Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:
Розробка автоматизованої системи бюро судово-медичної експертизи

Виконав: студент 2 курсу, групи 2АКТТП-22м
спеціальності 151 – Автоматизація
та комп'ютерно-інтегровані технології

Олександр ПОБЕРЕЖНЯК
Ім'я ПРІЗВИЩЕ

Керівник: к.т.н., доцент, доцент кафедри КСУ
степінь, звання, посада

Олег КОВАЛЮК
Ім'я ПРІЗВИЩЕ

«1» 12 2023 р.

Опонент: к.т.н., доцент, доцент каферди КСУ
степінь, звання, посада

Ярослав КУЛИК
Ім'я ПРІЗВИЩЕ

«5» 12 2023 р.

Допущено до захисту
Т.в.о.Зав. кафедри КСУ
_Марія ЮХИМЧУК
«7» 12 2023

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти другий (магістерський)
Галузь знань – 15 – Автоматизація та приладобудування
Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо - професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Т.в.о.Зав. кафедри КСУ

Марія ЮХИМЧУК

"09" жовтня 2023 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Побережняку Олександру Романовичу.
(прізвище, ім'я, по батькові)

1. Тема роботи. Розробка автоматизованої системи бюро судово-медичної експертизи
керівник роботи Ковалюк Олег Олександрович
затверджені наказом ВНТУ від "18" вересня 2023 року №247
2. Термін подання студентом роботи "1" грудня 2023 року
3. Вихідні дані до роботи: можливість авторизації в особистий кабінет експерта або завідуючого відділом; можливість відправки та отримання направлень між судово-медичними експертами; можливість надавати відповідь на отримане направлення з прикріпленим до відповіді файлом-документом; можливість завідуючого відділом призначити підлеглому судово-медичному експерту направлення, що отримане в відділ; можливість перегляду сторінок з публічною інформацією для неавторизованих користувачів.
4. Зміст текстової частини: вступ, аналіз автоматизованих систем підприємств й установ та огляд аналогів, проєктування автоматизованої системи бюро судово-медичної експертизи, розробка та тестування автоматизованої системи
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): аналіз схожих автоматизованих систем, вибір технологій, uml-діаграма варіантів використання, діаграма бази данних для типів колекцій неавторизованого користувача, діаграма бази данних для типів колекцій авторизованого користувача, вигляд навігаційного меню сторінок для неавторизованого користувача з відкритою вкладкою «воб сме», вигляд головної сторінки для неавторизованого користувача, вигляд компоненту перегляду pdf-документів для неавторизованого користувача, вигляд кабінету судово-медичного експерта, вигляд кабінету завідуючого відділом/відділенням, вигляд модального вікна отриманого направлення в залежності від стану направлення, вигляд модального вікна створення направлення, висновки.

1. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
4	Бурєннікова Н.В. д.е.н., професор кафедри ЕПВМ		

1. Дата видачі завдання "09" жовтня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк виконання етапів роботи	Примітка
1	Аналіз автоматизованих систем підприємств й установ та огляд аналогів	28.09.2023 р.	
2	Розрбка технічного завдання, аналіз і вибір технологій	15. 10.2023 р.	
3	Проєктування автоматизованої системи бюро судово-медичної експертизи	25. 10.2023 р.	
4	Побудова логіки серверної частини системи	29. 10.2023 р.	
5	Розробка Front-end частини системи	10.11.2023 р.	
6	Тестування роботи системи	20.11.2023 р.	
7	Розробка економічної частини	25.11.2023 р.	
8	Оформлення пояснювальної записки і графічного матеріалу	10. 12.2023 р.	
9	Попередній захист	14.12.2023 р.	
10	Остаточний захист	19.12.2023 р.	

Студент _____ Олександр ПОБЕРЕЖНЯК
(підпис) (*Ім'я ПРІЗВИЩЕ*)

Керівник роботи _____ Олег КОВАЛЮК
(підпис) (*Ім'я ПРІЗВИЩЕ*)

АНОТАЦІЯ

УДК 681.5.004.41

Побережняк О. Р. Розробка автоматизованої системи бюро судово-медичної експертизи. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп’ютерно-інтегровані технології, освітня програма – Інтелектуальні комп’ютерні системи. Вінниця: ВНТУ, 2023. 139 с.

На укр. мові. Бібліогр.: 66 назв; рис.: 63; табл. 11.

У магістерській кваліфікаційній роботі розроблено автоматизовану бюро судово-медичної експертизи. У оглядово-аналітичній частині роботи проведено аналіз предметної області розробки систем підприємств та установ, проведено аналіз предметної області розробки автоматизованої системи бюро судово-медичної експертизи, оглянуті та проаналізовані аналогічні системи, проведений аналіз та вибір технологій для вирішення проблеми. У практичній частині побудована серверна логіка системи, розроблені інтерфейси користувачів, наведено результати тестування доступності бази даних. У економічній частині проаналізований технічний рівень і розрахована собівартість реалізації розробки. Ілюстративна частина складається з 13 плакатів із результатами роботи.

Ключові слова: автоматизована система, система керування вмістом, направлення, судово-медичний експерт, компонент, функція.

ANNOTATION

UDC 681.5.004.41

Poberezhniak O. R. Development of an automated system of the bureau of forensic medical examination. Master's qualification work in specialty 151 - Automation and computer-integrated technologies, educational program - Intelligent computer systems. Vinnitsa: VNTU, 2023. 139 p.

In Ukrainian. Bibliogram.: 66 titles; Figure: 63; Table 11.

In the master's qualification work, an automated bureau of forensic medical examination was developed. In the review and analytical part of the work, an analysis of the subject area of the development of systems of enterprises and institutions was carried out, an analysis of the subject area of the development of an automated system of the bureau of forensic medical examination was carried out, similar systems were examined and analyzed, an analysis was carried out and technologies were chosen to solve the problem. In the practical part, the server logic of the system is built, user interfaces are developed, the results of testing the availability of the data base are given. The economic part analyzed the technical level and calculated the cost of development. The illustrative part consists of 13 posters with the results of the work.

Keywords: automated system, content management system, referral, forensic expert, component, function

ЗМІСТ

ВСТУП

Науково-технічний прогрес, що стрімко почав набирати оберти в другій половині XX століття, кардинально змінив підхід до обробки, зберігання та передачі інформації. Поява комп'ютерів, Інтернету та інших інформаційних технологій кардинально перетворила парадигму відкритості державних процесів і доступу населення до даних. У сфері звітності про закупівлі ця революція розширила можливості збору та аналізу інформації про витрати державних структур коштів на своє існування, забезпечивши більшу прозорість і доступність інформації.

Сучасна ділова та правова практика диктує необхідність ефективного використання інформаційних технологій для поліпшення функціонування різноманітних сфер діяльності. У цьому контексті, створення та оптимізація веб-сайтів стають ключовими завданнями для підтримки та розвитку різних організацій. Однією з таких важливих сфер є бюро судово-медичної експертизи, яка забезпечує важливі послуги у сфері судової медицини та експертизи.

У цьому дослідженні буде розглянуто важливі аспекти розробки та функціональності веб-сайту бюро судово-медичної експертизи, а також вивчимо переваги та можливості, які він надасть організації та її клієнтам у сфері судової медицини та експертизи. Додатково, буде розглянуто вплив цього проекту на підвищення рівня доступності та зручності для клієнтів, що шукають інформацію про бюро судово-медичної експертизи та його послуги. Веб-сайт створить можливість для онлайн-звернень, що покращить комунікацію між організацією та її клієнтами та сприятиме загальній ефективності роботи бюро.

Актуальність даної магістерської роботи виразно відображається в контексті сучасного суспільства та розвитку інформаційних технологій. Нижче наведено декілька ключових аргументів, що підкреслюють актуальність роботи з розробки веб-сайту для бюро судово-медичної експертизи:

1) Цифрова трансформація у сфері медицини та юстиції: Сучасні організації, які займаються судовою медициною та експертизою, вивчають можливості цифрової трансформації для поліпшення своєї роботи. Розробка веб-сайту для бюро судово-

медичної експертизи є важливим кроком у цьому напрямку.

2) Підвищення прозорості та доступності інформації: Веб-сайт дозволить організації надавати доступну та зрозумілу інформацію про свою діяльність, напрямки роботи та фінансовий стан. Це сприятиме підвищенню прозорості та довіри до бюро судово-медичної експертизи.

3) Зручність для клієнтів: Веб-сайт надасть можливість клієнтам швидко знаходити необхідну інформацію, звертатися до бюро та використовувати онлайн-звернення. Це зробить процес співпраці з бюро судово-медичної експертизи більш зручним та ефективним.

4) Глобальні тенденції в е-управлінні та е-сервісах: Розвиток інформаційних технологій перетворює сферу управління та надання послуг. Робота над веб-сайтом для бюро судово-медичної експертизи вписується в загальний контекст глобальних тенденцій в е-управлінні та е-сервісах.

Отже, дана магістерська робота має велике практичне значення та актуальність, оскільки вона спрямована на вирішення актуальних завдань в галузі судової медицини та експертизи, а також відповідає потребам сучасного інформаційного суспільства.

Метою дослідження є поліпшення якості обслуговування, підвищення та покращення умов подачі прозорої звітності, документообігу.

Об’єктом дослідження є організаційні процеси у бюро судово-медичної експертизи, що є державною організацією, та повинна подавати прозору звітність по витраченим коштам, проведеним роботам, напрямкам своєї діяльності, структурі закладу відповідно до чинного законодавства України.

Предметом дослідження є процес обміну документацією між судово-медичними експертами та впровадження автоматизованої системи бюро судово-медичної експертизи.

Новизна (Інноваційність) роботи полягає в комплексному підході до вирішення задач подачі прозорої звітності, відповідно до вимог Господарського кодексу України, побудові сучасного веб-сайту для організації, поліпшення цифровізації хоч невеликої, але частини державного апарату, задання вектору руху

та розвитку в цифровій сфері для структур схожого типу.

Практична цінність полягає в розробці програмного забезпечення та алгоритмів роботи системи бюро судово-медичної експертизи, полегшення подання звітності до компетентних органів перевірки.Також, практичною цінністю є зручний обмін документацією та керування завданнями між відділеннями установи.

Апробація. Представлені в роботі результати апробовані в результаті участі в конференції Всеукраїнська науково-практична Інтернет-конференція студентів, аспірантів та молодих науковців «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ (МН-2023)».

Публікації: Побережняк О.Р., Ковалюк О.О. «Розробка автоматизованої системи бюро судово-медичної експертизи», «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ», 2023.

1. АНАЛІЗ АВТОМАТИЗОВАНИХ СИСТЕМ ПІДПРИЄМСТВ Й УСТАНОВ ТА ОГЛЯД АНАЛОГІВ

1.1. Аналіз предметної області розробки систем автоматизації роботи підприємств та установ.

На сьогоднішній день кожна державна чи бізнес структура старається стрімко слідувати за напрямком розвитку науково-технічного прогресу, який надає багато різноманітних можливостей для аналітики, тестування, збільшення обсягів охоплення та інших методів ефективного досягнення цілей в цих структурах.
Практично кожна така структура рано чи пізно прийде до висновку, що її функціонування та існування стануть невигідними або навіть неможливими, якщо не вдатись до дій впровадження інтернет-технологій в процес її роботи. Для ведення бізнесу, до прикладу, з продажу товарів, це може бути проблема малого потоку покупців, яка могла виникнути із різноманітних причин, таких як: охоплення аудиторії, якій цей товар не буде цікавий, малий трафік людей біля точки продажу, відсутність або неправильне застосування методів реклами, тощо.

Користуючись тим, що наразі є величезний вибір стеків технологій для ведення онлайн роботи магазину, запуск якого вже нівелює проблему малого трафіку покупців біля фізичної точки продажу, можна вибрати технологію яка підійде конкретно обраному магазину для його цілей та потреб. Наприклад, платформа електронної комерції «Shopify»[1], яка дозволяє розгорнути магазин онлайн, охопити саме тих клієнтів, що будуть зацікавлені в покупці товарів, які реалізує точка продажу, аналізувати весь процес продажу і т.д.

Щодо державних структур, в більшості з них має бути прозора інформація, яку може побачити кожен бажаючий цього громадянин, наприклад для державних закупівель та тендерів на виконання робіт є інтернет-портал «Прозорро»[2], який виконує цю задачу повністю. Із законодавчої бази в Україні є Господарський кодекс України[3], який зобов’язує деякі ланки бізнесу та державного апарату проводити прозору звітність по виконаних роботах, напрямках занять, структурах

організацій, годинам роботи, та іншої інформації для населення.

Також не слід недооцінювати всіх переваг можливостей введення нових технологічних рішень в роботу підприємств чи установ, оскільки ці переваги можуть спростити роботу цього підприємства. Спрощення може полягати в автоматизації якихось дрібних дій, або не тільки дрібних, до прикладу збір аналітичних даних інтернет-магазину. Проаналізувавши дані про онлайн-покупців, їхні відгуки, найчастіші товари в кошиках, можна зробити висновок в тому, на який товар, чи на яку категорію товарів слід зробити акцент. Саме онлайн ведення такого бізнесу автоматизує дію обліку таких даних, оскільки їх легше обробити автоматично, правильно обравши необхідні технології.

Отже, проаналізувавши описані проблеми, можна дати визначення системі автоматизації роботи підприємства чи установи та охарактеризувати її. Система автоматизації роботи підприємства — це комплексна технічно-інформаційна структура, що створюється з метою автоматизації різних робочих чи бізнес-процесів у межах цієї установи або підприємства.

Характеристиками такої системи є:

- Інтеграція з робочими процесами. Система автоматизації об’єднує різні функціональні області підприємства, наприклад: логістика, аналітика, виробництво, облік кадрів, бухгалтерський облік для забезпечення їхньої ефективної взаємодії.
- Оптимізація робочих процесів. Система оптимізує та спрощує шаблонні процеси та завдання, автоматизує роботу з потрібними данними, що дозволяє прискорити виконання робочого процесу, та надає можливість мінімізації помилок.
- Централізоване зберігання та обробка даних. Система забезпечує безпечне зберігання великих обсягів інформації, контролює доступ до цієї інформації, що допомагає отримувати аналітичні дані для прийняття подальших рішень.
- Зменшення ручної праці. Автоматизація забезпечує зменшення використання ручної праці та ручного введення даних, що надає можливість ефективного використання робочого часу кадрового складу та інших

ресурсів, також мінімізує ймовірність виникнення помилок.

- Підвищення зручності та доступності. Впровадження систем автоматизації до робочого процесу, спрощує ці процеси та робить їх більш зручними та доступними для користувачів різних рівнів керування.
- Інтеграція з новітніми технологіями. Системи автоматизації підприємств чи установ можуть включати в себе сучасні технології, до прикладу, аналітика даннних, штучний інтелект, хмарні рішення, що розширить їхній функціонал та ефективність використання.
- Забезпечення безпеки. Система має враховувати та дотримуватись всіх правил кібербезпеки, правил зберігання та використання конфіденційної інформації, тощо.

1.2. Аналіз проблеми для розробки системи автоматизації бюро судово-медичної експертизи.

Працюючи в Вінницькому обласному бюро судово-медичної експертизи на посаді інженера-електроніка, передбачено перелік робочих прав та обов’язків, серед яких є подання щорічної та щоквартальної звітності до інтернет-ресурсів визначених Господарським кодексом України. Також законодавством визначений альтернативний метод подачі таких даних для прозорої звітності організації —це створення веб-сайту структури, куди потрібно вносити всю визначену законодавством звітність, вже не дублюючи її на інтернет-портали визначені законодавством України, на якому буде відображатись ця вся інформація, в режимі відкритого доступу, посилання на яке потрібно передати до відповідних органів контролю.

