 7.4.4
- MySQL 8.0

2.4.2 Вимоги до лінгвістичного забезпечення

Web-додаток має бути реалізований державною – українською мовою.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

- Веб-браузер: Internet Explorer 7.0 і вище, або Firefox 3.5 і вище, або Opera 9.5 і вище, або Safari 3.2.1 і вище, або Chrome 2 і вище.

4 Вимоги до складу й змісту робіт із введення web-додатку в експлуатацію

Для введення у доступність користування web-додатком його необхідно зробити доступним у мережі Інтернет, для чого необхідне доменне ім’я та місце на хостингу.

ДОДАТОК Б

ПЛАНУВАННЯ РОБІТ

Деталізація мети проекту методом SMART. Мета проекту: створити програмний додаток, який дозволить автоматизувати та покращити процес та методи управління проектами. Мета є досяжною, підґрунтям створення додатку є навички отримані на курсі з дисципліни Java, ООР, веб-розробка. Проект буде виконано вчасно, що підтверджується календарним планом проекту. Результати деталізації методом SMART розміщені у табл. Б.1.

Таблиця Б.1 − Деталізація мети методом SMART

Specific (конкретна)	Створити програмний додаток, який дозволить автоматизувати та покращити процес та методи управління проектами.
Measurable (вимірювана)	Результатом роботи проекту є робоча система з виконаними поставленими вимогами.
Achievable (досяжна)	Реалізації системи здійснюється за допомогою середовища розробки Intellij IDEA 2020.1, мови програмування Java, бази даних PostgreSQL, фреймворку Spring, технології Ajax.
Relevant (реалістична)	У наявності є всі необхідні технічні та програмні засоби. Розробники достатньо кваліфіковані для виконання поставлених задач.
Time-framed (обмежена у часі)	Ціль має часове обмеження. Робота повинна бути виконана у терміни, що були описані в календарному плані. Проект повинен бути виконаний згідно з календарним планом.

Планування змісту структури робіт. Основним інструментом для планування змісту структури робіт служить WBS діаграма − представлення проекту, виконане у вигляді ієрархічної структури робіт, що досягається за допомогою послідовної декомпозиції. Інструмент спрямований на детальне планування, оцінку вартості, визначення та розподіл персональної відповідальності виконавців та інші - тобто, на основні роботи і результати, що визначають зміст проекту.

Як правило, на верхньому рівні вказується сам проект, під ним (на першому рівні) - основні результати, кожен з яких, в свою чергу, деталізується, тобто наступний рівень завжди менше попереднього за обсягом робіт і, як правило, включає 2 і більше пакетів робіт. При цьому в різних гілках WBS може бути різна кількість рівнів в залежності від потрібного ступеня деталізації. Діаграма WBS зображена на рис. Б.1.

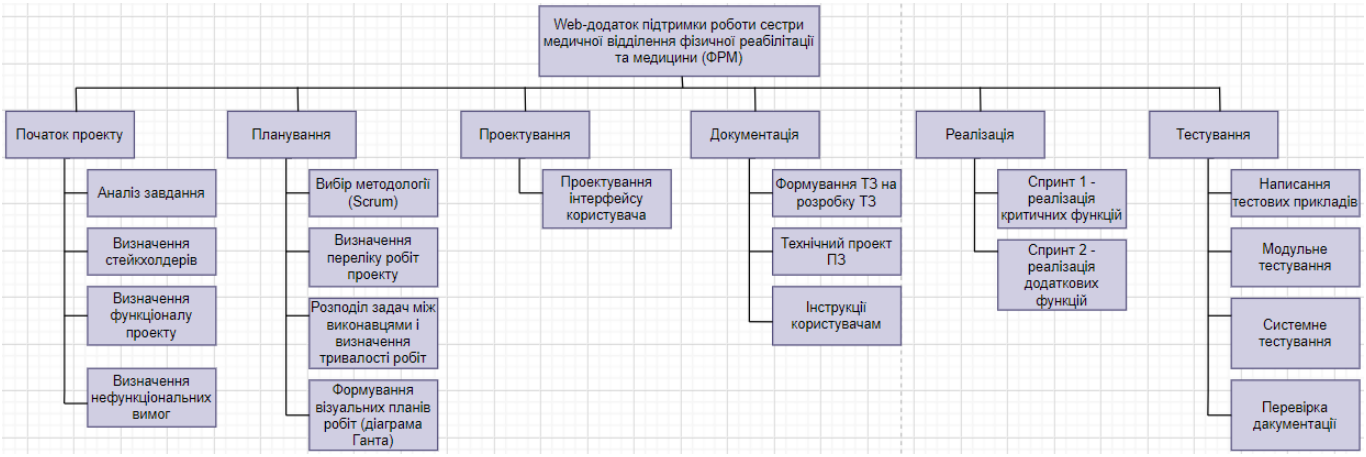


Рисунок Б.1 – WBS-структура проекту

Після того, як була побудована WBS структура проекту наступним етапом є розроблення OBS - склад, підпорядкованість, взаємодія і розподіл робіт по підрозділах і органам управління, між якими встановлюються певні відносини з приводу реалізації владних повноважень, потоків команд і інформації. Організаційна структура проекту стосується тільки внутрішньої організаційної структури проекту і не стосується відносин проектних груп чи учасників з батьківськими організаціями. Список виконавців, що функціонують в проекті представлений в таблиці Б.2 та на рисунку Б.2.

Таблиця Б.2 – Виконавці проекту

Роль	Ім’я	Проектна роль
Виконавець	Токар А.С.	Виконує розробку функціоналу системи та її інтерфейс.
Керівник проекту	Ващенко С.М.	Формує завдання на виконання проекту, корегує процес створення проекту та перевіряє виконання.

Діаграма OBS представлена на рисунку Б.2.

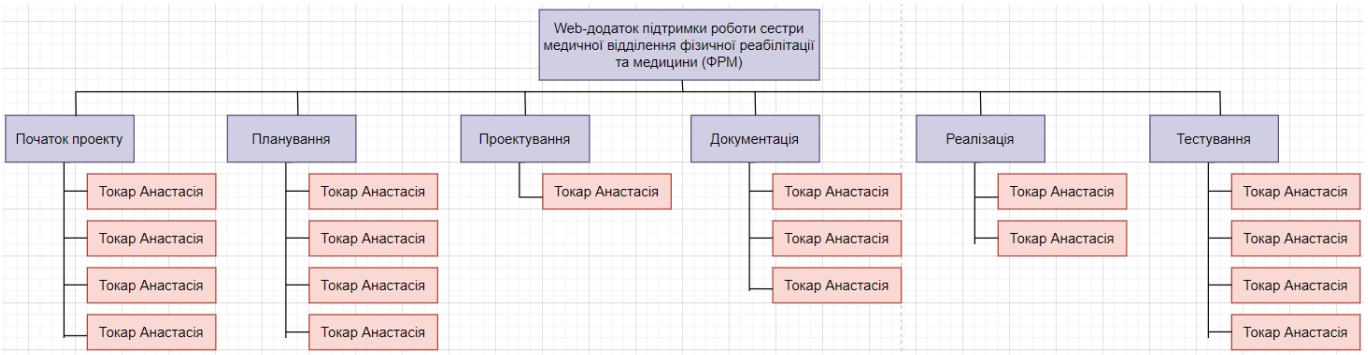


Рисунок Б.2 – OBS структура проекту

Діаграма Ганта. Діаграма Ганта вважається якісним інструментом для відображення цілей та задач. Основна відмінність цих інструментів - в способі демонстрації і відображення інформації.

На відміну від PDM – мережі, що пропонує мережеву модель, управління проектами з діаграмами Ганта засноване на форматі гістограм. Це допомагає відслідковувати відсоток робіт, виконаних по кожному завданню. Керівникам проектів дуже важливо правильно розподілити завдання і бути впевненими в тому, що проект буде завершений вчасно.

Основна увага діаграм Ганта зосереджено на процентному завершенні кожного завдання. Крім того, діаграми Ганта краще для проектів з невеликою кількістю взаємопов'язаних завдань.

Завдяки засобам програмного продукту Smartsheet була розроблена діаграма Ганта, яка у вигляді гістограми відображає тривалість кожного процесу, що був визначений на етапі формування WBS. Діаграма Ганта представлена на рисунку Б.3.

Аналіз ризиків. Ризик – ймовірнісна подія, яка може позитивно чи негативно вплинути на проект. Причиною виникнення ризиків є невизначеності, існуючі в кожному проекті. Ризики можуть бути «відомі» - ті, які визначені, оцінені, для яких можливе планування. Ризики «невідомі» - ті, які не ідентифіковані і не можуть бути прогнозовані. Хоча специфічні ризики і умови їх виникнення не визначені, але більшу частину ризиків можна передбачити.

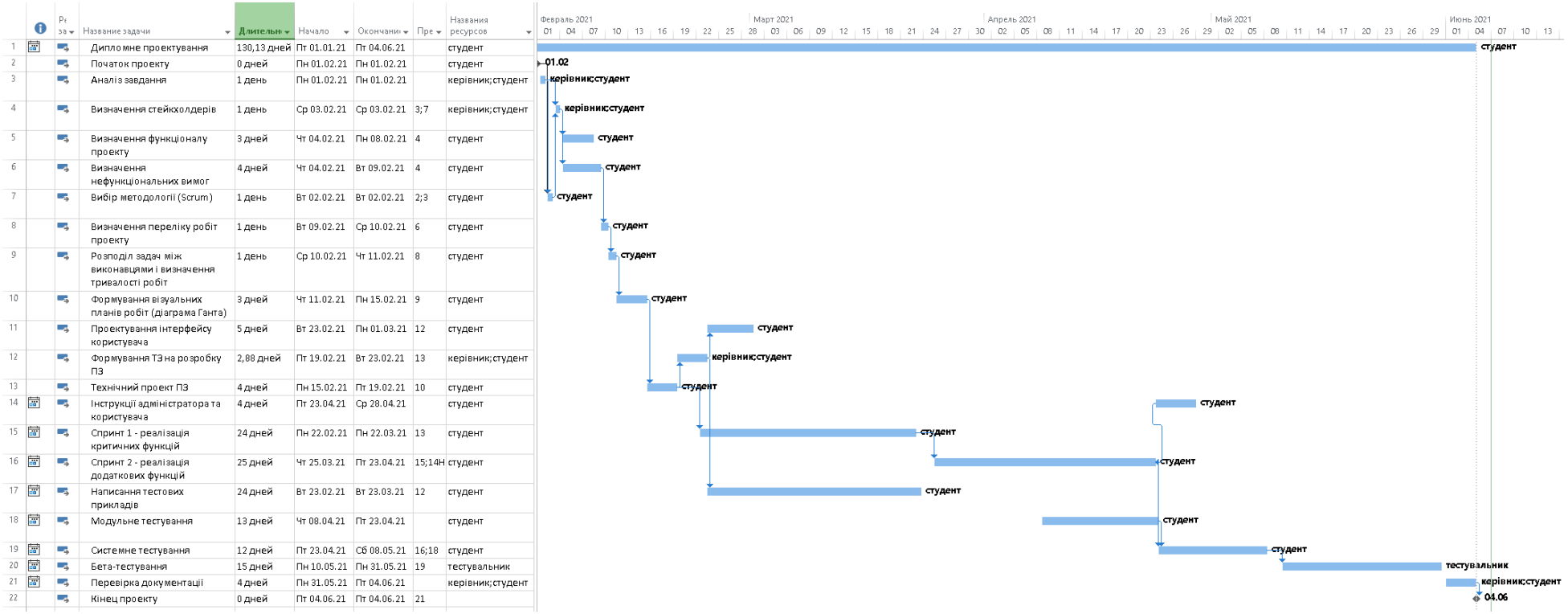
Ідентифікація ризиків - визначення ризиків, здатних вплинути на проект, і документування їх характеристик.

Ідентифікація ризиків визначає, які ризики здатні вплинути на проект, і документує характеристики цих ризиків. Ідентифікація ризиків не буде ефективною, якщо вона не буде проводитися регулярно протягом реалізації проекту.

Ідентифікація ризиків повинна залучати якомога більше учасників: менеджерів проекту, замовників, користувачів, незалежних фахівців.

Класифікація ризиків:

1. За імовірністю виникнення:
   - слабо ймовірнісні (Negligible) ;
   - мало ймовірнісні (Low);
   - імовірні (Medium);
   - досить імовірні (High);
   - майже імовірні (Critical).
2. За величиною впливу:
   - Мінімальна (Negligible);
   - Низька (Low);