Також провівши аналіз деяких робочих процесів установи, було прийняте рішення автоматизувати та спростити деякі з них, зокрема: обмін направленнями між експертами та відділеннями бюро судово-медичної експертизи, отримання експертом направлення на виконання судово-медичної експертизи, де адресату та адресанту буде видно статус (етап) виконання цієї експертизи.

Беручи до уваги всебічний розвиток технологій управління процесами, та їх автоматизації, складено список проблем, розкриття яких допоможе поставити задачі для системи автоматизації бюро судово-медичної експертизи:

1) Відсутність сайту державної установи. Вінницьке обласне бюро судово-медичної експертизи не має власного веб-сайту організації, на відміну аналогічних структур, але з інших областей України, які мають веб-сайти з обширною інформацією, такою як список відділів та відділень організації, напрямки їх занять та роботи, платні та безкоштовні послуги які надаються організацією, новини та фотогалерея структури, години роботи, розташування та контакти, інформація для населення, інформація для слідчих, тощо.

2) Неефективний обмін інформацією. Судово-медичні експертизи можуть бути розподілені по різних відділах та відділеннях. Це може призводити до неефективного обміну інформацією між ними та ускладнювати процес прийняття рішень.

3) Складнощі у зберіганні даних. Судово-медичні експертизи генерують велику кількість документів та даних, якими потрібно керувати, та зберігати їх. Наявність ефективної системи для зберігання та пошуку інформації може бути проблемою.

4) Публічний доступ до інформації. Згідно з Господарським кодексом України, бюро судово-медичної експертизи зобов'язане публікувати інформацію про свою діяльність та фінансову звітність. Важливо забезпечити доступність цієї інформації для громадськості.

5) Захист конфіденційної інформації. Оскільки інформація, яка обробляється в бюро судово-медичної експертизи, може бути конфіденційна, необхідно враховувати питання кібербезпеки та захисту даних.

6) Необхідність для покращення доступності та обслуговування клієнтів. Важливо забезпечити зручний інтерфейс та доступність інформації для клієнтів бюро судово-медичної експертизи та слідчих, які шукають необхідну інформацію.

Склавши список проблем, які потрібно вирішити, можна приступити до побудування плану вирішення цих питань. Список проблем розпочинається з проблеми відсутності веб-сайту державної установи Вінницьке обласне бюро

судово-медичної експертизи. Перед тим, як приступити до розробки веб-сайту, потрібно розуміти, яке він буде мати наповнення. Слід визначити, що повинно бути розміщене обов'язково, згідно чинного законодавства України, яким чином це повинно бути розміщене, як буде відбуватись адміністрування сайту, додавання, редагування чи видалення інформації на ньому.

Далі йде проблема неефективного обміну інформацією. Суть проблеми полягає в тому, що судово-медична експертиза, це комплексне рішення проблеми[4], яка бере свій початок від огляду місця події, якщо діло пов'язане з трупом, разом з правоохоронними органами, на якому задача судово-медичного експерта полягає в описі місця події, потерпілого, для подальшого проведення аналізу обставин за яких сталась смерть, що є необхідним для слідства. Також є експертизи потерпілих «відділу живих» де відбувається освідування потерпілого, обвинуваченого та інших осіб. Далі, ці дані направляються в обробку до експерта, у відділ, який спеціалізується випадком, який надійшов до бюро. Експертиза призначаєтсья експерту його безпосереднім начальником. Але один експерт не може повністю дослідити всі об'єкти сам. Тому, структура складається з багатьох відділів та відділень, до прикладу, коли судово-медичний експерт-танатолог проводить розтин в секційній залі, він надиктовує співробітнику лаборанту інформацію, яка буде використана в ході експертизи, для подальшого висновку експерта. Також, якщо це потрібно, експерт-танатолог вилучає потрібні йому об'єкти для дослідження на експертизи імунології, токсикології, гістології, криміналістики. Ці об'єкти поступають в відділення з направленями, результат дослідження яких вплине на точніший аналіз випадку та подальший висновок експерта. Проблема заключається в тому, що в процесі залучено багато ланок, а взаємодія між ними відбувається вербально. Тобто, направлення набираються, та друкуються відповідно до вимог, але доставка цих направлень відбувається фізично від експерта до експерта. Тому, доцільно, запровадити систему обміну цими направленями, та висновками, в якій експертам буде видно, які направлення до них поступили, на якій стадії вони знаходяться і т.д. Ця система обміну також має зберігати документи, в якості архіву, та бути захищеною, оскільки така інформація

може бути конфіденційною.

1.3. Огляд та аналіз аналогічних та схожих систем.

Перейдемо до огляду наявних схожих систем, опираючись на функціонал, та алгоритми роботи яких можна використати в впровадженні власної системи під наявні задачі та проблеми:

1.3.1. Системи електронного документообігу.

Багато судових і медичних установ використовують спеціалізовані системи електронного документообігу (ЕДО) для зберігання та обміну медичними даними та звітами. Такі системи зазвичай вимагають інтеграції з веб-сайтом для забезпечення публічного доступу до певної інформації. Нижче представлений список прикладів систем електронного документообігу:

- DocuWare: DocuWare[5] - це популярна система ЕДО, яка дозволяє організаціям зберігати, керувати та обмінюватися документами в електронному вигляді. Вона має різні функції, включаючи роботу з електронними формами, потоками роботи та інтеграцію з іншими програмами.
- Microsoft SharePoint: SharePoint від Microsoft[6] - це платформа для спільної роботи та управління документами, яка включає в себе функції системи ЕДО. Вона дозволяє створювати, зберігати та обмінюватися документами, а також керувати потоками роботи.
- ELO Digital Office: ELO[7] - це система управління документами, яка включає в себе функції ЕДО. Вона надає можливість організаціям автоматизувати робочі процеси, ефективно виконувати збереження та пошук документів.

Ці системи допомагають підвищити ефективність роботи з документами, зменшити витрати на друкування та обробку паперових документів, а також полегшити спільну роботу над проектами та завданнями в організації.

1.3.2. Системи електронного управління документами.

Системи EDMS використовуються для зберігання, керування та обміну електронними документами. Вони можуть бути налаштовані для реалізації різноманітних функцій, включаючи обробку фінансової інформації та публікацію на веб-сайті. Ось декілька прикладів EDMS:

- OpenText Documentum: OpenText Documentum[8] - це потужна система управління документами та змістом, яка використовується для керування документами, електронною поштою та іншою інформацією у великих організаціях.
- IBM FileNet: IBM FileNet[9] - це рішення для управління документами, яке надає можливість організаціям зберігати та керувати різноманітною електронною інформацією, включаючи документи, зображення та відео.
- Hyland OnBase: Hyland OnBase[10] - це EDMS, яке дозволяє автоматизувати бізнес-процеси, створювати електронні форми та реєструвати документи.
- M-Files: M-Files[11] - це платформа для управління документами та інформацією, яка надає можливість зберігати та керувати документами на основі вмісту та метаданих.

Ці системи EDMS допомагають покращити ефективність роботи з документами, зменшити витрати на їх обробку та покращити доступність інформації для співробітників та клієнтів організації. Вони також забезпечують безпеку даних та можливість створювати робочі потоки для автоматизації бізнес-процесів.

1.3.3. Системи управління веб-сайтом.

Оскільки однією із проблем є відсутність веб-сайту, шляхом її вирішення є розробка веб-сайту. Відповідно аналізу проблеми, на веб-сайті має бути інформація, яку можна буде додати, редагувати та оновлювати. Для цих проблем ідеально підходить рішення застосувати систему управління веб-сайтом. Системи управління веб-сайтом (CMS) - це програмні рішення, які дозволяють легко створювати, редагувати та керувати веб-сайтами без необхідності глибокого знання програмування. Нижче наведено декілька прикладів популярних систем

управління веб-сайтом:

- WordPress: WordPress[12] є однією з найпопулярніших і безкоштовних CMS у світі. Вона легка у використанні, має велику спільноту користувачів і розширень, ідеально підходить для блогів та невеликих підприємств.
- Joomla: Joomla[13] - це потужна і розширювана CMS, яка підходить для створення складних веб-сайтів, включаючи інтернет-магазини та соціальні мережі.
- Drupal: Drupal[14] - це CMS, яка використовується для створення великих та високопродуктивних веб-сайтів, включаючи корпоративні ресурси та інтернет-портали.

Висновки до розділу

Отже, виходячи з проаналізованої інформації та наявних аналогів, можна зробити висновок, що ідеальним варіантом, щоб створити потрібну систему, яка буде виконувати всі поставлені задачі, буде захищеною, високопродуктивною та оптимізованою, буде варіант спроектувати одну систему скомбінувавши різні системи з документообігу, системи керування веб-сайтом, та внесення до них власних розробок під наявні проблеми. Система повинна складатись з веб-сайту, на якому буде публікуватись інформація про діяльність відділів та відділень, та фінансова звітність. Також потрібно розробити систему обміном документами (документообігом) між експертами, яка буде захищена, та буде мати можливість збереження цих документів.

2. ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ БЮРО СУДОВО-МЕДИЧНОЇ ЕКСПЕРТИЗИ

2.1. Функціональність веб-застосунку.

Система повинна мати визначений завданням функціонал, який зможе задовольнити всі вимоги установи, виконувати поставлені задачі, та автоматизовувати роботу, яка визначена завданням.

Система буде побудована на базі веб-сайту, так як саме веб-сайт може відображатись на майже всіх пристроях, що мають доступ до мережі Інтернет, та браузер. Умовно цей веб-сайт буде розділений на дві частини. Перша частина буде розроблена для адміністрації бюро судово-медичної експертизи та судово-медичних експертів, в ній буде відбуватись документообіг між експертами та між адміністрацією. Друга частина веб-сайту буде розроблена для громадськості. В ній буде публікуватись інформація про фінансову звітність організації, інформація для населення, така як: години роботи бюро, години прийому громадян, контактна інформація, інформація про напрямки роботи відділів та відділень, тощо.

На першій частині сайту потрібно запровадити такі функції:

- Роль адміністратора. Адміністратор матиме права додавати, редагувати, видаляти інформацію про діяльність відділу чи відділення, редагувати інформацію про керівника відділення, додавати новини до другої частини сайту, додавати інформацію про фінансову звітність на другу частину сайту.
- Роль начальника відділу/відділення. Начальник відділення матиме права надсилати направлення підлеглим, мати доступ до прогресу виконання експертизи, приймати та відправляти направлення іншим відділенням.
- Роль судово-медичного експерта. Судово-медичний експерт матиме право приймати направлення на експертизу, відправляти направлення, приймати результат направлення, отримувати результат направлення.

Ролі користувачів, та варіанти використання системи зображені на рисунку 2.1.

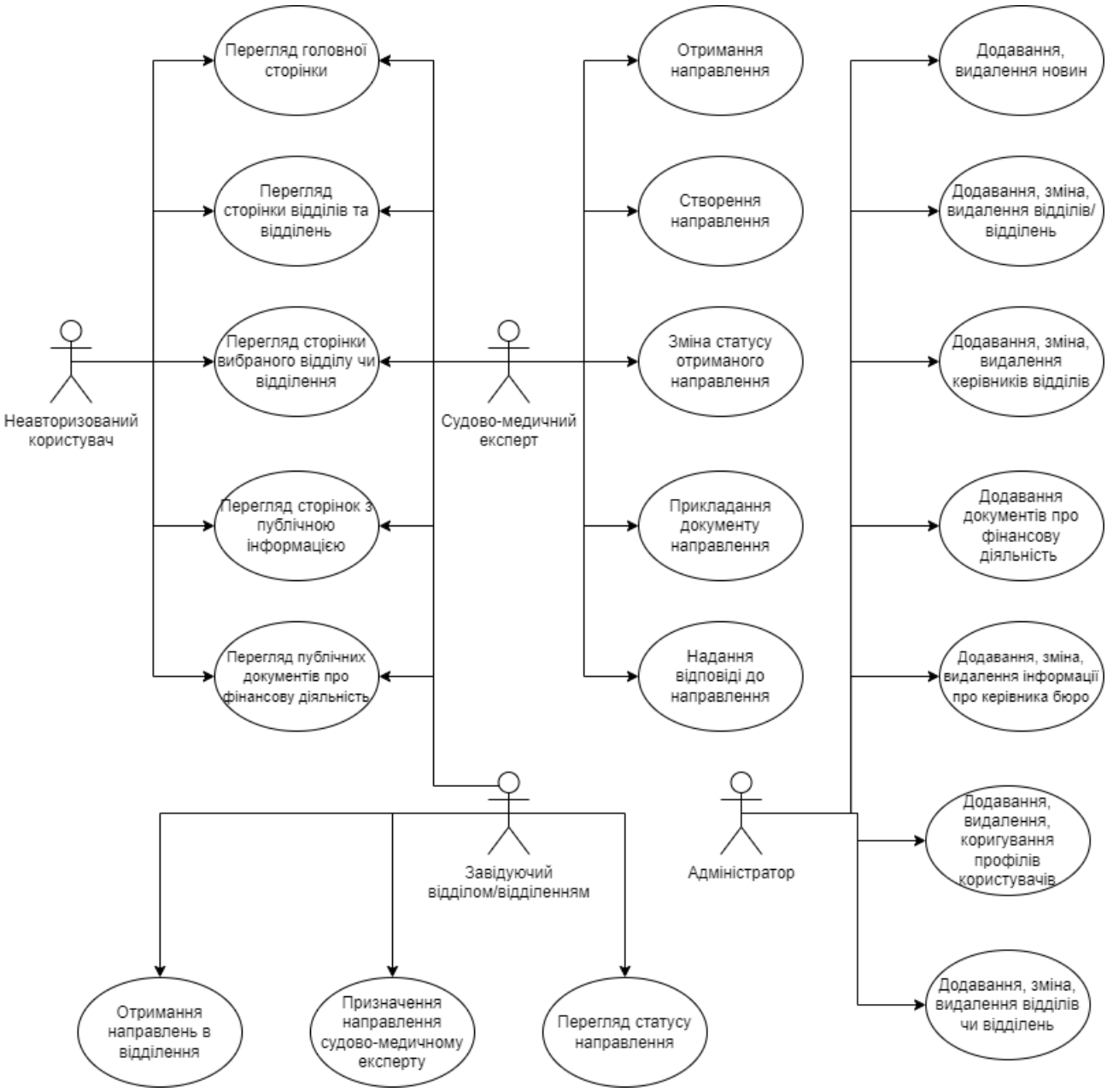


Рисунок 2.1 – UML-діаграма варіантів використання системи.

На другій частині сайту потрібно зробити таке наповнення:

- головну сторінку,
- сторінку для кожного відділу та відділення, на якій будуть описані напрямки роботи відділів чи відділень,
- інформація про завідувача відділом/відділенням,
- сторінка новин, кожна новина з якої буде відкриватись в окремій

сторінці,

- сторінку фотогалереї,
- сторінку з фінансовою інформацію, що буде складатись з посилань на сторінки, в яких буде відкрито PDF-документ,
- контакти – номери телефонів, адреси електронних пошт, години роботи, розташування,
- розділ кафедри судової медицини Вінницького національного медичного університету ім. М.І. Пирогова.

Виходячи з цих даних, можна зробити висновок, що потрібно розробити динамічний багатосторінковий веб-сайт. Розглянемо доступні методи вирішення проблеми. Потрібно розробити User Interface (надалі “UI”), використовуючи принципи та правила правильного User Experience (Надалі “UX”). Також потрібно виконати пошукову оптимізацію (Search Engine Optimization - SEO) щоб веб-сайт правильно та коректно індексувався пошуковими системами. Так як час від часу інформацію про діяльність відділів та відділень потрібно оновлювати, то можна використовувати кілька шляхів:

1. Кожного разу як потрібно змінити інформацію на сторінці, змінювати її в файлі сторінки (hard code), та завантажувати сторінку з оновленою інформацією на хостинг, замінюючи сторінку із застарілою інформацією. Так само і з додаванням нових pdf-документів – додавати їх до визначеної папки в хостингу через FTP-клієнт, потім додавати посилання на документ в сторінку фінансової інформації в список документів. Перевагами цього варіанту є: простота розробки, швидке отримання та завантаження даних. Недоліками варіанту є: застарілий метод розробки веб-сайтів, складність в оновленні інформації, складність в подальшій модернізації веб-сайту.
2. Застосувати розробку back-end частини, створити бази даних в яких буде зберігатись інформація, організувати передачу даних з клієнтської сторони на сервер і навпаки, щоб в front-end частині було мінімум статичної інформації (hard code), а більшість бралась з бази даних.Переваги: простота оновлення інформації на сайті, можливість створити будь-який необхідний

функціонал. Недоліки: тривалий час розробки.