	Режим задачи	Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
1		Дипломне проектування	130,13 дней	Пт 01.01.21	Пт 04.06.21		студент
2		Початок проекту	0 дней	Пн 01.02.21	Пн 01.02.21		студент
3		Аналіз завдання	1 день	Пн 01.02.21	Пн 01.02.21		керівник;студент
4		Визначення стейкхолдерів	1 день	Ср 03.02.21	Ср 03.02.21	3;7	керівник;студент
5		Визначення функціоналу проекту	3 дней	Чт 04.02.21	Пн 08.02.21	4	студент
6		Визначення нефункціональних вимог	4 дней	Чт 04.02.21	Вт 09.02.21	4	студент
7		Вибір методології (Scrum)	1 день	Вт 02.02.21	Вт 02.02.21	2;3	студент
8		Визначення переліку робіт проекту	1 день	Вт 09.02.21	Ср 10.02.21	6	студент
9		Розподіл задач між виконавцями і визначення тривалості робіт	1 день	Ср 10.02.21	Чт 11.02.21	8	студент
10		Формування візуальних планів робіт (діаграма Ганта)	3 дней	Чт 11.02.21	Пн 15.02.21	9	студент
11		Проектування інтерфейсу користувача	5 дней	Вт 23.02.21	Пн 01.03.21	12	студент
12		Формування ТЗ на розробку ПЗ	2,88 дней	Пт 19.02.21	Вт 23.02.21	13	керівник;студент
13		Технічний проект ПЗ	4 дней	Пн 15.02.21	Пт 19.02.21	10	студент
14		Інструкції адміністратора та користувача	4 дней	Пт 23.04.21	Ср 28.04.21		студент
15		Спринт 1 - реалізація критичних функцій	24 дней	Пн 22.02.21	Пн 22.03.21	13	студент
16		Спринт 2 - реалізація додаткових функцій	25 дней	Чт 25.03.21	Пт 23.04.21	15;14Н	студент
17		Написання тестових прикладів	24 дней	Вт 23.02.21	Вт 23.03.21	12	студент
18		Модульне тестування	13 дней	Чт 08.04.21	Пт 23.04.21		студент
19		Системне тестування	12 дней	Пт 23.04.21	Сб 08.05.21	16;18	студент
20		Бета-тестування	15 дней	Пн 10.05.21	Пн 31.05.21	19	тестувальник
21		Перевірка документації	4 дней	Пн 31.05.21	Пт 04.06.21		керівник;студент
22		Кінець проекту	0 дней	Пт 04.06.21	Пт 04.06.21	21	

Рисунок Б.3 – Діаграма Ганта

— Середня (Medium);
— Висока (High);
— Максимальна (Critical).

На основі цих даних була проведена класифікація ризиків для даного проекту, що наведена в таблиці Б.3.

Таблиця Б.3 – Класифікація ризиків

№	Назва ризику	Ймовірність	Величина втрат	Рівень ризику	План дій
1	Зміна ТЗ на етапі розробки	2	4	Середній	Проведення переаналізу проектування, стрімке впровадження змін у проекті
2	Недотримання термінів календарного плану	2	3	Середній	
3	Технологічні ризики, пов'язані з розробкою і налагодженням технології для розповсюдження завдань серед команди	3	5	Високий	Детальне проектування системи, налагодження роботи алгоритму обмінами задач

Продовження табл. Б.3

4	Збій в роботі готового продукту	3	3	Середній	Дотримання планування розробки та технічного завдання, аналіз пропозицій керівника.
5	Виникнення незапланованих робіт по проекту	3	4	Середній	Аналіз необхідних робіт та виділення резерву з ресурсів для можливих майбутніх додаткових робіт
6	Висока залежність від ключових співробітників (Недостатня кваліфікація для створення чи використання технологій)	1	5	Середній	Відбір якісних співробітників або вивчення нових технологій для вдалого використання

Продовження табл. Б.3

7	Залежність від ключових співробітників (хвороба)	2	2	Низький	Підтримка емоційного стану та здоров’я учасників проекту за рахунок фізичних вправ та часу на відпочинок

Використовуючи дану класифікацію, була побудована матриця ризиків, що представлена в таблиці Б.4.

Таблиця Б.4 - Матриця ризиків

5					
4					
3			4	5	3
2		7	2	1	9
1		8	10		6
Ймовірність / Величина втрат	1	2	3	4	5

ДОДАТОК В

КОДИ ОСНОВНИХ ФАЙЛІВ

config.js

```
module.exports = {
   HOST: "localhost",
   USER: "root",
   PASSWORD: "root",
   DB: "nastia",
   dialect: "mysql",
   pool: {
      max: 5,
      min: 0,
      acquire: 30000,
      idle: 10000
   }
};
```

admin.controller.js

```
const db = require("../models");
const User = db.user;
const Message = db.message;
const Role = db.role;
var bcrypt = require("bcryptjs");

exports.createUser = (req, res) => {
   User.create({
      username: req.body.username,
      email: req.body.email,
      password: bcrypt.hashSync(req.body.password, 8),
      roleId: req.body.roleId
   }).then(user => {
      if (user) {
```

```
            res.send({ message: "User was registered successfully!" });
        }
    }).catch(err => {
        res.status(500).send({ message: err.message });
    });
};

exports.getAllMessage = (req, res) => {
    Message.findAll({
        attributes: ['id', 'description', 'priority'],
        include: {
            model: User,
            attributes: ['username'],
            include: {
                model: Role,
                attributes: ['name'],
            }
        },
        raw: true
    }).then(messages => {
        var priority = ["Високий", "Средний", "Низький"];
        messages.forEach((element, index) => {
            element.priority = priority[element.priority - 1];
        });
        res.send(messages);
    }).catch(err => {
        res.status(500).send({ message: err.message });
    });
};

exports.deleteMessage = (req, res) => {
    Message.destroy({
        where: {
            id: req.body.IdMessage
```

```
        }
    }).then(messages => {
        res.send("");
    }).catch(err => {
        res.status(500).send({ message: err.message });
    });
};
```

auth.controller.js

```
const db = require("../models");
const User = db.user;
const Role = db.role;

//const Op = db.Sequelize.Op;

var jwt = require("jsonwebtoken");
var bcrypt = require("bcryptjs");

exports.signup = (req, res) => {
    Role.findOne({ where: { name: req.body.rolename } }).then(role => {
        User.create({
            username: req.body.username,
            email: req.body.email,
            password: bcrypt.hashSync(req.body.password, 8),
            roleId: role.id
        }).then(user => {
            if (user) {
                res.send({ message: "User was registered successfully!" });
            }
        }).catch(err => {
            res.status(500).send({ message: err.message });
        });
    });
};
```

```
exports.signin = (req, res) => {
    User.findOne({
        where: {
            email: req.body.email
        }
    })
        .then(user => {
            if (!user) {
                return res.status(404).send({ message: "User Not found." });
            }

            var passwordIsValid = bcrypt.compareSync(
                req.body.password,
                user.password
            );

            if (!passwordIsValid) {
                return res.status(401).send({
                    accessToken: null,
                    message: "Invalid Password!"
                });
            }

            Role.findByPk(user.roleId).then(role => {
                var authoritie = role.name.toUpperCase();
                var token = jwt.sign({ id: user.id, roleid: role.id }, "secret-key", {
                    expiresIn: 900 // 15 min
                });
                res.status(200).send({
                    username: user.username,
                    email: user.email,
                    roles: authoritie,
                    accessToken: token
                });
```

```
        });
    })
    .catch(err => {
        res.status(500).send({ message: err.message });
    });
};
```