3. Використати доступні системи управління вмістом (Content Manage System – CMS) з наявним прикладним програмним інтерфейсом (Application Program Interface – API), організувати взаємодію сторони клієнта та сторони сервера через цю систему управління вмістом.Переваги: простота в оновленні інформації на сайті, можливість використати потрібний функціонал, можливість встановлювати та використовувати доповнення та плагіни від спільноти, короткий час розробки. Недоліки: обмеження по функціоналу.

Оскільки розробка автоматизованої системи бюро судово-медичної експертизи включає в себе використання сучасного підходу, та використання новітніх технологій розробки, слід обрати варіант з використанням системи управління вмістом. Ще однією причиною для вибору саме системи контролю вмістом, в якості бекенд-частини застосунку є достатній рівень функціоналу, що пропонують такі системи.

2.2. Аналіз технологій для вирішення задачі.

2.2.1 Аналіз технологій для розробки користувацьких інтерфейсів.

Можна прописати всі умови відображення в браузері на «нативних» мовах – це HTML, CSS та JavaScript, але щоб досягти кращого результату за менший проміжок часу, потрібно обирати фреймворки та бібліотеки. Ось деякі з них:

1. React.

React — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Meta (раніше Facebook) і спільнотою індивідуальних розробників.[15, 16, 17]

React дозволяє розробникам створювати великі веб застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React

обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC[18], таких як AngularJS[19]. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб застосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, наприклад бібліотеками управління станом, такими як Redux, MobX, Zustand.

2. Vue.js

Vue.js — JavaScript-фреймворк, що використовує шаблон MVVM для створення інтерфейсів користувача на основі моделей даних[20], через реактивне зв'язування даних[21]. Vue використовує синтаксис шаблонів[22] на основі HTML, що дозволяє декларативно зв’язувати рендеринг DOM з основними екземплярами даних в Vue. Всі Vue шаблони валідні HTML, і можуть бути розпарсені браузерами та HTML парсерами. Всередині Vue компілює шаблони в рендерингові функції віртуального DOM. В поєднанні з реактивною системою, Vue здатний розумно обчислити кількість компонентів для ре-рендингу та застосувати мінімальну кількість маніпуляцій з DOM, коли стан застосунку зміниться.

В Vue ви можете використовувати синтаксис шаблонів або напряму писати рендерингові функції використовуючи JSX. Для того, щоб це зробити просто слід замінити шаблон на рендерингову функцію. Рендерингова функція відкриває можливості для потужних патернів базованих на компонентах — для прикладу, нова транзитна система тепер повністю базована на компонентах, що використовує рендерингові функції всередині[23].

Одна із найвиразніших особливостей Vue — це ненав'язлива реактивна система. Моделі це просто плоскі JavaScript об'єкти. Це робить керування станами дуже простим та інтуїтивним. Vue надає оптимізований ре-рендеринг з коробки без потреби робити що-небудь додатково. Кожен компонент слідкує за своїми реактивними залежностями під час рендерингу, тому система знає точно коли має

відбуватись ре-рендеринг і які компоненти потрібно ре-рендерити.

3. Next.js

Next.js — це веб фреймворк з відкритим вихідним кодом, створений приватною компанією Vercel, що дозволяє розробляти веб-додатки на основі React із рендерингом на стороні сервера та генерацією статичних веб-сайтів.

Документація React згадує Next.js серед «Рекомендованих наборів інструментів» і рекомендує його розробникам під час «створення серверного веб-сайту за допомогою Node.js»[24]. У той час як традиційні сайти React можуть відтворювати свій вміст лише в браузері на стороні клієнта, Next.js розширює цю функціональність, додаючи можливість генерації веб-сайту на стороні сервера.

Авторські права та торгові марки для Next.js належать компанії Vercel[25], яка підтримує та очолює розробку фреймворку як програмного забезпечення з відкритим вихідним кодом[26].

Next.js — це фреймворк React, який розширює його можливості додатковими функціями, включаючи рендеринг на стороні сервера та генерацію статичних веб-сайтів[27]. React — це бібліотека JavaScript, яка використовується для створення веб-додатків, які повністю працюють в браузері на стороні клієнта за допомогою JavaScript.

Однак розробники визнають проблеми такого підходу: неможливість роботи веб застосунку при вимкненому JavaScript в браузері користувача або його відсутності, потенційні проблеми з безпекою[28], значне збільшення часу початкового завантаження сторінки, оскільки необхідно відразу отримати від сервера повний код застосунку та шкода для пошукової оптимізації сайту. Такі фреймворки, як Next.js, розв'язують ці проблеми, дозволяючи окремим частинам або всьому веб-сайту генеруватися на стороні сервера перед відправленням клієнту[29]. Next.js є одним із найпопулярніших фреймворків для React.

2.2.2. Аналіз систем керування вмістом.

Між собою конкурує багато систем керування вмістом для веб сайтів, вони

відрізняються лиш сферами застосування, деяким функціоналом. Доцільно буде зібрати низку варіантів, що підходять саме під поставлену задачу, провести їх аналіз, зваживши всі переваги та недоліки, оцінивши наявність потрібного інструментарію потрібно обрати одну із представлених:

1) Netlify CMS

Netlify CMS є відкритою системою управління контентом[30]. Вона має інтуїтивно зрозумілі робочі процеси та користувацький інтерфейс, який легко використовувати. Netlify CMS працює з багатьма генераторами статичних сайтів, що надає вам перевагу у створенні швидких, гнучких та безпечних веб-проектів.

Основна перевага роботи з Netlify CMS полягає в тому, що весь контент та вихідний код зберігаються в Git-репозиторії замовника. Звісно, ви також можете організувати свій контент у підтеках. Додатково, ви можете використовувати додаткові моделі контенту для своєї зручності[31].

Вам просто не потрібен backend та бази даних. Однак це вплине на функціональність проекту. Ця система побудована на безсерверних, headless технологіях на основі клієнтського JavaScript, повторно використовуваних інтерфейсів програмування застосунків (API) та попередньо створеного розмітки. Ця структура робить її більш безпечною, ніж серверна CMS, наприклад WordPress.

Netlify CMS - це дуже корисний інструмент для швидкого управління та створення контенту з численними перевагами та спеціалізованими інструментами. З правильним відображенням контенту ви можете зробити досвід користувача легким та цікавим.

2) Contentful.

Contentful - це хмарна CMS, яка дозволяє легко створювати, оновлювати та публікувати контент через API. Вона має React SDK, який полегшує інтеграцію з React-додатками.

Основні риси та характеристики Contentful включають:

- Гнучкість та складність контенту[32]: Contentful дозволяє вам створювати і

керувати будь-яким типом контенту, включаючи текст, зображення, відео, аудіо, каталоги продукції, маркетинговий контент тощо. Ви можете визначати власні моделі даних та властивості контенту для відображення вашої специфічної структури.

- API-орієнтованість: Contentful базується на API, що дозволяє легко інтегрувати контент у ваші веб-сайти, мобільні додатки, IoT-пристрої та інші цифрові канали. Ви можете отримувати доступ до свого контенту через RESTful або GraphQL API.
- Множинні канали розповсюдження: Ви можете публікувати свій контент на різних каналах і мовах. Contentful надає можливість створювати варіанти контенту для різних цільових аудиторій та каналів розповсюдження.
- Колаборація і версіювання: Можливість спільно працювати над контентом, внесення змін і контроль версій допомагають командам ефективно співпрацювати над контентом.
- Шаблони та інтеграції: Contentful надає широкий вибір готових шаблонів для створення сторінок та додатків, а також інтеграції з популярними фронтенд-фреймворками, такими як React, Angular, Vue.js тощо.
- Автоматизація і розширення: Ви можете автоматизувати рутинні завдання і розширювати функціональність Contentful, використовуючи власні скрипти і плагіни.

Contentful є популярним вибором для веб-розробників і команд, які шукають потужну та гнучку систему управління контентом, яка дозволяє легко інтегрувати контент у всі свої цифрові проекти.

3) Strapi.

Strapi — це система керування вмістом (CMS) і веб-платформа, розроблена як програмне забезпечення з відкритим кодом. Він належить до групи headless (безголових) CMS і тому не надає власного інтерфейсу, а лише інтерфейс програмування (API), через який дані доставляються іншим програмам. Цей API автоматично генерується з моделі даних, звідки також походить назва CMS (

Bootstrap your API)[33]. Програмне забезпечення розробляється однойменним французьким виробником, який пропонує власні додаткові функції, а також підтримку CMS. Система була написана на JavaScript для середовища виконання Node.js. Система контролю вмістом має такі особливості :

З одного боку, Strapi є CMS, оскільки він пропонує функції, які зазвичай є в CMS, наприклад текстовий редактор, керування користувачами та медіа-бібліотеку. З іншого боку, це веб-фреймворк для веб-сервісів, оскільки він дає змогу розробляти власні функції та кінцеві точки API[34].

Зберігання даних:

Strapi може використовувати різні бази даних для зберігання даних, включаючи реляційні бази даних SQLite , MySQL і MariaDB[35] . Моделі даних створюються за допомогою конфігураційних файлів, записаних у форматі даних JSON . Однак також можна використовувати візуальний редактор, який потім генерує файли конфігурації[36]. Потім ці моделі даних створюються як таблиці бази даних у відповідній базі даних.

API:

На основі налаштованих моделей даних автоматично створюється API, який надає кінцеві точки для читання, створення, редагування та видалення записів для кожної моделі даних[37]. Одночасно створюється доступна документація, яка описує ці кінцеві точки[38]. API надає функції для фільтрації та розбиття даних на сторінки, а також доступ для запису. За допомогою плагіна GraphQL також можна ефективно запитувати зв’язки складних даних. Однак API не пропонує повнотекстовий пошук , керування користувачами та доступ у реальному часі в моделі push[39].

Програмування:

API можна програмувати, змінювати або розширювати в JavaScript через проміжне програмне забезпечення. Це дозволяє, наприклад, інтегрувати ваші власні кінцеві точки, бізнес-логіку та перевірку даних . Strapi базується на веб-фреймворку Koa.js, наступнику Express.js . Це означає, що розробники також можуть використовувати існуюче проміжне програмне забезпечення, сумісне з

Koa.js[40].

Інтерфейс користувача:

Як CMS Strapi також пропонує інтерфейс користувача для адміністраторів, авторів і користувачів без попередніх технічних знань через веб-додаток . Створення записів і написання текстів можливі через веб-додаток. Поля введення та функції програми адаптуються до відповідних моделей даних. Наприклад, якщо поле дати зберігається в типі даних, інтерфейс користувача відображає календар як поле введення. Для створення текстів доступний редактор WYSIWYG . Також можливе керування користувачами та правами доступу. Ви можете створювати власні ролі[41] та призначати їх користувачам. Веб-програма може бути адаптована розробниками до потреб власних користувачів. Існуючі компоненти CMS також можна розширити або замінити. Бібліотека React JavaScript використовується для програмування веб-додатку .

2.2.3. Вибір технологій для розробки системи.

Найкращим варіантом технології для розробки інтерфейсів користувача, серед наведених, є використання фреймворку Next.js, оскільки цей фреймворк надає змогу користуватися різними методами відображення розмітки для браузера, включаючи методи відображення бібліотеки React. Найбільшим плюсом фреймворку є те, що сторінки можна оптимізувати за всіма правилами SEO[42] (Search Engine Optimization), які є дуже важливими для індексації веб-сайту різними пошуковими системами. В цьому плані використання бібліотеки React мало б великий мінус, оскільки компоненти React, це функції, що повертають JSX-розмітку, яка потім з допомогою транспілятора babel.js[43], перетворює цю JSX-розмітку на HTML-розмітку з стилями CSS та Javascript-код, котрий відображається в браузері клієнта. Проблема полягає в тому, що перетворення JSX на зрозумілий для браузера код відбувається на стороні клієнта, а не сервера, тому пошукові системи не можуть просканувати інформацію сайту та додати його до індексації[48]. З використанням серверних компонент Next.js ця проблема нівелюється, оскільки на запит сторінки повертається відразу код, зрозумілий для

браузера (HTML, CSS та Javascript), а транспіляція з JSX відбувається на сервері. Тому для розробки системи варто обрати фреймворк Next.js, як інструмент для розробки інтерфейсу користувача.

Як систему керування вмістом варто обрати CMS Strapi, оскільки ця система контролю вмістом гарно зарекомендувала себе в різних проектах, має широку спільноту розробників, корисну та зрозумілу документацію, великий набір плагінів. Також це headless (безголова) CMS[45], це означає, що система сфокусована на бекенд частині, та не пропонує фронтенд частини, де б відображались дані з неї. Це дає змогу подальшого масштабування автоматизованої системи бюро судово-медичної експертизи під відображення не тільки в вигляді сайту, а наприклад, мобільного додатку, чи прикладної програми. Також обрана система керування вмісту має зручний інструментарій для створення та заповнення таблиць в базі даних, влаштування зв’язків[46] між ними, та налаштування доступу до цих таблиць.

3.РОЗРОБКА ТА ТЕСТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ

3.1. Побудова логіки серверної частини системи.

Розробку слід почати саме з серверної частини додатку, оскільки робота фронтенд частини напряму залежить від тих даних, які прийдуть з серверу. Для системи автоматизації бюро судово-медичної експертизи, в якості серверної частини обрано Content Managing System “Strapi”.

3.1.1. Ініціалізація проекту Strapi.

Для початку, цю систему потрібно ініціалізувати та встановити. На офіційному сайті CMS “Strapi” є інструкція по встановленню. Першим кроком буде створення директорії проекту, оскільки веб-застосунок ділиться на 2 основні частини, а саме фронтенд та система контролю вмістом, в директорії проекту в процесі ініціалізації частин проекту буде створено папки “frontend” та “cms”. В першій надалі буде ініціалізовуватись та розроблятись Next.js проект (фронтенд частина веб застосунку). Папку «cms» вручну створювати не потрібно, вона буде створена в процесі ініціалізації проекту Strapi. Для ініціалізації проекту Strapi потрібно відкрити термінал в директорії проекту та використати команду:

```
npx create-strapi-app@latest cms –quickstart
```

, [47] де «cms» — назва проекту, яка також буде використана для назви папки проекту. Після успішного виконання команди, можна запускати сервер з системою контролем вмістом. Для запуску цього серверу, потрібно відкрити папку cms в терміналі та ввести команду “npm run develop”, після чого починається процес будування програмних контекстів, створення адмін-панелі, завантаження користувацьких та вбудованих плагінів, зчитування змінних розробки, генерація типів, та інших процесів. Коли всі потрібні дані для роботи системи контролю вмісту будуть завантажені та налаштовані, в командному рядку буде інформація про адресу та порт, на якій розгорнуто цей сервер.

Далі потрібно створити акаунт до системи Strapi. Це потрібно для авторизації

в ній. Потрібно ввести електронну адресу, та придумати складний пароль. Після того, як створено акаунт, потрібно пройти авторизацію. Після авторизації відкриваються всі можливості системи контролю вмістом Strapi, знімок екрану вигляду панелі адміністратора системи контролю вмістом Strapi продемонстрована на рисунку 3.1.