sister.controller.js

```
const db = require("../models");
const Patient = db.patient;
const Sity = db.sity;
const History = db.history;
const Department = db.department;
const Message = db.message;
const { Op } = require("sequelize");
const DataBase = require("../middleware/dataBase");
const { message } = require("../models");

exports.setPatient = (req, res) => {
    Patient.create({
        name: req.body.username,
        date: new Date(req.body.date),
        code: req.body.code,
        sityId: req.body.sityId,
        departmentId: req.body.departmentId
    }).then(user => {
        if (user) {
            res.send({ message: "Patient was registered successfully!" });
        }
    }).catch(err => {
        res.status(500).send({ message: err.message });
    });
};
```

```
exports.getPatient = (req, res) => {
    Patient.findOne({
        where: {
            code: req.body.code
        },
        include: {
            model: Sity,
            attributes: ['name']
        }
    }).then(user => {
        if (user) {
            var result = {
                name: user.name,
                userId: user.id,
                sityId: user.sityId,
                sityName: user.sity.name
            };
            res.status(200);
            res.send(result);
        } else {
            res.status(404).send({ message: 'Cannot find' });
        }
    }).catch(err => {
        res.status(500).send({ message: err.message });
    });
};

exports.setProcedure = (req, res) => {
    History.create({
        date: new Date(req.body.date),
        durationTreatment: req.body.durationTreatment,
        userId: "e7806ac0-c6c2-11eb-ac80-a3c8cca07c6e", //змінити
        patientId: req.body.patientId,
        numberOfParts: req.body.numberOfParts,
```

```
        sityId: req.body.sityId,
        departmentId: req.body.departmentId
    }).then(user => {
        if (user) {
            res.send({ message: "Procedure was registered successfully!" });
        }
    }).catch(err => {
        res.status(500).send({ message: err.message });
    });
};

exports.getHistories = (req, res) => {
    var isSity = (req.body.isSity == 'true');
    var include = isSity?{
        model: Sity,
        attributes: ['name'],
    }:{
        model: Department,
        attributes: ['name'],
    };
    History.findAll({
        attributes: ['durationTreatment', 'numberOfParts', [(isSity?'sityId':'departmentId') , 'id']],
        where: {
            date: {
                [Op.gte]: new Date(req.body.dateStart), //>=
                [Op.lte]: new Date(req.body.dateEnd) //<=
            }
        },
        include: [include],
        raw: true
    }).then(histories => {
        DataBase.GetParameters(isSity).then(results => {
            if (histories) {
                histories.forEach( function(data) {
```

```
                data['criteriaName'] = data[(isSity?'sity.name':'department.name')];
                delete data[(isSity?'sity.name':'department.name')];
              });
        }
        let returns = [];
        results.forEach(result => {
          var test = DataBase.FindParametersFromArray(histories,result);
          returns.push(test);
        });
        res.send(returns);
      });

  }).catch(err => {
    res.status(500).send({ message: err.message });
  });
};

exports.sentMessage = (req, res) => {
  Message.create({
    description: req.body.description,
    priority: req.body.priority,
    userId: "442f1490-c6f7-11eb-b4f5-23034eceeb79", //змінити
  }).then(message => {
    if (message) {
      res.send({ message: "Message was registered successfully!" });
    }
  }).catch(err => {
    res.status(500).send({ message: err.message });
  });
};
```

dataBase.js

```
const db = require("../models");
const Sequelize = require('sequelize');
```

```
const Department = db.department;
const Patient = db.patient;
const Sity = db.sity;
const Role = db.role;
const Message = db.message;
const User = db.user;

function getSitiesAndDepartments() {
   return new Promise(function (resolve, reject) {
      Department.findAll({
         attributes: [`name`, `id`],
         raw: true
      }).then((departments) => {
         let departmentsName = departments;
         Sity.findAll({
            attributes: [`name`, `id`],
            raw: true
         }).then((sities) => {
            let sitiesName = sities;
            resolve({
               departmentsName: departmentsName,
               sitiesName: sitiesName
            });
         });
      });
   });
};

function getMaxCodePatient() {
   return new Promise(function (resolve, reject) {
      Patient.findOne({
         order: [['code', 'DESC']],
      }).then((patient) => {
         if (!patient) {
```

```
                resolve(1);
            } else {
                resolve(patient.code + 1);
            }
        });
    });
};

function GetParameters(isSity) {
    return new Promise(function (resolve, reject) {
        if (isSity) {
            Sity.findAll({
                attributes: ['id', 'name'],
                raw: true
            }).then((sities) => {
                resolve(sities);
            });
        } else {
            Department.findAll({
                attributes: ['id', 'name'],
                raw: true
            }).then((departments) => {
                resolve(departments);
            });
        }
    });
}

function FindParametersFromArray(array, Find) {
    var arr = array.filter(a => a.id == Find.id);
    var result = arr.reduce(function (acc, cur, i) {
        acc.durationTreatment += cur.durationTreatment;
        acc.numberOfParts += cur.numberOfParts;
        return acc;
```

```
    }, {
        durationTreatment: 0,
        numberOfParts: 0,
        id: Find.id,
        criteriaName: Find.name
    });
    return result;
}

function getRoles() {
    return new Promise(function (resolve, reject) {
        Role.findAll({
            attributes: ['id', 'name'],
            raw: true
        }).then((roles) => {
            resolve(roles);
        });
    });
}

function getMessages() {
    return new Promise(function (resolve, reject) {
        Message.findAll({
            attributes: ['id','description', 'priority'],
            include:{
                model: User,
                attributes: ['username'],
                include:{
                    model: Role,
                    attributes: ['name'],
                }
            },
            raw: true
        }).then(messages => {
```

```
            var priority = ["Високий", "Средний", "Низький"];
            messages.forEach((element, index) => {
               element.priority = priority[element.priority - 1];
            });
            resolve(messages);
         }).catch(err => {
            resolve({ message: err.message });
         });
   });
}

const dataBase = {
   getSitiesAndDepartments: getSitiesAndDepartments,
   getMaxCodePatient: getMaxCodePatient,
   GetParameters: GetParameters,
   FindParametersFromArray: FindParametersFromArray,
   getRoles:getRoles,
   getMessages:getMessages
};

module.exports = dataBase;
```

index.js

```
const verifySignUp = require("./verifySignUp");
const dataBase = require("./dataBase");

module.exports = {
   verifySignUp,
   dataBase
};
```

verifySignUp.js

```
const db = require("../models");
const ROLES = db.ROLES;
```

```
const User = db.user;

function checkDuplicateUsernameOrEmail(req, res, next) {
  User.findOne({
    where: {
      email: req.body.email
    }
  }).then(user => {
    if (user) {
      res.status(400).send({
        message: "Failed! Email is already in use!"
      });
      return;
    }

    next();
  });
};

function checkRolesExisted(req, res, next) {
  if (req.body.roles) {
    if (!ROLES.includes(req.body.rolename)) {
      res.status(400).send({
        message: "Failed! Role does not exist = " + req.body.rolename
      });
      return;
    }
  }

  next();
};

function verifyToken(req, res, next) {
  let token = req.headers["x-access-token"];
```

```
    if (!token) {
        return res.status(403).send({
            message: "No token provided!"
        });
    }

    jwt.verify(token, config.secret, (err, decoded) => {
        if (err) {
            if (err.message == 'jwt expired') {
                return res.status(401).send("Time working token lost!");
            }
            return res.status(401).send("Unauthorized!");
        }
        req.userId = decoded.id;
        next();
    });
};

function isAdmin(req, res, next) {
    User.findByPk(req.userId).then(user => {
        user.getRoles().then(roles => {
            for (let i = 0; i < roles.length; i++) {
                if (roles[i].name === "Адміністратор") {
                    next();
                    return;
                }
            }
            res.status(403).send({
                message: "Require Admin Role!"
            });
            return;
        });
    });
```

```
};

function isUser(req, res, next) {
   User.findByPk(req.userId).then(user => {
      if(user){
         next();
      }else{
         if (!isUser) {
            res.status(403).send("Require User Role!");
            return;
         }
      }
   });
};


const verifySignUp = {
   checkDuplicateUsernameOrEmail: checkDuplicateUsernameOrEmail,
   checkRolesExisted: checkRolesExisted,
   verifyToken:verifyToken,
   isAdmin:isAdmin,
   isUser:isUser
};

module.exports = verifySignUp;
```

department.model.js

```
module.exports = (sequelize, Sequelize) => {
   const Department = sequelize.define("department", {
      id: {
         type: Sequelize.UUID,
         defaultValue: Sequelize.UUIDV1,
         primaryKey: true
      },
```

```
        name: {
            type: Sequelize.STRING
        }
    });

    return Department;
};
```

history.model.js

```
module.exports = (sequelize, Sequelize) => {
    const History = sequelize.define("history", {
        id: {
            type: Sequelize.UUID,
            defaultValue: Sequelize.UUIDV1,
            primaryKey: true
        },
        date: {
            type: Sequelize.DATEONLY
        },
        durationTreatment: {
            type: Sequelize.INTEGER
        },
        numberOfParts: {
            type: Sequelize.INTEGER
        }
    });

    return History;
};
```

index.js

```
const config = require("../config/config.js");
var bcrypt = require("bcryptjs");
const Sequelize = require("sequelize");
```

```
const sequelize = new Sequelize(
    config.DB,
    config.USER,
    config.PASSWORD,
    {
        host: config.HOST,
        dialect: config.dialect,
        operatorsAliases: false,

        pool: {
            max: config.pool.max,
            min: config.pool.min,
            acquire: config.pool.acquire,
            idle: config.pool.idle
        }
    }
);

const db = {};

db.Sequelize = Sequelize;
db.sequelize = sequelize;

db.user = require("../models/user.model.js")(sequelize, Sequelize);
db.role = require("../models/role.model.js")(sequelize, Sequelize);

db.history = require("../models/history.model.js")(sequelize, Sequelize);
db.department = require("../models/department.model.js")(sequelize, Sequelize);
db.patient = require("../models/patient.model.js")(sequelize, Sequelize);
db.sity = require("../models/sity.model.js")(sequelize, Sequelize);
db.message = require("../models/message.model.js")(sequelize, Sequelize);

db.role.hasMany(db.user);
```

```
db.user.hasMany(db.history);
db.user.hasMany(db.message);
db.sity.hasMany(db.patient);
db.message.belongsTo(db.user, {
   foreignKey: 'userId'
});
db.patient.belongsTo(db.sity, {
   foreignKey: 'sityId'
});

db.history.belongsTo(db.sity, {
   foreignKey: 'sityId'
});

db.history.belongsTo(db.department, {
   foreignKey: 'departmentId'
});

db.user.belongsTo(db.role, {
   foreignKey: 'roleId'
});

db.patient.hasMany(db.history);
db.sity.hasMany(db.history);
db.department.hasMany(db.history);
db.department.hasMany(db.patient);

db.ROLE = ["Сестра медична лазерної терапії", "Сестра процедур з парафіном", "Масажист"
, "Старша сестра"];
var fs = require('fs');
var path = require('path');
var data = fs.readFileSync(path.join(__dirname, '../data/Departaments.txt'), 'utf8');
var departaments = data.split(",");
db.DEPARTAMENT = departaments;
```

```
db.SITY = ["СУМИ", "КОНОТОП", "ШОСТКА", "ОХТИРКА", "РОМНИ", "ГЛУХІВ", "ЛЕБЕДИН", "КРОЛЕВЕЦЬ", "БУРИНЬ", "ПУТИВЛЬ", "ВОРОЖБА", "СЕРЕДИНА-БУДА"];

db.Initial = function (roles, Role, User) {
  roles.forEach(element => {
    Role.create({
      name: element
    });
  });
  Role.create({
    name: "Аміністратор"
  }).then(role => {
    User.create({
      username: "Admin",
      email: "admin@gmail.com",
      password: bcrypt.hashSync("admin", 8),
      roleId: role.id
    });
  });
  db.DEPARTAMENT.forEach(element => {
    db.department.create({
      name: element
    });
  });
  db.SITY.forEach(element => {
    db.sity.create({
      name: element
    });
  });
};

module.exports = db;
```

message.model.js

```
module.exports = (sequelize, Sequelize) => {
    const Message = sequelize.define("message", {
        id: {
            type: Sequelize.UUID,
            defaultValue: Sequelize.UUIDV1,
            primaryKey: true
        },
        description: {
            type: Sequelize.STRING
        },
        priority: {
            type: Sequelize.INTEGER
        }
    });

    return Message;
};
```

patient.model.js

```
module.exports = (sequelize, Sequelize) => {
    const Patient = sequelize.define("patient", {
        id: {
            type: Sequelize.UUID,
            defaultValue: Sequelize.UUIDV1,
            primaryKey: true
        },
        name: {
            type: Sequelize.STRING
        },
        date: {
            type: Sequelize.DATEONLY
        },
        code: {
            type: Sequelize.INTEGER,
```

```
            unique: true
        }
    });

    return Patient;
};
```

role.model.js

```
module.exports = (sequelize, Sequelize) => {
    const Role = sequelize.define("role", {
        id: {
            type: Sequelize.UUID,
            defaultValue: Sequelize.UUIDV1,
            primaryKey: true
        },
        name: {
            type: Sequelize.STRING
        }
    });

    return Role;
};
```

sity.model.js

```
module.exports = (sequelize, Sequelize) => {
    const Sity = sequelize.define("sity", {
        id: {
            type: Sequelize.UUID,
            defaultValue: Sequelize.UUIDV1,
            primaryKey: true
        },
        name: {
            type: Sequelize.STRING
        }
```

```
    });
    return Sity;
};
```

user.model.js

```
module.exports = (sequelize, Sequelize) => {
    const User = sequelize.define("user", {
        id: {
            type: Sequelize.UUID,
            defaultValue: Sequelize.UUIDV1,
            primaryKey: true
        },
        username: {
            type: Sequelize.STRING
        },
        email: {
            type: Sequelize.STRING
        },
        password: {
            type: Sequelize.STRING
        }
    });

    return User;
};
```