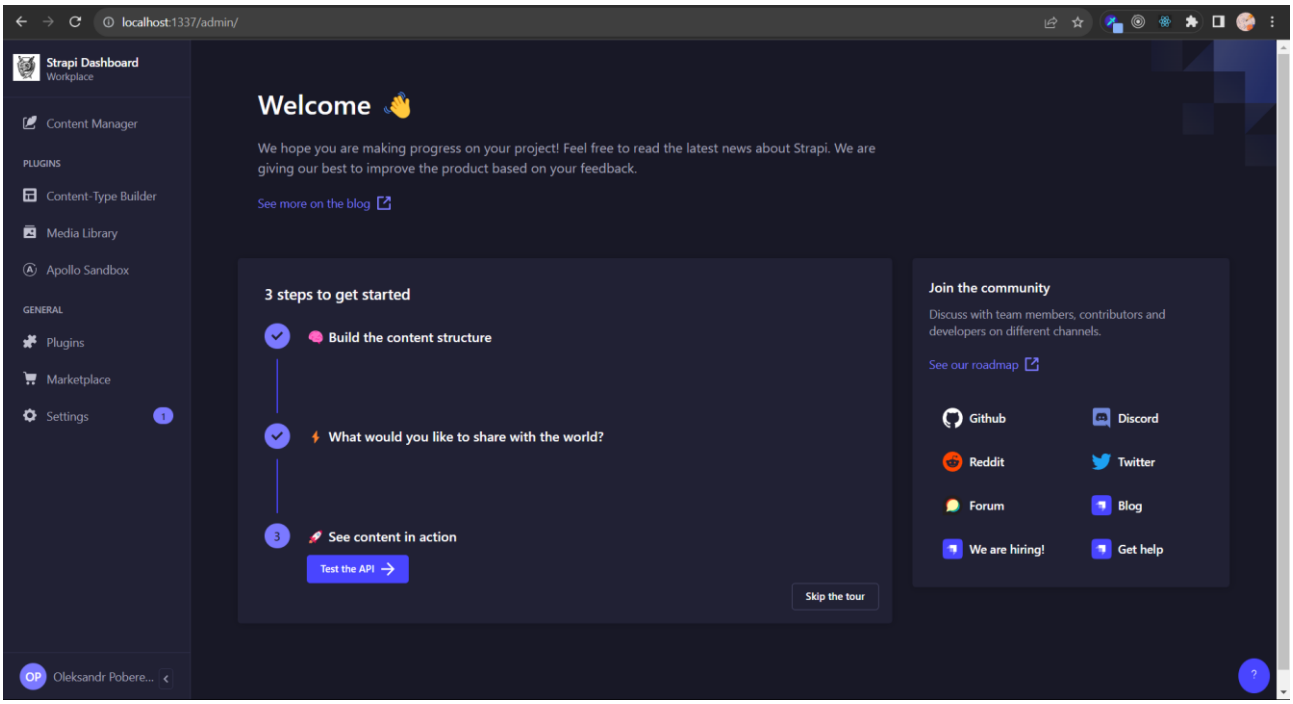


Рисунок 3.1 – Скриншот панелі адміністратора системи керування вмістом Strapi.

Наведу перелік основних вкладок в панелі адміністратора, з якими найчастіше буде вестись робота, та опишу їх функціонал.

- Content Manager (Менеджер вмісту) – елемент системи керування вмістом, в якому здійснюється створення, редагування та видалення контенту (вмісту) [34];
- Content-type Builder (Конструктор типів вмісту) – основний плагін системи, робота без якого неможлива. Надає змогу створювати типи вмісту, редагувати чи видаляти їх. З Content-type builder адміністратор може створювати типи колекцій та окремі типи[49];
- Media Library (Бібліотека медіафайлів) – основний плагін системи, який включає в систему можливість роботи з медіа. Це можуть бути

зображення форматів JPEG, PNG, GIF, SVG, TIFF, ICO, DVU, відео форматів MP4, MPEG, Quicktime, WMV, AVI, FLV, аудіофайли форматів MP3, WAV, OGG чи інші файли форматів CSV, ZIP, PDF, Excel, JSON і т.д. Плагін потрібен в системі контролю вмістом, оскільки автоматизована система бюро-судово медичної експертизи повинна працювати з документами, десь зберігати їх, та керувати ними[50];

- Settings (Налаштування) – меню налаштувань Strapi. Буде використовуватись в роботі з плагіном User-Permissions для надання прав користувачам.

3.1.2. Побудова моделей для неавторизованих користувачів.

Щоб зберігати дані в Strapi, потрібно створити моделі, типи контенту. Модель має вигляд таблиці з полями, та може мати зв’язки з іншими моделями. Для створення моделі потрібно перейти в «Content-Type Builder», і в колонці з типами колекцій створити новий тип колекції. Для частини додатку, котра призначена для користувачів, що не є співробітниками бюро судово-медичної експертизи потрібно створити такі типи колекцій:

- department. Буде зберігати інформацію про відділи та відділення установи. Тип колекції буде мати поля: title типу text – назва відділу чи відділенмня, slug типу текст – транслітерована коротка назва відділу/відділення для пошуку та навігації, поле description з типом даних Rich Text (для можливості введення інформації використовуючи мову розмітки markdown) ,встановити зв’язок з типом колекції head-of-department з відношенням один до одного (Одне відділення може мати та відноситись до одного завідуючого відділенням);
- head-of-department. Буде зберігати інформацію про завідуючого відділом чи відділенням. Тип колекції повинен мати поля: name з типом даних text – для прізвища, імені та по-батькові завідуючого, поле Photo типу Single Media, з дозволеним типом медіаданих «Image», поле biography з типом даних Rich Text, поле contacts з типом даних JSON, що буде містити об’єкт

з масивом tel – який буде зберігати публічні номери телефонів завідуючого відділом/відділенням, та полем email в який буде вноситись адреса електронної пошти. Також потрібно встановити зв’язок з типом колекції department, один до одного (один завідуючий має та відноситься до одного відділення);

- news. Буде зберігати новини, які будуть додаватись на сайт. Тип колекції повинен мати поля: title, типу Text, в яке буде вноситись заголовок новини, поле slug типу Text – коротке посилання на новину, previewImage з типом даних Single Media «Image», та поле description з типом даних Rich Text в яке буде вноситись тіло новини з можливістю форматування мовою markdown;
- ldepartment. Матиме такі ж поля як і department, зв`язок з head-of-department один до одного (ldepartment має і відноситься до одного head-of-department). Але служитиме для зберігання інформації про районні відділи та відділення бюро судово-медичної експертизи.
- fininfo. Тип колекції служитиме для зберігання публічної фінансової інформації. Матиме такі поля: title типу text, що буде заголовком сторінки інформації, slug типу text – коротке посилання на запис, document типу Single File Media, що буде зберігати документ з фінансовою інформацією установи.

Так як система контролю вмісту, це набір пакетів та модулів, котрий зібраний в одну програму – повноцінний веб-застосунок з інтерфейсом користувача, то можна відкрити файли коду цієї системи, та переглянути зміни в коді при застосуванні змін через інтерфейс користувача. Для перегляду можна відкрити папку «cms» в будь-якому інтегрованому середовищі розробки (в моєму випадку Visual Studio Code). Файли схем моделей (типів колекцій) знаходяться за шляхом:

папка_проекту/src/api/назва_типу_колекції

в цій директорії потрібно відкрити папку «content-types», де буде ще одна папка з назвою типу колекції, а в ній файл «schema.json»[51]. Файл «schema.json» містить JSON-об’єкт, що являється інструкцією для побудування моделі бази даних.

В коді схема моделі матиме вигляд, зображений на рисунку 3.2.

```
{
  "kind": "collectionType",
  "collectionName": "departments",
  "info": {
    "singularName": "department",
    "pluralName": "departments",
    "displayName": "department",
    "description": ""
  },
  "options": {
    "draftAndPublish": true
  },
  "pluginOptions": {},
  "attributes": {
    "title": {
      "type": "string",
      "required": true
    },
    "slug": {
      "type": "string",
      "required": true,
      "regex": "^[a-z0-9]+(?:-[a-z0-9]+)*$",
      "unique": true
    },
    "head_of_department": {
      "type": "relation",
      "relation": "oneToOne",
      "target": "api::head-of-department.head-of-department",
      "mappedBy": "department"
    },
    "description": {
      "type": "richtext",
      "required": true
    }
  }
}
```

Рисунок 3.2 – Скриншот файлу schema.json для типу колекції department.

Маючи дані про поля та зв'язки таблиць бази даних (типів колекцій), в аналітичних цілях можна побудувати діаграму бази даних[52] з таблицями, визначених для доступу користувача, котрий не повинен бути авторизований в системи бюро судово-медичної експертизи. На рисунку 3.3 наведена діаграма щойно створених таблиць бази даних.

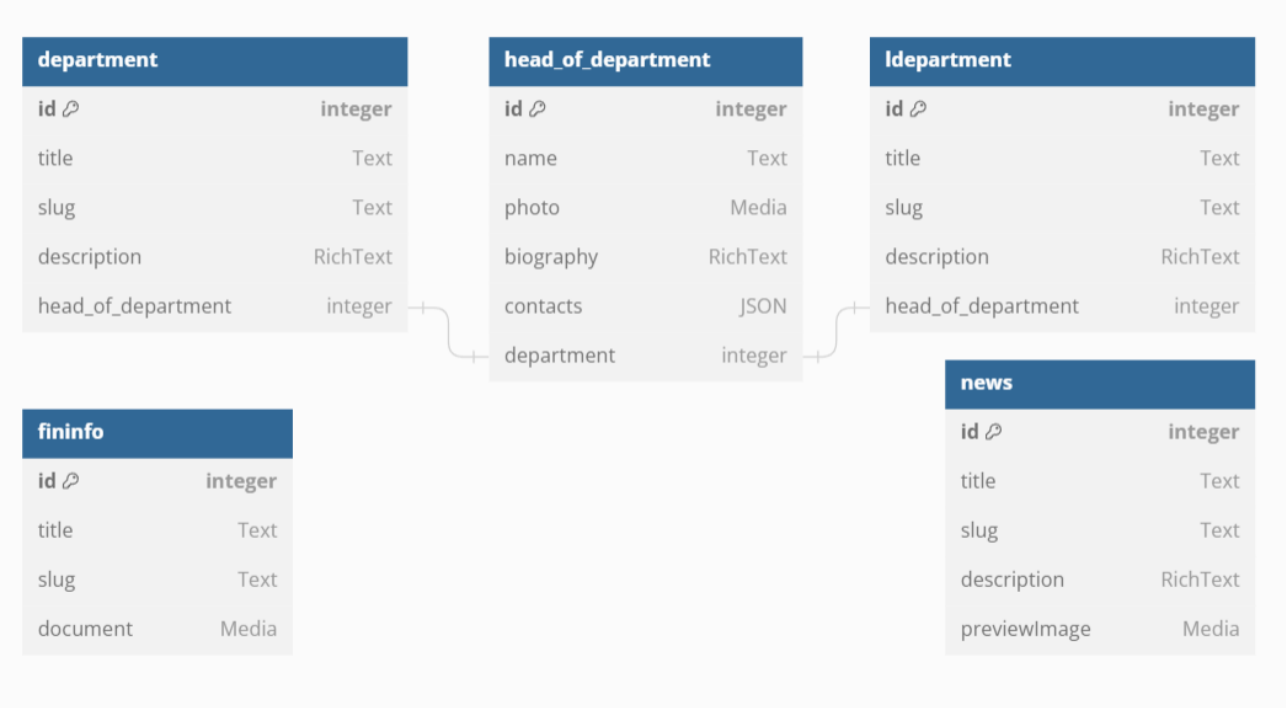


Рисунок 3.3 – Діаграма бази даних з таблицями для неавторизованого користувача.

Наступним кроком буде налаштування доступу до таблиць із вмістом. В Strapi для цього існують налаштування доступу по Roles тих, хто запитує інформацію. За замовчуванням в Strapi є дві ролі для користувача:

1) Public. Роль дається для всіх запитів із відсутніми данними авторизації,
2) Authenticated. Роль дається для запитів з данними авторизації.

Оскільки звичайний відвідувач сайту не повинен виконувати ніяких авторизаційних дій для перегляду публічної інформації, тоді потрібно надати права доступу «find» та «findOne» до типів колекцій визначених вище, в розділі плагіну Users & Permissions Plugin в налаштуваннях системи контролю вмістом.

3.1.3. Побудова моделей для авторизованих користувачів.

Побудуємо моделі для частини системи, де буде відбуватись обмін направленнями та відповідями на ці направлення між судово-медичними експертами. Процес відправки та отримання направлень має виглядати таким чином: при виконанні судово-медичної експертизи, судово-медичному експерту

танатологу потрібно створити направлення на дослідження вилучених, потрібних для експертизи об’єктів дослідження, в відділення токсикології. Для цього судово-медичний експерт бере об’єкт для дослідження, до прикладу, на аналіз в відділення токсикології буде надсилатись кров, заповнює об’єктом пробірку, закорковує, та опломбовує її, наносячи на пломбу номер пломби та особистий підпис. В застосунку ці дані будуть вказуватись в документі з направленням. Оскільки об’єкти дослідження потрібно отримувати разом із направленням, а в випадку з відправкою направлення через систему, що розробляється направлення може бути доставлене раніше ніж об’єкти дослідження, тому слід створити поле status з типом даних Enumeration, в якому потрібно обрати один варіант з цього списку. В варіанти до списку status потрібно додати такі: «надіслане», «отримане», «в обробці» та «з відповіддю». При створенні направлення, та заповнення всіх необхідних полів до його існування та відправці, направленню, в фронтенд частині додатку повинний даватись статус «надіслане». Статус направлення «Отримане» повинен застосовуватись коли експерт-отримувач отримає також і об’єкти для досліджень. Статус «В обробці» повинен ставитись коли експерт-отримувач почав роботу над направленим йому об’єктом дослідження. Статус «З відповіддю» має встановлюватись коли експерт-отримувач надішле експерту-надсилачу документ-відповідь на направлення.Також потрібно створити поле responseDocument з типом даних Single Media File[53], яке повинно заповнюватись на момент відповіді експертом-отримувачем направлення.

Значить, для впровадження обміну направленнями потрібні такі типи колекцій:

- referral. Направлення;
- cabinetDepartament. Окрема від демонстративної для користувачів без авторизації колекція відділень та відділів;
- cabinetExpert. Колекція судово-медичних експертів;
- cabinetHeadOfDepartment. Колекція завідуючих відділами чи відділення.

Для повного функціонування направлення потрібні такі поля для типу колекції referral:

- referralTitle. Заголовок та номер направлення. Тип даних Short Text;

- referalDocument. Файл направлення. Тип даних Single Media File;
- status. Статус відправлення. Тип даних Enumeration List з значеннями «надіслане», «отримане», «в обробці» та «з відповіддю»;
- responseDocument. Файл відповіді експерта-отримувача. Тип даних Single Media File.

Тип колекції referral повинен мати зв'язки з такими типами колекцій:

- Зв'язок sender_department. Зв'язок з типом колекції cabinetDepartment, відповідає відділенню/відділу з якого надсилається направлення. Тип зв'язку – багато до одного (відділення має багато направлених направлень);
- Зв'язок recipient_department. Зв'язок з типом колекції cabinetDepartment, відповідає відділенню/відділу до якого надсилається направлення. Тип зв'язку – багато до одного (відділення має багато отриманих направлень);
- Зв'язок recipient. Зв'язок з типом колекції cabinetExpert, відповідає судово-медичному експерту, який отримує направлення. Тип зв'язку – багато до одного (судово-медичний експерт має багато отриманих направлень);
- Зв'язок sender. Зв'язок з типом колекції cabinetExpert, відповідає судово-медичному експерту, який відправляє направлення. Тип зв'язку – багато до одного (судово-медичний експерт має багато відправлених направлень).