admin.routers.js

```
const { verifySignUp } = require("../middleware");
const controller = require("../controllers/admin.controller");
const { dataBase } = require("../middleware");

module.exports = function (app) {
    app.post(
        "/admin/createUser",
```

```
    [/*verifySignUp.verifyToken,
    verifySignUp.isAdmin*/],
    controller.createUser
);

app.get('/admin/createUser',
    [/*verifySignUp.verifyToken,
    verifySignUp.isAdmin*/],
    (req, res) => {
        dataBase.getRoles().then(result => {
            res.render("createUser.hbs", {
                roles: result,
                layout: 'layoutAdmin.hbs'
            });
        });
    });

app.get('/admin/technicaSupport',
    [/*verifySignUp.verifyToken,
    verifySignUp.isAdmin*/],
    (req, res) => {
        dataBase.getMessages().then(result => {
            res.render("technicaSupport.hbs", {
                messages: result,
                layout: 'layoutAdmin.hbs'
            });
        });
    });

app.post(
    "/admin/getAllMessage",
    [/*verifySignUp.verifyToken,
    verifySignUp.isAdmin*/],
    controller.getAllMessage
```

```
  );

  app.post(
    "/admin/deleteMessage",
    [/*verifySignUp.verifyToken,
    verifySignUp.isAdmin*/],
    controller.deleteMessage
  );
};
```

auth.routes.js

```
const { verifySignUp } = require("../middleware");
const controller = require("../controllers/auth.controller");

module.exports = function (app) {
  app.use(function (req, res, next) {
    res.header(
      "Access-Control-Allow-Headers",
      "x-access-token, Origin, Content-Type, Accept"
    );
    next();
  });

  app.get('/login', (req, res) => {
    res.render("singnIn.hbs", { layout: 'layoutLogin.hbs' });
   });

  app.post(
    "/register",
    [
      verifySignUp.checkDuplicateUsernameOrEmail,
      verifySignUp.checkRolesExisted
    ],
    controller.signup
```

```
  );

  app.post("/login", controller.signin);
};
```

sister.routes.js

```
const { verifySignUp } = require("../middleware");
const controller = require("../controllers/auth.controller");
const sistercontroller = require("../controllers/sister.controller");
const { dataBase } = require("../middleware");

module.exports = function (app) {
  app.use(function (req, res, next) {
    res.header(
      "Access-Control-Allow-Headers",
      "x-access-token, Origin, Content-Type, Accept"
    );
    next();
  });

  app.post("/setPatient",
    [/*verifySignUp.verifyToken,
    verifySignUp.isUser*/],
    sistercontroller.setPatient);

  app.post('/getPatient',
    [/*verifySignUp.verifyToken,
    verifySignUp.isUser*/],
    sistercontroller.getPatient);

  app.post('/setProcedure',
    [/*verifySignUp.verifyToken,
    verifySignUp.isUser*/],
    sistercontroller.setProcedure);
```

```
app.post('/getHistories',
    [/*verifySignUp.verifyToken,
    verifySignUp.isUser*/],
    sistercontroller.getHistories);

app.post('/sentMessage',
    [/*verifySignUp.verifyToken,
    verifySignUp.isUser*/],
    sistercontroller.sentMessage);

app.post('/getMaxCodePatient',
    [/*verifySignUp.verifyToken,
    verifySignUp.isUser*/],
    (req, res) => {
        dataBase.getMaxCodePatient().then(result => {
            res.send(" + result);
        });
    });

app.get('/createPatient',
    [/*verifySignUp.verifyToken,
    verifySignUp.isUser*/],
    (req, res) => {
        dataBase.getSitiesAndDepartments().then(result => {
            res.render("createPatient.hbs", {
                sitiesName: result.sitiesName,
                departmentsName: result.departmentsName
            });
        });
    });

app.get('/createProcedure',
    [/*verifySignUp.verifyToken,
```

```
        verifySignUp.isUser*/],
        (req, res) => {
            dataBase.getSitiesAndDepartments().then(result => {
                res.render("createProcedure.hbs", {
                    sitiesName: result.sitiesName,
                    departmentsName: result.departmentsName
                });
            });
        });

    app.get('/history',
        [/*verifySignUp.verifyToken,
        verifySignUp.isUser*/],
        (req, res) => {
            res.render("history.hbs");
        });

    app.get('/help',
        [/*verifySignUp.verifyToken,
        verifySignUp.isUser*/],
        (req, res) => {
            res.render("help.hbs");
        });

};
```

styles.css

```
.vertical-center {
    min-height: 100%;  /* Fallback for browsers do NOT support vh unit */
    min-height: 100vh; /* These two lines are counted as one :-)       */

    display: flex;
    align-items: center;
  }
```

```
.center {
  display: flex;
  justify-content: center;
  align-items: center;
}

  .navbarMain {
    padding: .5rem 1rem;
  }

.text-center {
  text-align: center;
}

.mright10 {
  margin-right: 10px;
}

.menu{
  background-color: floralwhite;
}
```

layoutAdmin.hbs

```
<!DOCTYPE html>
<html>

<head>
    <title>{{title}}</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="css/styles.css">
    <script src="/admin/js/project.js"></script>
    <script src="/js/handlebars.js"></script>
    <link href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet"
id="bootstrap-css">
```

```
    <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <script src="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
    <script src="http://code.jquery.com/ui/1.11.0/jquery-ui.js"></script>
</head>

<body style="background-color: floralwhite;">
    {{> menuAdmin}}

    {{{body}}}

</body>
<html>
```

layoutLogin.hbs

```
<!DOCTYPE html>
<html>

<head>
    <title>{{title}}</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="css/styles.css">
    <script src="js/login.js"></script>
    <link href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet"
id="bootstrap-css">
    <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <script src="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
</head>

<body>

    {{{body}}}

</body>
```

```
<html>
```

layoutUser.hbs

```
<!DOCTYPE html>
<html>

<head>
    <title>{{title}}</title>
    <meta charset="utf-8" />
    <link rel="stylesheet" href="/css/styles.css">
    <script src="/js/handlebars.js"></script>
    <script src="/js/project.js"></script>
    <link href="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet"
id="bootstrap-css">
    <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <script src="//maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
    <script src="http://code.jquery.com/ui/1.11.0/jquery-ui.js"></script>
</head>

<body style="background-color: floralwhite;">
    {{> menuUser}}

    {{{body}}}

</body>
<html>
```