Тип колекції cabinetDepartment повинен мати такі поля та зв'язки:

- Поле title. Тип даних Text. Повинне містити інформацію з назвою відділення;
- Зв'язок head_of_department. Зв'язок з типом колекції cabinetHeadOfDepartment, відповідає завідуючому відділом/відділенням. Тип зв'язку – один до одного (Відділення має та належить до одного Завідувача відділенням);
- Зв'язок experts. Зв'язок з типом колекції CabinetExpert, відповідає експертам, що працюють в відділенні. Тип зв'язку – один до багатьох (Відділення має багато Судово-медичних експертів);

- Зв’язок sent_referrals. Зв’язок з типом колекції referral, відповідає направленням, які надіслані з відділення. Тип зв’язку – один до багатьох (Відділення має багато Відправлених направлень);
- Зв’язок received_referrals. Зв’язок з типом колекції referral, відповідає направленням які надійшли до відділення. Тип зв’язку – один до багатьох (Відділення має багато Отриманих відправлень).

Тип колекції cabinetExpert повинен мати такі поля та зв’язки:

- Поле name. Тип даних Text. Повинне містити інформацію про ім’я, прізвище, та по-батькові судово-медичного експерта;
- Зв’язок department. Зв’язок з типом колекції cabinetDepartment, відповідає відділенню в якому працює судмедексперт. Тип зв’язку – багато до одного (Відділення має багато Судмедекспертів);
- Зв’язок received_refferals. Зв’язок з типом колекції referrals, відповідає за отримані судово-медичним експертом направлення. Тип зв’язку – один до багатьох (Судмедексперт має багато Отриманих направлень);
- Зв’язок sent_refferals. Зв’язок з типом колекції referrals, відповідає за відправлені судмедекспертом направлення. Тип зв’язку – один до багатьох (Судмедексперт має багато Відправлених направлень).

Тип колекції cabinetHeadOfDepartment повинен містити такі поля та зв’язки:

- Поле name. Тип даних Text. Повинне містити інформацію про ім’я, прізвище та по-батькові завідуючого відділенням;
- Зв’язок department. Зв’язок з типом колекції cabinetDepartment, відповідає за відділ/відділення, який очолює завідувач відділенням/відділом. Тип зв’язку – один до одного (Завідувач відділенням має та належить до одного Відділення).

Наступним кроком буде налаштування плагіну «Users & Permissions Plugin» для новостворених моделей. Потрібно створити дві нових ролі, для цього потрібно перейти до налаштувань Strapi, обрати секцію «Users & Permissions Plugin» та перейти на вкладку «Roles». Далі потрібно створити дві нових ролі, а саме: судово-медичний експерт та завідуючий відділом. Оскільки в Strapi вже з моменту

ініціалізації цього плагіну існує дві ролі: Public та Authenticated, слід перейменувати роль Authenticated на «Expert», щоб не створювати зайвих ролей. Залишилось створити роль завідуючого відділом, для цього слід натиснути кнопку «+ Add new role», та вказати в полі назви нової ролі значення «HeadOfDepartment».

Наступним кроком буде надання новоствореним ролям прав для типів колекцій. Потрібно надати доступ до find та findOne усіх існуючих типів колекцій. Для типу колекції referrals також потрібно додати доступ до create, update та delete. На цьому етапі налаштування CMS Strapi можна вважати завершеним. На рисунку 3.4 наведена діаграма таблиць, створених для авторизованих користувачів.

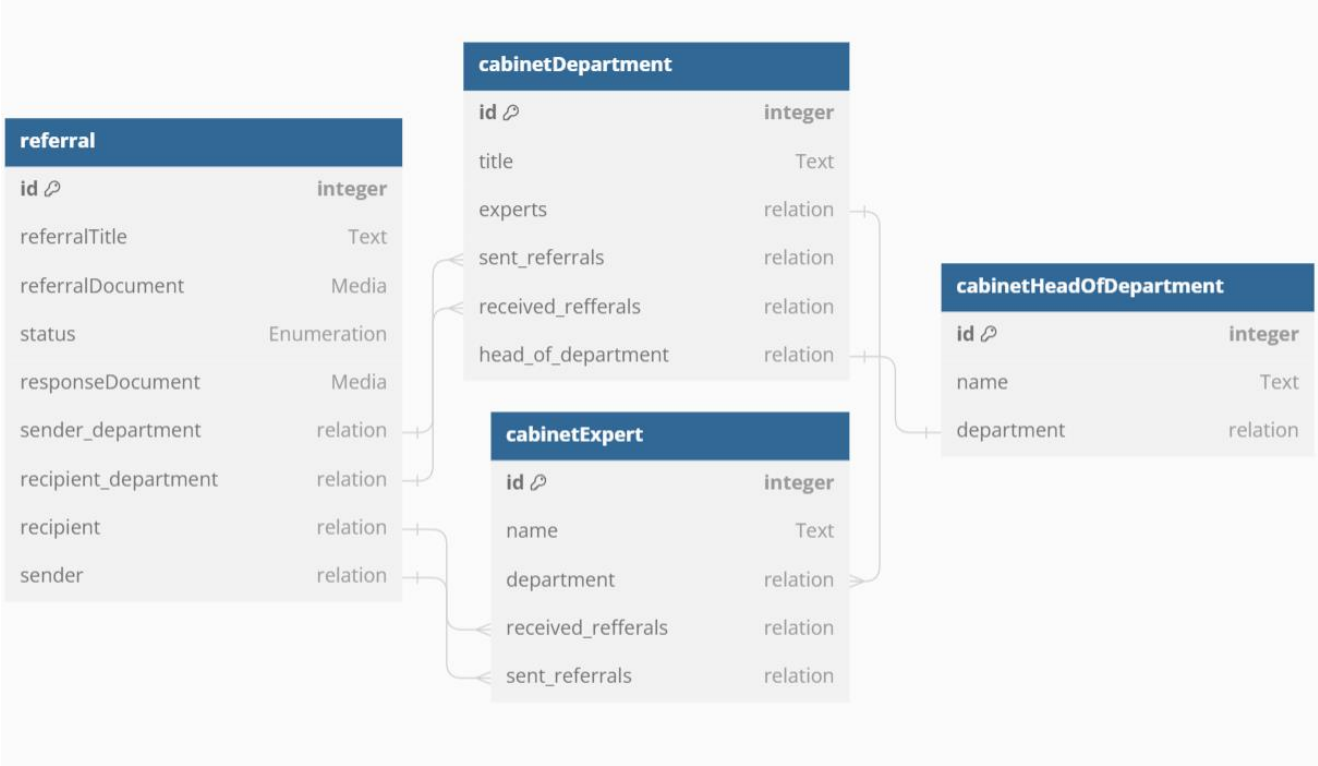


Рисунок 3.4 – Діаграма бази даних, таблиць, котрі доступні тільки авторизованим користувачам

3.2. Розробка Front-end частини системи.

3.2.1. Ініціалізація проекту Next.js.

Першим кроком розробки фронтенд частини застосунку буде ініціалізація проекту Next.js[54]. Для цього перейдемо в папку проекту, потрібно відкрити цю

папку в терміналі, та застосувати команду:

npx create-next-app@latest

Після чого відбувається перехід в меню ініціалізації проекту, що зображене на рисунку 3.5.

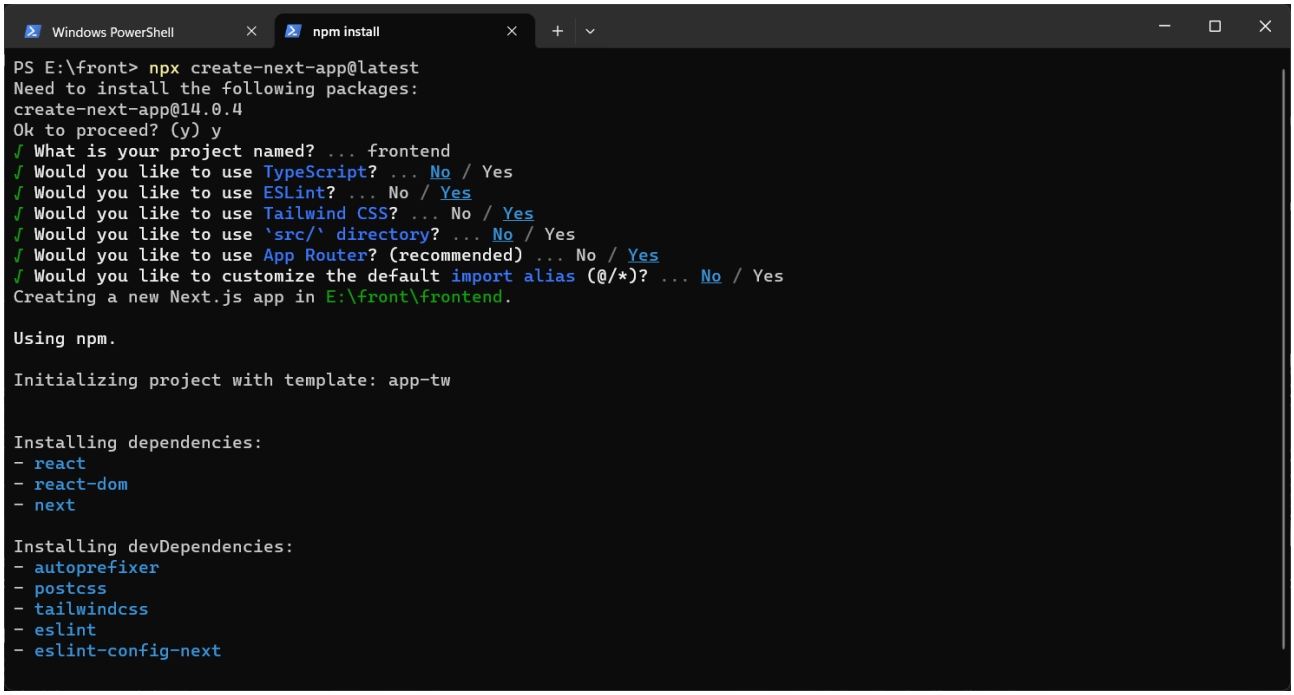


Рисунок 3.5 – Скриншот командного рядка при виконанні ініціалізації Next.js проекту.

Потрібно виконати ряд умов, перш ніж починається встановлення всіх необхідних пакетів та модулів. Першою умовою є встановлення імені проекту, раніше було встановлено, що проект умовно поділений на дві частини, це фронтенд частина та частина системи контролю вмістом, тому даємо Next.js проекту ім’я «frontend». Після чого йде вибір технологій та бібліотек, які потрібно включити до проекту. Система буде писатись на нативному JavaScript, тому TypeScript не потрібен, обираємо відмову. Далі йде питання чи потрібен в проекті інструмент ESLint[55]. ESLint — це інструмент статичного аналізу коду для виявлення проблемних шаблонів, знайдених у коді JavaScript. Правила в ESLint можна налаштувати, а також можна визначити та завантажити налаштовані правила. ESLint охоплює питання якості коду та стилю кодування. Оскільки бібліотека

принесе лише користь, оберемо згоду на її використання. Чи бажаєте використовувати Tailwind CSS[56]? В розділі аналізу та вибору технологій було проаналізовано та обгрунтовано вибір інструментів та бібліотек для стилізації фронтенд частини веб-застосунку, тому обираємо згоду. Опускаємо створення папки src/ оскільки в фронтенд частині додатку не буде зберігатись ніяких медіафайлів, окрім логотипу установи, натомість вони будуть запитуватись в системи контролю вмістом. Передостаннім пунктом ініціалізації проекту Next.js є вибір роутера (системи маршрутизації та навігації по веб-застосунку). Загалом, в Next.js існує дві системи маршрутизації між сторінками, це: Pages Router та App Router. Нижче наведено порівняльну таблицю цих систем маршрутизації.

Таблиця 3.1 – Порівняльна таблиця App та Pages роутерів

Pages Router	App Router
Простий та легкий в використанні	Більш гнучкий
Гарне рішення для невеликих додатків	Гарне рішення для великих додатків
Краще підходить для статичних сторінок	Краще підходить для динамічних сторінок

Аналізуючи дані наведені в Таблиці 3.1, можна зробити висновок, що використання App Router для проекту буде доцільнішим за Pages Router, тому погоджуємося на його використання при ініціалізації проекту Next.js. Останнім пунктом перед початком завантаження всіх потрібних модулів та бібліотек є вибір чи використати заміну імпортування модулів за замовчуванням, обираємо відмову, оскільки позитивний вибір не потягне за собою суттєвих позитивних змін для розробки проекту.

Після вибору всіх параметрів та першого налаштування проекту Next.js, завантажиться файл package.json та package-lock.json, які містять інструкції для проекту, а саме: ім’я проекту, версію проекту, залежності проекту, залежності проекту під час розробки (dev dependencies). Вигляд файлу package.json продемонтрований на рисунку 3.6. Після завантаження файлів package.json та

package-lock.json починається завантаження всіх необхідних бібліотек та модулів, що прописані в залежностях в папку node_modules/ проекту[57], створиться файл конфігурації Next.js проекту з ім’ям next.config.js, робоча папка app, в якій мають розміщуватись всі сторінки проекту, папка public, в якій слід розміщувати іконки, зображення, шрифти та інші медіафайли.

Рисунок 3.6 – Вміст файлу package.json.

Тепер слід приступити до розробки, для початку потрібно зібрати версію проекту для розробки, та запустити локальний сервер, щоб не робити це вручну, в Next.js існують node packages manager scripts (скрипти менеджеру пакетів node.js), для розробки потрібно використовувати скрипт “dev”, для того щоб його запустити, потрібно відкрити папку з проектом Next.js в терміналі, та прописати команду npm

run dev, починається процес збору розробницької версії проекту та розгортання її на локальному сервері з IP-адресою 127.0.0.1 (localhost – власна локальна адреса комп'ютера), на порту 3000. Далі можна відкрити цю адресу в браузері:

http://localhost:3000/

Оскільки Next.js це фреймворк для розробки веб-застосунків та сайтів, в ньому існує стартовий шаблон проекту, що при першому запуску розробник міг бачити структуру сторінок (зображена на рисунку 3.7.) та маршрутизацію по ним. Стартова сторінка проекту знаходиться за адресою http://localhost:3000/, файл з кодом до якої розміщений в папці app проекту, та має назву layout.js.

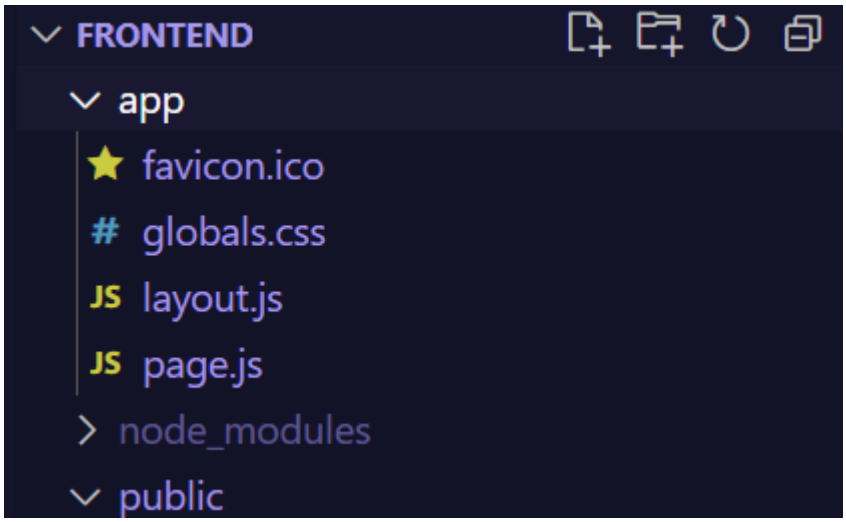


Рисунок 3.7 – Стартова структура сторінок в директорії app.

Файл layout.js не являється сторінкою, але використовується для обгортки сторінок, що містяться в директорії разом із ним, та для дочірніх директорій[58]. Такий підхід потрібен для того, щоб не повторювати код, він називається DRY – Don't Repeat Yourself[59] (Не повторюй себе) – це принцип розробки, який широко використовується в усіх областях програмування, оскільки дублювання коду це марна трата часу та ресурсу, яка зробить логіку проекту ненадійною, прикладом для пояснення послужить зміна старого коду та додавання нового функціоналу. Наприклад, на сайті є частина Header – верхня частина сайту, на якій часто розміщується логотип, і до прикладу навігаційна панель сайту. Змоделюємо ситуацію що в проекті створено вже більш ніж 20 сторінок, на кожній з яких

прописаний Header, і раптом, розробник розуміє, що йому потрібно змінити посилання в навігаційній панелі веб-застосунку на інше, а щоб це зробити, йому потрібно змінити посилання в навігаційній панелі в Header на кожній з 20 сторінок сайту. Такий підхід є недоцільним, тому що є висока ймовірність допущення помилки в одній з 20 цих сторінок, також буде витрачено більше часу розробки, тому слід використовувати файл layout.js, який буде обгорткою для кожної сторінки веб-сайту. В файлі layout.js слід включити всі елементи, які будуть повторюватись на кожній сторінці, і в випадку коли слід щось змінити для всіх сторінок, не змінювати це на кожній сторінці проекту, а змінити все в одному місці проекту.

В цьому випадку для всіх сторінок буде застосовано Header (верхня частина сайту з навігацією по ньому та логотип) та Footer (нижня частина сайту, на якій розміщується додаткова інформація, та корисні посилання, які розробник не хоче розміщувати в Header). Тому слід додати їх в layout.js, а між ними додати ключове слово {children} на місці якого будуть розміщені сторінки проекту. В програмному коді сторінка layout.js продемонстрована на рисунку 3.8.

```
13
14  export default function RootLayout({ children }) {
15    return (
16      <html lang="en">
17        <body className={inter.className + " " + "min-h-screen flex flex-col bg-gradient-to-t
18          <Header />
19          {children}
20          <Footer />
21        </body>
22      </html>
23    )
24  }
25
```

Рисунок 3.8 – Програмний код файлу layout.js.

Сторінка яка розміщується всередині layout.js повинна мати назву page.js або page.jsx. На одному шляху повинен бути лише один файл page.jsx, маршрут[60] будується від структури директорії app/, що зображено на рисунку 3.9. Тобто, щоб створити стартову сторінку, яка буде за адресою localhost:3000, потрібно створити файл page.js прямо в директорії app/, але щоб створити сторінку яка буде за адресою

localhost:3000/departments, не можна створювати файл departments.js в директорії app/, таким чином сторінка не буде індексуватись сайтом. Натомість потрібно створити директорію departments/ всередині app/, і вже в директорії departments/ створювати файл page.js. Page.js відрисовується відразу після layout.js.

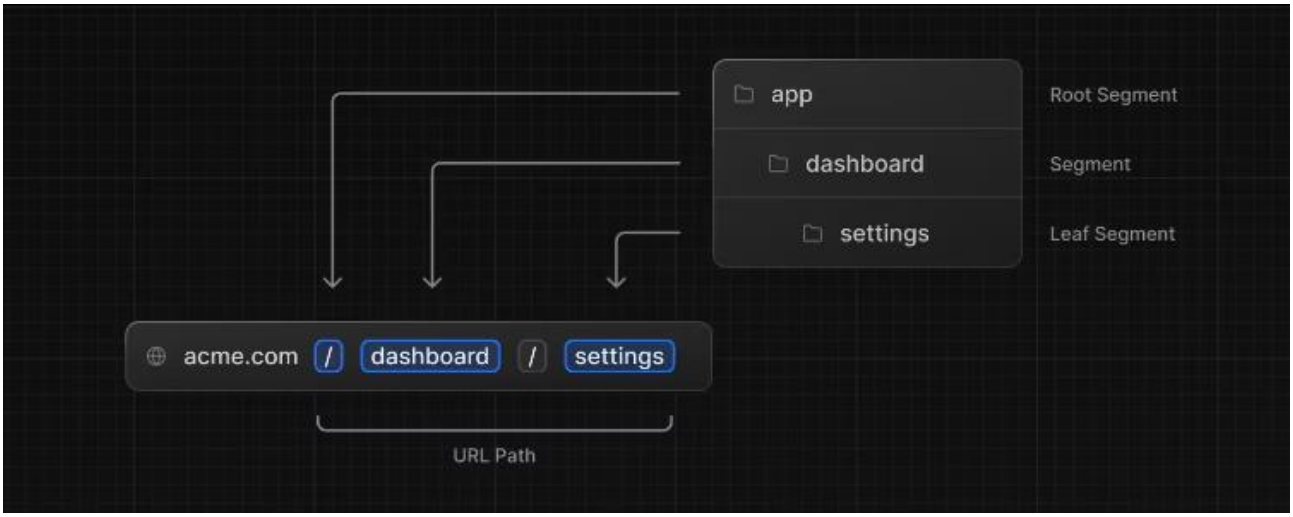


Рисунок 3.9 – Побудова маршруту сайту від директорії app.

3.2.2. Розробка панелі навігації та компоненту Header.

Наступним кроком потрібно обрати створення компонентів Header та Footer, що будуть спільними для всіх сторінок веб-застосунку, тому будуть розміщені в файлі layout.js. Компоненти – частини додатку, що можуть повторюватись, компоненти створюються за принципом чистих функцій. Чиста функція – це функція, яка при однакових вхідних значеннях має виводити однакові вихідні значення. В випадку компонент, функція повинна отримувати якісь props, та виводити JSX-розмітку. Приступимо до написання компоненту Header. В кореневій директорії проекту Next.js створимо папку components, в якій створимо папку Header, а в ній файл Header.jsx. Створення папки components потрібно для того, щоб унеможливити відкривання користувачем компоненти замість сторінки. До прикладу, якщо створити компоненту Header в директорії app/Header/Header.jsx, то користувач, ввівши в адресній строці браузера localhost:3000/Header/Header.jsx відкриє компоненту Header, що може призвести до небажаних наслідків чи помилок.

В файлі Header.jsx створимо стрілкову функцію Header, та експортуємо її за замовчуванням, щоб React розумів, що це саме компонента, вивід якої (return) потрібно перетворити в HTML розмітку, та відрисувати на сторінці. По завданню, в верхній частині сайту, повинен міститись логотип, та меню навігації по сайту з такими пунктами:

- Головна. Посилання на головну сторінку сайту;
- ВОБ СМЕ. Меню з розділами:
  - Про нас. В цьому підрозділі повинні бути посилання на сторінки історії установи, фотогалереї, адміністрації бюро та начальника бюро,
  - Структура бюро. Тут повинні бути посилання на сторінки відділів та відділень організації.
- Інформація. Інформація про діяльність закладу. Розділ повинен містити такі підрозділи:
  - Інформація. Підрозділ повинен містити посилання на загальні положення, нормативну базу, публічну інформацію, інформацію для слідчих;
  - Фінансова діяльність. Підрозділ повинен містити посилання на сторінки з публічною інформацією про фінансову діяльність установи останніх років, та посилання на сторінку кошторису;
  - Мапа сайту. Підрозділ має містити посилання на сторінку мапи сайту.
- Контакти. Контактна інформація та розташування закладу;
- Кафедра. Посилання на сайт кафедри судово-медичної експертизи Вінницького Національного Медичного Університету ім. М.І. Пирогова.

Виходячи з цієї інформації, слід зазначити, що кожна вкладка навігаційного меню повинна мати посилання, текстову назву для відображення в меню, категорію та підкатегорію. Тому слід написати масив об’єктів headerNavStaticState, який буде містити в собі об’єкти-вкладки навігаційного меню код яких зображено на рисунку 3.10.

```
{
    link: '/departments/viddilennia-cherhovykh-sm-ekspertiv',
    text: 'Відділення чергових СМ експертів',
    subcategory: 'Структура бюро',
    category: 'ВОБ СМЕ'
},
{
    link: '16',
    text: 'Загальні положення',
    subcategory: 'Інформація',
    category: 'Інформація'
},
```

Рисунок 3.10 – Програмний вигляд об’єкту вкладки навігаційного меню.

Масив об’єктів з інформацією про пункти навігаційного меню створюється з ціллю відображення кожного об’єкту цього масиву однією функцією-методом масиву, яка має назву map. Метод масиву map перебирає кожен елемент масиву, що дає змогу відфільтрувати масив чи змінити його, та записує результати в новий масив. В випадку створення навігаційного меню метод map потрібен для того, щоб перебрати всі елементи масиву, взяти інформацію кожного елементу, та сформувати на даних цієї інформації навігаційну панель веб-застосунку.

Першим кроком створення навігаційного меню, є створення ненумерованого списку з пунктами навігації найвищого рівня всередині тегу <nav></nav>, а саме, створення списку з записами: «Головна», «ВОБ СМЕ», «Інформація», «Контакти» та «Кафедра» код цього ненумерованого код найвищого рівня навігаційної панелі зображений на рисунку 3.11. До кожного елементу ненумерованого списку слід додати обробник подій onClick, який буде обробляти подію одноразового кліку на елемент списку. Так як перехід за посиланням повинен відбуватись тільки при виборі елементу меню «Головна», то на цей елемент списку, в обробник подій onClick буде додана функція navigate(“/”) – яка виконує перенаправлення користувача за адресою localhost:3000/ , тобто, на головну сторінку веб-застосунку, також разом із нею виконується функція setOpenedTab(undefined). На решту елементів списку, в обробник подій onClick передається функція

changeOpenedTab(), яка приймає в параметри елемент DOM-дерева, з якого надалі візьметься назва вкладки, на яку потрібно змінити відкритий пункт меню.

```
213     <nav className='w-full flex flex-row justify-center items-center relative px-2'>
214         <ul className="h-16 flex flex-row justify-between items-center text-white w-full font-probapro text-lg lea
215             <li
216                 onClick={(e) => { navigate('/'); setOpenedTab(undefined) }}
217                 className={openedTab === "Головна" ? `${'nav-li-style'} ${'nav-li-style-active'}` : `${'nav-li-sty
218                 Головна
219             </li>
220             <li
221                 onClick={(e) => changeOpenedTab(e)}
222                 className={openedTab === "ВОБ СМЕ" ? `${'nav-li-style'} ${'nav-li-style-active'}` : `${'nav-li-sty
223                 ВОБ СМЕ
224             </li>
225             <li
226                 onClick={(e) => changeOpenedTab(e)}
227                 className={openedTab === "Інформація" ? `${'nav-li-style'} ${'nav-li-style-active'}` : `${'nav-li-
228                 Інформація
229             </li>
230             <li
231                 onClick={(e) => changeOpenedTab(e)}
232                 className={openedTab === "Контакти" ? `${'nav-li-style'} ${'nav-li-style-active'}` : `${'nav-li-st
233                 Контакти
234             </li>
235             <li
236                 onClick={(e) => changeOpenedTab(e)}
237                 className={openedTab === "Кафедра" ? `${'nav-li-style'} ${'nav-li-style-active'}` : `${'nav-li-sty
238                 Кафедра
239             </li>
240         </ul>
241     </nav>
```

Рисунок 3.11 – Програмний код першого рівня навігаційної панелі веб-застосунку.

Також для кожного елементу списку пунктів меню першого рівня дається клас, згідно цих класів застосовуються Tailwind CSS стилі до пункту меню. При надані класу, впроваджена перевірка, чи саме цей пункт меню являється відкритим в навігаційній панелі додатку, якщо так, йому надаються особливі стилі, для індикації того, що користувач знаходиться саме в цьому пункті меню.

Щоб реалізувати логіку відкриття вкладок другого рівня (підкатегорій пунку меню з посиланнями чи інформацією) використаний хук useState:

const [openedTab, setOpenedTab] = useState(undefined) ,

з бібліотеки React. useState(undefined) повертає масив з двох елементів, першим елементом буде переданий в useState() параметр, в даному випадку (undefined - невизначений), оскільки на початку пункт меню користувачем є не обраний, та не потрібно показувати другий рівень навігаційної панелі, другим елементом масиву

є функція, яка буде перевизначати значення першого елементу масиву. Суть використання useState() полягає в тому, що в цьому масиві буде зберігатись інформація локального стану, кожна зміна якого приведе до перерисовки компоненти, в якій використовується цей локальний стан, без перезавантаження сторінки, так в проекті отримується динамічність на стороні клієнта. Логіка навігаційної панелі така, що коли користувач обирає пункт першого рівня меню, йому відображається другий рівень меню, який відповідає пункту першого рівня, а якщо користувач натисне другий раз по обраному пункту меню першого рівня, другий рівень меню потрібно приховати, також коли користувач обере інший пункт першого рівня меню, то потрібно відрисувати другий рівень меню, який відповідає першому, у випадку коли користувач обирає пункт «Головна» першого рівня меню, потрібно встановити значення undefined для openedTab, щоб другий рівень навігаційного меню закрився, саме для цього, до пункту меню «Головна» в обробник подій onClick також передана функція setOpenedTab(undefined).

Наступним кроком буде написання функції changeOpenedTab(), задачею якої буде зміна другого рівня навігаційного меню, в залежності від першого. Для того щоб визначити, до якого пункту меню першого рівня буде застосована функція, в параметр функції потрібно передати елемент, до якого спрацює обробник події onClick, після того як елемент був переданий до функції, потрібно взяти його значення, зрівняти з попереднім (значенням попередньої відкритої вкладки першого рівня меню), що міститься в змінній prevTab першого елементу масиву повернутого з useState:

```
const [prevTab, setPrevTab] = useState(undefined) ,
```

якщо значення обраного елементу списку співпадає з значенням попереднього відкритого пункту навігаційного меню першого рівня, другий рівень навігаційного меню потрібно приховати, встановивши для openedTab значення undefined, та перезаписати в змінну prevTab значення теперішньої обраної вкладки меню. Якщо значення обраного елементу не співпадає з значенням попереднього відкритого пункту меню, то потрібно встановити змінній openedTab значення елементу переданого в функцію. Також всю логіку функції потрібно огорнути перевіркою чи

переданий елемент до функції, якщо елемент списку першого рівня не передається, то значення змінним prevTab та openedTab потрібно встановити undefined.

Програмний код функції changeOpenedTab наведений на рисунку 3.12.

```
const changeOpenedTab = (e) => {
    if (!e) {
        setOpenedTab(undefined)
        setPrevTab(undefined)
    } else {

        if (prevTab === e.target.childNodes[0].data) {
            setPrevTab(undefined)
            setOpenedTab(undefined)
        } else {
            setOpenedTab(e.target.childNodes[0].data)
            setPrevTab(e.target.childNodes[0].data)
        }

    }
}
```

Рисунок 3.12 – Програмний код функції changeOpenedTab().