menuAdmin.hbs

```
<div class="navbarMain text-center navbar-light bg-secondary">
  <h1>Електронний журнал</h1>
  <h3>Сумська область клінічна лікарня</h3>
</div>
<nav class="navbar navbar-expand-lg navbar-light bg-info">
```

```
    <ul class="nav navbar-nav mx-auto">
      <li class="nav-item">
        <h3>Адмін</h3>
      </li>
    </ul>
  </nav>
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand"></a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"
      aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="nav navbar-nav mx-auto">
        <li class="nav-item">
          <a class="nav-link" href="/admin/createUser">Створити користувача<span class="sr-
only">(current)</span></a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="/admin/technicaSupport">Техпідтримка</a>
        </li>
      </ul>
    </div>
  </nav>
```

menuUser.hbs

```
  <div class="navbarMain text-center navbar-light bg-secondary">
    <h1>Електронний журнал</h1>
    <h3>Сумська область клінічна лікарня</h3>
  </div>
  <nav class="navbar navbar-expand-lg navbar-light bg-info">
```

```
        <ul class="nav navbar-nav mx-auto">
          <li class="nav-item">
            <h3>Сестра лазерної терапії</h3>
          </li>
        </ul>
      </nav>
      <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <a class="navbar-brand"></a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"
          aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarSupportedContent">
          <ul class="nav navbar-nav mx-auto">
            <li class="nav-item">
              <a class="nav-link" href="/createPatient">Створити картку пацієнта <span class="sr-
only">(current)</span></a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="/createProcedure">Запис процедури</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="/history">Переглянути денний звіт</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="/help">Написати до техпідтримки</a>
            </li>
          </ul>
        </div>
      </nav>
```

createPatient.hbs

```
<div class="menu">
   <h2 class="text-center">Створення картки пацієнта</h2>
   <form onsubmit="createPatient(this,'message');return false" action="">
      <div class="table-responsive">
         <table class="table table-bordered table-striped table-highlight">
            <thead class="text-center">
               <th>ПІБ</th>
               <th>Код пацієнта</th>
               <th>Відділення</th>
               <th>Дата народження</th>
               <th>Область</th>
            </thead>
            <tbody>
               <tr>
                  <td><input type="text" class="form-control" id="username" required /></td>
                  <td><input type="text" class="form-control" id="patientCode" readonly /></td>
                  <td><select name="departament" id="departament" class="form-control">
                        {{#each departmentsName}}
                        <option value="{{this.id}}">{{this.name}}</option>
                        {{/each}}
                     </select>
                  </td>
                  <td><input type="date" id="day" class="form-control" required /></td>
                  <td><select name="sity" id="sity" class="form-control">
                        {{#each sitiesName}}
                        <option value="{{this.id}}">{{this.name}}</option>
                        {{/each}}
                     </select>
                  </td>
               </tr>
            </tbody>
         </table>
      </div>
```

```
        <div class="text-right">
          <button type="submit" class="btn btn-success mright10">Завантажити
            картку</button>
        </div>
        <div class="alert alert-success" id="message" role="alert" hidden>
          Збережено
        </div>
      </form>
    </div>

    <script>
      window.onload = function () {
        document.getElementById('day').valueAsDate = new Date();
        setMaxCodePatient('patientCode');
      }
    </script>
```

createProcedure.hbs

```
    <div class="menu">
      <h2 class="text-center">Запис процедури</h2>
      <form id="ProsedureForm"  onsubmit="createProcedure(this,'message');return false" action=""
>
        <div class="table-responsive">
          <table class="table table-bordered table-striped table-highlight">
            <thead class="text-center">
              <th>Код пацієнта</th>
              <th style="max-width: 100px;">Дата</th>
              <th style="max-width: 100px;">Відділення</th>
              <th style="max-width: 60px;">Кількість полів</th>
              <th style="max-width: 60px;">Тривалість лікування</th>
            </thead>
            <tbody>
              <tr>
```

```
                    <td><input type="text" class="form-
control" id="code" onfocusout="FindPatient(this,'message','ProsedureForm')" required /></td>
                    <td style="max-width: 100px;"><input type="date" id="day" class="form-
control" required />
                    </td>
                    <td style="max-
width: 100px;"><select name="departament" id="departament" class="form-control">
                        {{#each departmentsName}}
                        <option value="{{this.id}}">{{this.name}}</option>
                        {{/each}}
                      </select>
                    </td>
                    <td style="max-width: 60px;"><select name="Number" id="numberPart"
                        class="form-control"></select></td>
                    <td style="max-width: 60px;"><select name="Time" id="time" class="form-
control"></select></td>
                  </tr>
                  <tr>
                    <td>ПІБ:<input type="text" class="form-control" id="userId" style="pointer-
events: none;" hidden/>
                      <input type="text" class="form-control" id="username" style="pointer-
events: none;" required/></td>
                  </tr>
                  <tr>
                    <td>Область:<option value="" id="sity"></option></td>
                  </tr>
                </tbody>
              </table>
            </div>
            <div class="text-right">
              <button type="submit" class="btn btn-success mright10">Зберегти</button>
            </div>
            <div class="alert alert-success" id="message" role="alert" hidden>
              Збережено
```

```
            </div>
        </form>
    </div>

    <script>
        window.onload = function () {
            document.getElementById('day').valueAsDate = new Date();
            document.getElementById("day").setAttribute("max", new Date().toISOString().split("T")[0]
);
            var min = 1,
                max = 10,
                select = document.getElementById('numberPart');

            for (var i = min; i <= max; i++) {
                var opt = document.createElement('option');
                opt.value = i;
                opt.innerHTML = i;
                select.appendChild(opt);
            }

            var min = 3,
                max = 30,
                select = document.getElementById('time');

            for (var i = min; i <= max; i++) {
                var opt = document.createElement('option');
                opt.value = i;
                opt.innerHTML = i;
                select.appendChild(opt);
            }
        }
    </script>
```

createUser.hbs

```
<div class="menu">
  <h2 class="text-center">Створення користувача</h2>
  <form onsubmit="createUser(this,'message');return false" action="">
    <div class="table-responsive">
      <table class="table table-bordered table-striped table-highlight">
        <thead class="text-center">
          <th>ПІП</th>
          <th>Роль</th>
          <th>Логін</th>
          <th>Паролі</th>
        </thead>
        <tbody>
          <tr>
            <td><input type="text" class="form-control" id="username" required /></td>
            <td><select name="roleId" id="roleId" class="form-control">
                {{#each roles}}
                <option value="{{this.id}}">{{this.name}}</option>
                {{/each}}
              </select>
            </td>
            <td><input type="text" class="form-control" id="email" required /></td>
            <td><input type="text" class="form-control" id="password" required /></td>
          </tr>
        </tbody>
      </table>
    </div>
    <div class="text-right">
      <button type="submit" class="btn btn-
success mright10">Створити користувача</button>
    </div>
    <div class="alert alert-success" id="message" role="alert" hidden>
      Збережено
    </div>
  </form>
```

```
</div>
```