Тепер в компоненті Header наявна логіка зміни другого рівня меню для навігаційної панелі, але її відображення не застосовано. Потрібно описати логіку відображення в return ( ) цього компонента. Потрібно запровадити перевірку, другий рівень меню слід відрисовувати тільки в випадку, коли значення змінної openedTab не дорівнює undefined, тому створюємо блок, до його стилів даємо умову, якщо openedTab !== undefined, то надаємо стилі, які дозволять відображення цього блоку, інакше, додати клас стилю hidden, який служить для того, щоб приховати блок з DOM-дерева сторінки. Після того як умова видимості блоку меню з елементами другого рівня буде виконана, потрібно додати перевірку того, чи не відкрита вкладка «Контакти», оскільки на цій вкладці не міститься інформації, що буде братись з масиву об’єктів headerNavStaticState, якщо openedTab === “Контакти”, то відобразити наповнення вкладки «Контакти» заданих як HTML, якщо ні, то взяти масив subcategories, який містить в собі назви всіх підкатегорій, які існують в масиві headerNavStaticState, застосувати до нього метод map(), щоб

пройтись по кожному елементу масиву, та вивести назву підкатегорії, під назвою підкатегорії слід вивести text тих об’єктів, в яких subcategories відповідає назві підкатегорії, для цього потрібно взяти масив filteredObjects, який складається з об’єктів масиву headerNavStaticState, в яких category дорівнює openedTab, застосувати до нього метод filter(), де потрібно обрати тільки ті елементи масиву, в яких subcategory дорівнює обраній підкатегорії, filter() поверне новий масив, до якого потрібно застосувати метод map() для того, щоб вивести в підкатегорію, ті елементи, які їй відповідають, та зробити їх посиланнями на сторінки з адресою, вказаною для них в масиві об’єктів headerNavStaticState. Програмний код зображений на рисунку 3.13.

```
243     <div className={openedTab !== undefined ? `${'hidden absolute'} ${'header__bottom__details__active'}` : `${'hidder
244         <div className="container mx-auto px-4">
245             <div className='flex flex-row gap-10'>
246                 {openedTab !== 'Контакти' ? subcategories.map((subcategory) => (
247                     <div key={subcategory}>
248                         <h3 className="text-xl">{subcategory}</h3>
249                         <ul className="mt-5 ml-5 flex flex-col gap-3 max-h-[200px] flex-wrap">
250                             {filteredObjects
251                                 .filter((item) => item.subcategory === subcategory)
252                                 .map((item) => (
253                                     <li className='topic' key={item.link} onClick={() => { setOpenedTab(undefined); se
254                                 ))}
255                         </ul>
256                     </div>
257                 )) :
258                     <div className='flex flex-row justify-evenly w-full'>
259                         <div>
260                             <iframe
261                                 src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d2605.5129444067893!2d28.476
262                                 width="400"
263                                 height="300"
264                                 style={{ border: 0 }}
265                                 allowFullScreen=""
266                                 referrerPolicy="no-referrer-when-downgrade"
267                             >
268                                 <img src={'/logoWhiteTest.png'} alt="" />
269                             </iframe>
```

Рисунок 3.13 – Програмний код фільтрації та виведення пунктів списку меню по підкатегоріям.

Завершальним етапом розробки компонента Header буде додання логотипу та стилізація елементів компонента з використанням готових класів бібліотеки Tailwind CSS. Для цього до кожного елемента, який потрібно стилізувати, слід додати атрибут className та передати імена класів, стилі яких потрібно

застосувати до елементу. Результат розробки компоненти Header, що доданий в файл layout.js наведений на рисунку 3.14.

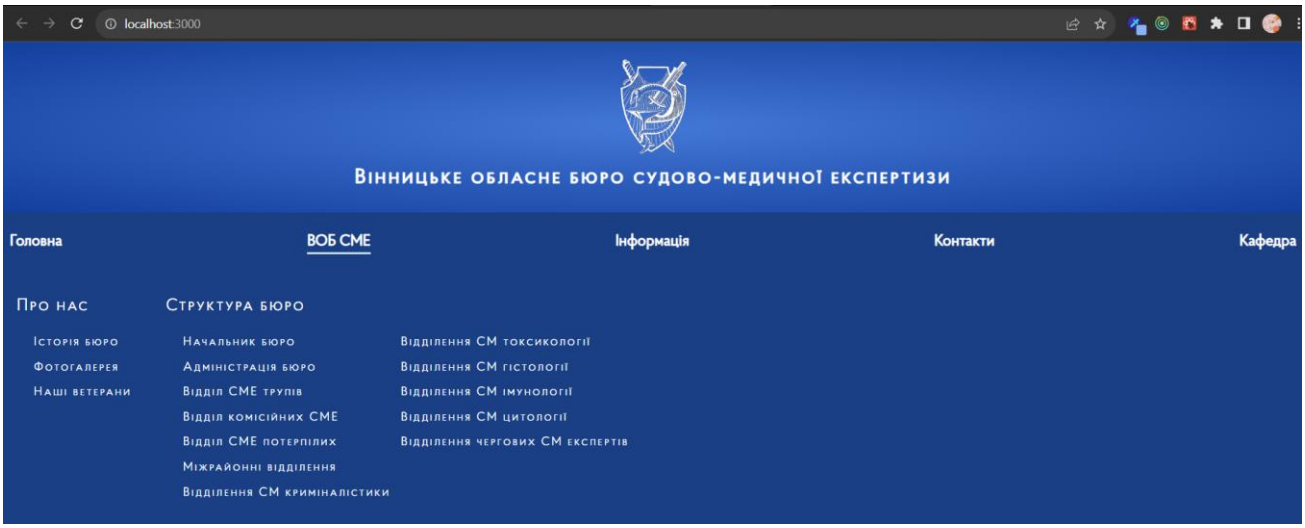


Рисунок 3.14 – Відрисований на обгортці сторінки компонент Header, з відкритим навігаційним меню на вкладці «ВОБ СМЕ».

3.2.3. Розробка головної сторінки додатку.

Згідно плану розробки фронтенд частини, на головній сторінці додатку повинен бути розміщений блок з новинами.

Так як головна сторінка повинна знаходитись за адресою localhost:3000/ , то в каталозі проекту, файл page.jsx, повинен бути створений прямо в папці app/, біля файлу layout.js. Додамо цей файл, створимо в ньому функцію Home(), та експортуємо її за замовчуванням, щоб в Next.js був доступ до її перегляду та можливість включення сторінки до загальної маршрутизації по сторінкам.

Створимо блок, в якому створимо тег <h1> з вмістом «Новини», далі перейдемо до розробки компоненту <AllNews />, який буде отримувати новини з системи контролю вмістом Strapi, та відображати їх на головній сторінці веб-застосунку.

В папці components, що знаходиться в корені каталогу проекту, створимо папку AllNews, а в ній файл AllNews.jsx. В файлі AllNews.jsx створимо функцію AllNews та експортуємо її за замовчуванням. Перейдемо до файлу page.jsx, імпортуємо компонент <AllNews /> та додамо її на сторінку, щоб компонент

відображався в застосунку, код головної сторінки зображений на рисунку 3.15.

```
import AllNews from "@/components/News/News";

export default function Home() {
  return (
    <main className="text-[black] font-probapro text-xl py-4">
      <div className="container mx-auto px-5">
        <h1>Новини</h1>
        <AllNews />
      </div>
    </main>
  )
}
```

Рисунок 3.15 – Програмний код головної сторінки веб-застосунку.

Перейдемо до написання функціоналу компонента AllNews. Отже, компонент має звертатись до системи контролю вмістом Strapi із запитом на отримання новин, цей вміст з серверної частини додатку повертається в вигляді JSON, тому потрібно по отриманню вмісту, перетворити його в вигляд HTML.

Першим кроком в написанні компоненту AllNews буде формування запиту на отримання вмісту. Для цього потрібно відправити запит на сервер з CMS. Потрібно переконатись що сервер Strapi запущений та працює, а потім взяти посилання на нього і порт, на якому він розміщений, так як частина CMS також знаходиться в розробці, та запущена в режимі розробника, то вона буде знаходитись на локальній адресі комп’ютера, це адреса 127.0.0.1 або localhost, та порт 1337. В цілях безпеки, адресу серверу CMS, на яку буде робитись запит, потрібно приховати, щоб її не було видно в файлах коду. Найкращим варіантом це зробити, буде використання змінних середовища (Process Environment Variables). Щоб ввести в проект ці змінні середовища, потрібно створити в корені каталогу файл без назви, з розширенням .env. Додати в нього ці змінні в форматі:

НАЗВА_ЗМІННОЇ=ЗНАЧЕННЯ .

Зберегти цей файл, та додати змінні середовища до файлу конфігурацій next.config.js. Створимо дві змінні, перша – посилання на адресу системи контролю

вмістом Strapi, друга – посилання на API цієї системи контролю вмістом. Назвемо їх STRAPI_URL та STRAPI_API_URL, дамо їм рядкові значення, для першої – “http://localhost:1337” для другої – “http://localhost:1337/api”. Додамо змінні середовища до конфігурації next.config.js. Потрібно до об’єкту nextConfig додати об’єкт з назвою env, та передати змінні середовища в вигляді ключ:значення , де ключ – одноіменна з змінною середовища назва, а значення – посилання на цю змінну через process.env. Скриншот зміненого файлу конфігурацій next.config.js продемонстрований на рисунку 3.16.

Рисунок 3.16 – Програмний код файлу конфігурацій next.config.js з вказаними в ньому змінними середовища .env, та налаштуваннями доступу до зображень з системи контролю вмістом Strapi.

Запити на сервер в Next.js робляться з допомогою вбудованої в фреймворк функції fetch(), але для зручної та комфортної обробки запитів та їх результатів,

функцію fetch() можна модифікувати. Оскільки в подальшому потрібно буде відправляти не тільки GET-запити, а й PUT, POST та DELETE, додавати заголовки, та тіла для запитів, а отримувати не Promise, а JSON, то потрібно в корені каталогу створити директорію lib, в якій будуть зібрані додаткові файли з функціями для проекту, створити файл api.js, і в ньому створити асинхронну функцію fetcher() котра в параметри буде брати посилання для запиту, та об’єкт з налашутваннями запиту, функція повинна перевіряти чи передали в неї конфігурацію для запиту, якщо так, то додавати ці конфігурації в запит, якщо ні, то робити GET-запит на вказану в першому параметрі адресу, та повертати JSON, який прийде з серверу в відповідь на цей запит. Програмний код цієї функції наведений на рисунку 3.17.

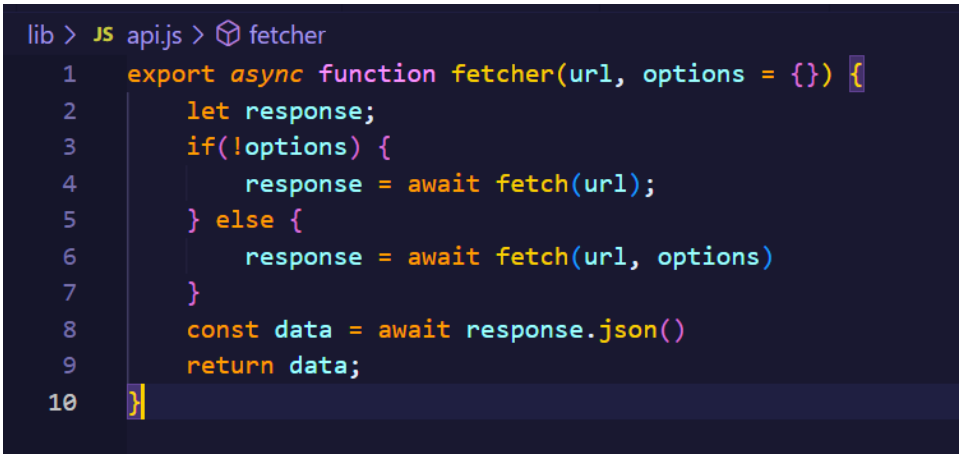


Рисунок 3.17 – Програмний код функції fetcher().

Оскільки кістяк безпечного надсилання запитів та зручного отримання відповідей побудовано, настав час використати цей функціонал в розробці компоненту AllNews. Першим кроком імпортуємо щойнонаписану функцію fetcher до компоненту, далі створимо константу newsData, якій присвоїмо результат виконання асинхронної функції fetcher(), так як функція fetcher асинхронна, перед її викликом потрібно додати ключове слово “await”. Першим параметром, переданим для функції fetcher() має бути адреса, за якою потрібно зробити GET-запит на новини. Тому для першого параметру потрібно відкрити форматований рядок, передати змінну STRAPI_API_URL, оскільки модель новин можна отримати з API Strapi, та звернутись за ендпоінтом newss, також потрібно отримати пов'язані

з моделлю зображення (якщо вони є), передати параметри pagination, для того, щоб не навантажувати сервер великим обсягом відповіді, а отримати новини посторінково, також передати параметри сортування. Наступним кроком буде відображення отриманих новин в вигляді HTML на сторінці. Так як з серверу приходить об'єкт JSON, дані в ньому містяться в вкладеному об'єкті data. Data, у відповіді серверу – це масив об'єктів, де кожен об'єкт це елемент з моделі новин, тобто одна новина. Щоб розмістити їх на веб-сайті, слід застосувати до цього масиву метод map(), та передати в JSX id новини та її attributes. Оскільки кожна новина є шаблонною, потрібно створити для неї окремий компонент, під назвою NewsCard. Компонент NewsCard буде приймати в props: id новини, та її attributes, attributes містять всю інформацію про новину, таку як назва, опис, зображення, дата створення та дата публікації, транслітерована назва новини для подальшої маршрутизації, має назву slug.

Розробку компонента NewsCard слід почати з написання функції конвертування формату дати публікації, оскільки формат дати, що повертається з системи контролю вмістом є незручним для читання та сприймання користувачем. Для цього напишемо функцію convertDate, що буде приймати на вхід рядок дати, та буде повертати рядок дати формату «день місяць рік». Програмний код функції convertDate зазначений на рисунку 3.18.

```
const convertDate = (dateString) => {
    const months = [
      'січня',
      'лютого',
      'березня',
      'квітня',
      'травня',
      'червня',
      'липня',
      'серпня',
      'вересня',
      'жовтня',
      'листопада',
      'грудня'
    ];
    const date = new Date(dateString);
    const day = date.getUTCDate();
    const month = months[date.getUTCMonth()];
    const year = date.getUTCFullYear();

    return `${day} ${month} ${year}`;
  }
```

Рисунок 3.18 – Програмний код функції convertDate().

Задачею компоненту NewsCard буде конвертування вхідних даних в JSX розмітку. Оскільки компонент викликається при переборі масиву об’єктів з новинами, то можна сказати точно, що без вхідних даних компонент викликаний не буде, тоді потрібно брати дані з attributes, та відображати їх в HTML-тегах. Картка новини повинна мати наступний вигляд: блок, що містить дату публікації новини, зображення новини, якщо воно є в об’єкті відповіді серверу, якщо його немає, вставити шаблонне абстрактне зображення паперової газети, під зображенням новини повинен міститись заголовок новини що міститься в attributes.title, далі повинно бути розміщено перших 4 стрічки опису новини з attributes.description, які перериваються трьома крапками, в самому низу компонента має бути кнопка «Перейти до новини», яка буде вести користувача за адресою “localhost:3000/news/${attributes.slug}”, де ${attributes.slug} заміниться значенням поля slug новини, яку користувач відкриє. Код компоненту зображений на рисунку 3.19.

```
export default function NewsCard({ id, item }) {
    const convertDate = (dateString) => {···
      }
    return (
        <div className="relative w-[30%] max-md:w-[45%] max-[492px]:w-[100%] flex flex-col items-start justify-
            <div className="absolute top-1.5 left-1 flex flex-row gap-x-2 items-center">
                <Image src={'/icn-time.svg'} width={20} height={20} alt="time"/>
                <p className="text-base text-[gray]">{convertDate(item.publishedAt)}</p>
            </div>
            {item.previewImage.data ?
                <Image
                    src={`${process.env.STRAPI_URL}${item.previewImage.data.attributes.url}`}
                    width={300}
                    height={item.previewImage.data.attributes.height}
                    alt={item.previewImage.data.attributes.alternativeText ? item.previewImage.data.attributes.
                    style={{ objectFit: "cover", backgroundColor: 'rgba(12,12,12,0.3)', height: '200px' }}
                    className="mx-auto"
                /> :
                <div className="h-[200px] w-[100%] flex justify-center items-center">
                    <FontAwesomeNewspaper color={"gray"} size={'2xl'} />
                </div>
            }
            <h2 className="font-semibold">{item.title}</h2>
            <p className="line-clamp-4 w-[100%]">{item.description}</p>
            <Link href={`news/${item.slug}`} className="absolute bottom-2 right-5 hover:text-headerFirst te
        </div>
    )
}
```

Рисунок 3.19 – Програмний код компонента NewsCard.