help.hbs

```
<div class="menu">
    <h2 class="text-center mb-3">Оформлення Заявки до тех підтримки</h2>
    <div class="center mb-3">
        sdasdsds<br>
        dsad<br>
        das
    </div>
    <form action="" onsubmit="sentMessage(this,'message');return false">
        <div class="center mb-3">
            <textarea style="min-
width: 80%" id="description" name="description" rows="5" cols="33" required></textarea>
        </div>
        <div class="center mb-3">
            Оберфть пріоритет проблеми:
        </div>
        <div class="center mb-3">
            <select name="select" id="priority" style="min-width: 150px;  text-align-last:center;">
                <option value="1" selected>Високий</option>
                <option value="2">Средний</option>
                <option value="3">Низький</option>
            </select>
        </div>
        <div class="center mb-3">
            <button type="submit" class="btn btn-success mright10">Відправити</button>
        </div>
        <div class="alert alert-success" id="message" role="alert" hidden>
            Збережено
        </div>
    </form>
</div>
```

history.hbs

```
<div class="menu">
  <h2 class="text-center">Звіт</h2>
  <form action="" onsubmit="getHistories(this,'message');return false">
    <div class="input-group">
      <div class="input-group-prepend">
        <span class="input-group-text" id="">Start and End date</span>
      </div>
      <input style="width: 160px;" type="date" id="dateStart" onclick="SetMaxdate('dateEnd',t
his)"
        class="form-control" required />
      <input style="width: 160px;" type="date" id="dateEnd" onclick="SetMindate('dateStart',th
is)"
        class="form-control mright10" required />
      <button type="submit" class="btn btn-primary mright10">Побудувати</button>
      <div class="input-group-prepend">
        <div class="input-group-text">
          <span class="input-group-text mright10" name="contact" id="1">
            <input type="radio" name="radio" id="radio1"
              onclick="CheckRadio('radio1','Text',['Область','Відділення']);" checked>
            По областям
          </span>
          <span class="input-group-text" name="contact" id="2">
            <input type="radio" name="radio" id="radio2"
              onclick="CheckRadio('radio1','Text',['Область','Відділення']);">
            По відділенням
          </span>
        </div>
      </div>
    </div>
  </form>
  <div class="table-responsive">
    <table class="table table-bordered table-striped table-highlight">
      <thead class="text-center">
```

```
                <th style="max-width: 50px;">№</th>
                <th style="max-width: 100px;" id="Text">Область</th>
                <th>Лікувалось в ФРМ</th>
                <th>Всього процедур</th>
            </thead>
            <tbody id="bodyId">
            </tbody>
        </table>
    </div>
    <form action="">
        <div class="text-right">
            <button type="submit" class="btn btn-success mright10">Завантажити</button>
        </div>
    </form>
    <div class="alert alert-success" id="message" role="alert" hidden>
        Збережено
    </div>
</div>

<script id="List-history" type="text/x-handlebars-template">
    \{{#each this}}
        <tr>
            <td style="max-width: 50px;">\{{@index}}</td>
            <td style="max-width: 100px;">\{{this.criteriaName}}</td>
            <td>\{{this.numberOfParts}}</td>
            <td>\{{this.durationTreatment}}</td>
        </tr>
    \{{/each}}
</script>

<script>
    window.onload = function () {
        document.getElementById('dateStart').valueAsDate = new Date();
        document.getElementById('dateEnd').valueAsDate = new Date();
```

```
            document.getElementById("dateStart").setAttribute("max", new Date().toISOString().split("
T")[0]);
            document.getElementById("dateEnd").setAttribute("max", new Date().toISOString().split("T
")[0]);
        }
    </script>
```

singnIn.hbs

```
    <div class="vertical-center">
        <aside class="col-md-5 mx-auto">
            <div class="card">
                <article class="card-body">
                    <h4 class="card-title mb-4 mt-1">Sign in</h4>
                    <form name="user_form">
                        <div class="form-group">
                            <label>Your email</label>
                            <input name="email" id="email" class="form-
control" placeholder="Email" type="email">
                        </div>
                        <div class="form-group">
                            <label>Your password</label>
                            <input name="password" id="password" class="form-
control" placeholder="******" type="password">
                        </div>
                        <div class="form-group">
                            <div class="alert alert-danger" id="message" role="alert" hidden></div>
                        </div>
                        <div class="form-group">
                            <button type="submit" class="btn btn-primary btn-
block" onclick="return LoginClick()"> Login
                            </button>
                        </div>
                        <div class="alert alert-danger" role="alert">
                            Пароль невірний
```

```
                </div>
            </form>
        </article>
    </div>
</aside>
</div>
```

technicaSupport.hbs

```
<div class="menu">
    <h2 class="text-center">Повідомлення</h2>
    <div class="table-responsive">
        <table class="table table-bordered table-striped table-highlight">
            <thead class="text-center">
                <th style="width: 350px;">ПІБ сестри</th>
                <th style="width: 150px;">Роль</th>
                <th style="width: 50px;">Пріоритет</th>
                <th>Опис</th>
                <th style="width: 50px;">Дія</th>
            </thead>
            <tbody id="bodyId">
                {{#each messages}}
                <tr>
                    <td style="width: 350px;">{{ "user.username" }}</td>
                    <td style="width: 150px;">{{ "user.role.name" }}</td>
                    <td style="width: 50px;">{{ priority }}</td>
                    <td>{{ description }}</td>
                    <td style="width: 50px;"><button type="button" class="btn btn-
danger" onclick="deleteMessage('{{ id }}')">Видалити</button></td>
                </tr>
                {{/each}}
            </tbody>
        </table>
    </div>
</div>
```

```
<script id="List-history" type="text/x-handlebars-template">
  \{{#each this}}
    <tr>
      <td style="width: 350px;">\{{ "user.username" }}</td>
      <td style="width: 150px;">\{{ "user.role.name" }}</td>
      <td style="width: 50px;">\{{ priority }}</td>
      <td>\{{ description }}</td>
      <td style="width: 50px;"><button type="button" class="btn btn-
danger" onclick="deleteMessage('\{{ id }}')">Видалити</button></td>
    </tr>
  \{{/each}}
</script>
```

app.js

```
const express = require('express');
const app = express();
const expressHbs = require("express-handlebars");
const hbs = require("hbs");
var path = require('path');
const bodyParser = require("body-parser");
app.engine("hbs", expressHbs(
 {
  layoutsDir: "views/layouts",
  defaultLayout: "layoutUser",
  extname: "hbs"
 }
));
app.set("view engine", "hbs");
app.use(express.static(__dirname + '/views'));
hbs.registerPartials(__dirname + "/views/partials");

app.use(bodyParser.json());
```

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

require('./app/routers/auth.routes')(app);
require('./app/routers/sister.routes')(app);
require('./app/routers/admin.routers')(app);
const port = 3000;

const db = require("./app/models");

db.sequelize.sync({ force: false }).then(() => {
  console.log('Drop and Resync Db');
  //db.Initial(db.ROLE, db.role, db.user);
});

app.post('/', (req, res) => {
  res.send('');
});

app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`);
});
```