На цьому етапі розробка компоненту новини NewsCard закінчена, компонент потрібно імпортувати в компоненті AllNews, та вставити в метод перебору масиву новин з відповіді серверу newsData. Програмний код компонента AllNews наведений на рисунку 3.20.

components > News > News.jsx > AllNews

```
1  import { fetcher } from "@/lib/api"
2  import NewsCard from "./NewsCard";
3
4  export default async function AllNews () {
5      const newsData = await fetcher(`${process.env.STRAPI_API_URL}/newss?populate=previewImage
6                          &pagination[start]=0&pagination[limit]=3&sort[0]=id:desc&sort[1]=id`)
7      return (
8          <div className="container mx-auto px-5 text-[black] flex flex-row gap-x-[3.3%] max-md:flex-wrap
9                          max-md:gap-y-3">
10             {newsData.data.map(news => <NewsCard
11                 key={news.id}
12                 id={news.id}
13                 item={news.attributes}
14             />)}
15         </div>
16
17     )
18 }
```

Рисунок 3.20 – Програмний код компонента AllNews.

Для перевірки результатів виконання цього коду, потрібно відкрити головну сторінку веб-сайту за адресою “localhost:3000/”, там будуть відображені новини, отримані з серверу системи керування вмістом Strapi, що зображено на рисунку 3.21.

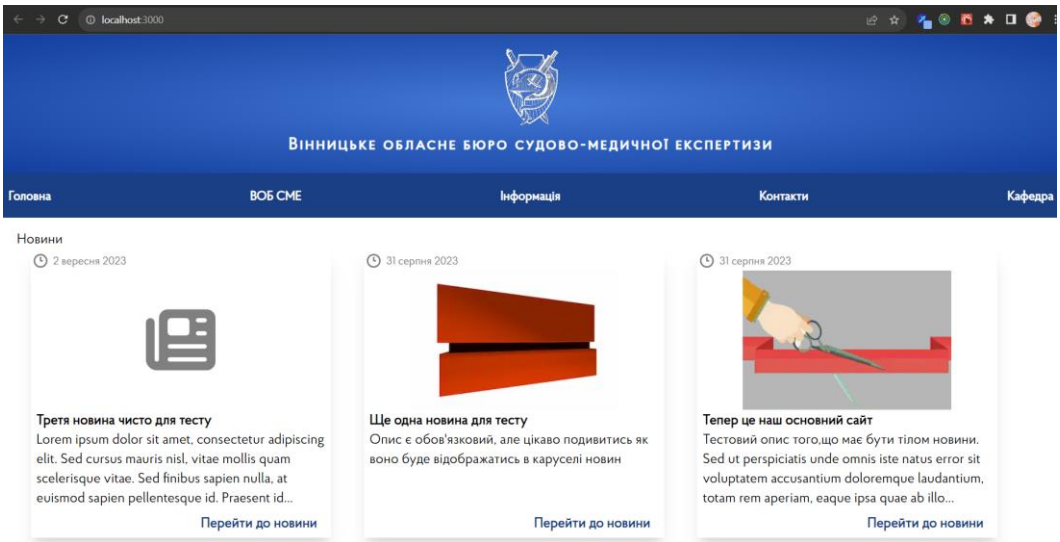


Рисунок 3.21 – Вигляд компоненту AllNews на головній сторінці сайту.

3.2.4. Розробка сторінок відділів та відділень.

Розробку сторінок відділів та відділень слід розпочати з розробки сторінки-шаблону, на яку буде посилатись кожен пункт відділів та відділень з навігаційної панелі Strapi. В Next.js це можна реалізувати впровадивши Dynamic Routes (Динамічні маршрути). Це маршрути, які створюються в випадках, коли розробник заздалегідь не знає точних імен сегментів і хоче створити маршрут з динамічних даних, тоді він може використовувати динамічні сегменти, які заповнюються під час запиту або попередньо відображаються під час створення.

Динамічні сегменти створюються огортанням імені директорії в квадратні дужки “[ ]”. Для прикладу, коли користувач переходить по посиланню http://localhost:3000/departments/viddilennia-sm-kryminalistyky з навігаційного меню, що відповідає пунктам «ВОБ СМЕ – Структура бюро – Відділення СМ криміналістики», то маршрут в директорії app/ повинен виглядати наступним чином:

app/departments/[department]/page.jsx .

, де “[department]” це динамічний сегмент, який може бути як “viddilennia-sm-kryminalistyky”, так і “viddil-sme-poterpilykh” чи “mizhraionni-viddilennia”. Такий підхід надає можливість створення шаблонної сторінки, дані для відображення якої будуть залежати від значення переданого в динамічний сегмент.

Створимо директорію departments/ в папці app/ проекту, а в ній директорію з назвою [department]/ з файлом page.jsx, в результаті шлях має виглядати так:

app/departments/[department]/page.jsx .

В page.jsx оголосимо асинхронну функцію Department, яка приймає в параметри об’єкт з полем params, що буде передаватись до функції фреймворком Next.js, та експортуємо цю функцію за замовчуванням. Далі потрібно отримати з серверу CMS Strapi, дані про вибраний відділ чи відділення, за полем slug. Для того щоб отримувати дані про відділення по полю slug, в частині з розробки логіки системи контролю вмістом Strapi було встановлено плагін “Slugify”, котрий дозволяє це зробити. Тому потрібно звернутись за посиланням сервера, на endpoint плагіну “Slugify”, вказуючи значення slug відділу чи відділення, дані про яке

потрібно отримати. Посилання на відділи відділення в навігаційному меню Header точно відповідають полям slug в системі контролю вмістом Strapi, тому значення для пошуку будуть братись з адресного рядка браузера користувача, до прикладу, якщо користувач перейшов в меню за посиланням на Відділення СМ криміналістики, то в адресному рядку буде відкрите посилання:

http://localhost:3000/departments/viddilennia-sm-kryminalistyky .

В цьому випадку виконається функція Department, яка повинна відрисовувати сторінку Відділення СМ криміналістики, де “ viddilennia-sm-kryminalistyky” – це значення поля slug в записі моделі department в CMS Strapi. Асинхронна функція Department отримує це значення як параметр, записаний в об’єкт params, в поле department системою маршрутизації фреймворку Next.js. Маючи ці дані слід створити запит на отримання інформації про відділення. Посилання для запиту буде виглядати наступним чином:

‘${process.env.STRAPI_API_URL}/slugify/slugs/department/${params.department}?po

pulate= head_of_department&populate=head_of_department.photo’ ,

де “${process.env.STRAPI_API_URL}” – це “http://localhost:1337/api”, “slugify/slugs” – кінцева точка (endpoint) плагіну Slugify, “department” – назва моделі, запис в якій потрібно знайти та повернути, “${params.department}” – той самий slug з адресного рядку браузера користувача, “populate=head_of_department” – також повернути інформацію про зв’язок з моделлю “head_of_department”, “populate=head_of_department.photo” – також включити до відповіді дані про поле “photo” зв’язаної моделі “head_of_department”.

Правильний запит для отримання достатньої відповіді сервера сформований, тепер потрібно записати цю інформацію в константи, в функції сторінки Department, щоб зручніше відрисувати їх та перетворити в JSX розмітку. Відповідь серверу запишемо в константу departmentData. Інформацію про завідуючого відділом/відділенням запишемо в константу headOfDepartment. Дані про фотографію завідуючого відділом/відділенням в констану headOfDepartmentPhoto.

Тепер потрібно правильно відобразити вміст поля description в HTML. Оскільки в моделі department системи контролю вмістом Strapi, полю description

присвоєно тип даних Rich Text, то значення в нього записане мовою Markdown, яку не розуміє сторінка HTML, тому потрібно це markdown значення перетворити в HTML теги. Напишемо функцію markdownToHTML(), що буде отримувати на вхід рядок з markdown, та повертати рядок перетворений в HTML. Створимо файл markdownToHTML.js в директорії lib/, що знаходиться в корені проекту. Для перетворення markdown в HTML існують бібліотеки “remark” та “remark-html”, встановимо їх виконавши в терміналі команду:

npm install remark remark-html .

Тепер, в функції markdownToHTML імпортуємо функцію remark() з бібліотеки “remark”, та функцію html() з бібліотеки remark-html. Створимо константу result, в яку запишемо результат виконання асинхронних функцій перетворення markdown на HTML: remark().use(html).process(markdown); , де markdown – вхідний параметр функції markdownToHTML. Програмний код функції markdownToHTML наведений на рисунку 3.22.

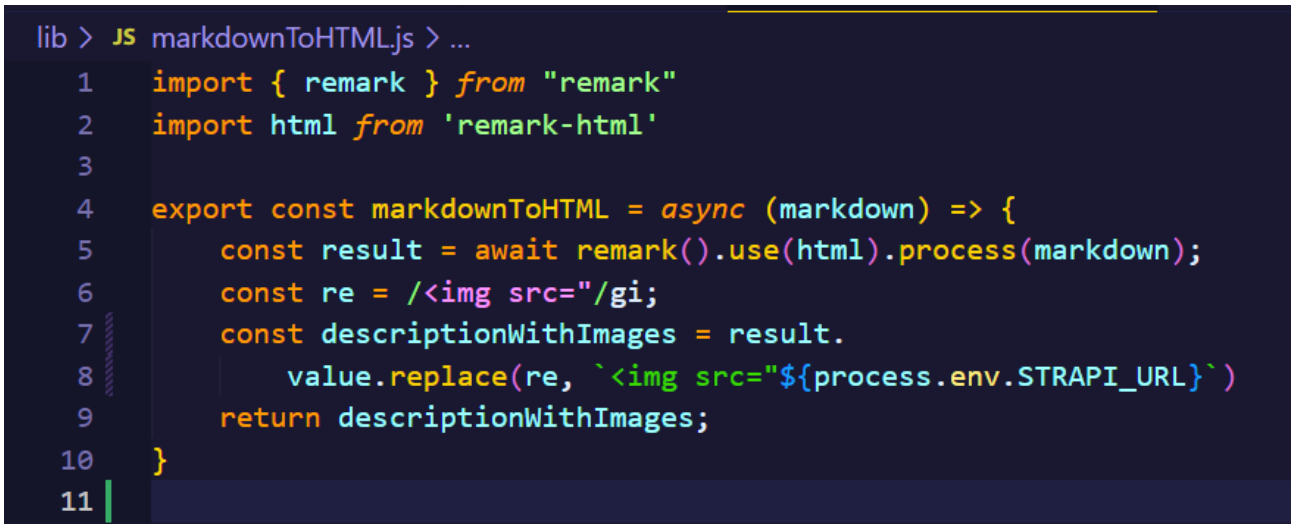


Рисунок 3.22 – Програмний код функції перетворення markdown на HTML.

Так як в описі відділення може бути присутня завантажена на сервер Strapi фотографія, то потрібно замінити посилання на неї, оскільки воно вказано як абсолютний шлях типу “/uploads/ім’я_файлу_зображення.розширення”, а в клієнтській частині додатку потрібне повне посилання вигляду “http://localhost:1337/uploads/ім’я_файлу_зображення.розширення ”. Тому в result

потрібно замінити всі “<img src=” ” на “<img src=”http://localhost:1337 ”. Адресу серверу системи контролю вмістом візьмемо з змінної локального середовища «STRAPI_URL». Запишемо результат пошуку та заміни символів в змінну descriptionWithImages та повернемо цей резуьтат з функції markdownToHTML().

Написавши функцію markdownToHTML(), потрібно застосувати її до поля description з відповіді сервера на запит інформації про відділ/відділення. Запишемо результат переведення в змінну description. Програмний код сторінки Department продемонстрований на рисунку 3.23.

app > departments > [department] > page.jsx > ...

```jsx
import DepartmentPage from "@/components/departmentPage/DepartmentPage";
import { fetcher } from "@/lib/api";
import { markdownToHTML } from "@/lib/markdownToHTML";

export default async function Department({ params }) {
    const departmentData = await fetcher(`${process.env.STRAPI_API_URL}/slugify/slugs/department/${params.department}
        ?populate=head_of_department&populate=head_of_department.photo`)
    const headOfDepartment = departmentData.data.attributes.head_of_department.data ?
        departmentData.data.attributes.head_of_department.data.attributes : undefined
    const headOfDepartmentPhoto = headOfDepartment !== undefined && headOfDepartment.photo.data.attributes
    const description = await markdownToHTML(departmentData.data.attributes.description)
    return (
        <DepartmentPage departmentData={departmentData} headOfDepartment={headOfDepartment}
            headOfDepartmentPhoto={headOfDepartmentPhoto} description={description} />
    )
}
```

Рисунок 3.23 – Програмний код сторінки Department.

Отримані дані заносяться в компонент <DepartmentPage />, яка є шаблоном для виводу інформації про відділення. Компонент DepartmentPage отримує як вхідні параметри аргументи departmentData, headOfDepartment, headOfDepartmentPhoto та description. Дані підставляються в HTML таким же методом як і зазвичай, та як раніше в проекті, але дані description заносяться в блок з описом відділу/відділення методом вставки HTML-розмітки до блоку «небезпечним методом», через атрибут “dangeruoslySetInnerHTML”. Тому що JSX розмітка не має методів .appendChild як в нативному JavaScript та буде вставляти HTML-розмітку як звичайний текст, що призведе до нечитабельності сторінки та неправильного відображення вмісту, також буде відсутня стилізація до цього вмісту. Метод вставлення HTML-розмітки в блок всередині розмітки JSX, методом «небезпечного вставлення HTML» [61] зображений на рисунку 3.24.

```
ents > departmentPage > DepartmentPage.jsx > DepartmentPage
import Link from "next/link";
import FontAwesomeEmail from "../fontAwesomeClientComponents/Email";
import FontAwesomePhone from "../fontAwesomeClientComponents/Phone";
import Image from "next/image";

export default function DepartmentPage({departmentData, headOfDepartment, headOfDepartmentPhoto, description}) {
    return (
        <main className="container mx-auto px-5 text-[black] font-probapro flex flex-col py-8">
            <h1 className="text-center font-semibold text-2xl text-headerSecond mb-6">
                {departmentData.data.attributes.title ? departmentData.data.attributes.title : departmentData.data.attributes.name}
            </h1>
            <div dangerouslySetInnerHTML={{__html: description}} className="dangerously-set-department"></div>

            {departmentData.data.attributes.head_of_department.data &&
            <>
            <h2 className="text-center font-semibold text-2xl text-headerSecond my-6">
                Завідувчий {departmentData.data.attributes.title ?
                    (departmentData.data.attributes.title.toLowerCase().includes('відділення')
                        ? "відділенням"
                        : "відділом") :
                    departmentData.data.attributes.name.toLowerCase().includes('відділення') ? "відділенням" : "відділом"}
            </h2>
            <div className="flex flex-row justify-evenly">
                <div className="flex flex-col font-probaprosmbd text-headerSecond gap-y-3">
                    <h2 className=" text-headerFirst text-xl mb-4 ">{headOfDepartment.name}</h2>
                    {headOfDepartment.contacts.tel && headOfDepartment.contacts.tel.map((tel, index) => <div key={index}
                                                                                                             className="flex flex-row gap-4 items-center">
                        <FontAwesomePhone />
                        <Link href={`tel:${tel}`} style={{letterSpacing: '1px'}}>{tel}</Link>
                    </div>)}
                    <div className="flex flex-row gap-4 items-center">
                        <FontAwesomeEmail />
                        <Link href={`mailto:${headOfDepartment.contacts.email}`}>{headOfDepartment.contacts.email}</Link>
                    </div>
                </div>
                <Image src={`${process.env.STRAPI_URL}${headOfDepartmentPhoto.url}`} width={300} height={400}
                       alt={headOfDepartmentPhoto.name} />
            </div></>}
        </main>
    )
}
```

Рисунок 3.24 – Програмний код компоненту DepartmentPage.

3.2.5. Впровадження в систему процесу авторизації.

Оскільки система ділиться на дві частини: перша – для користувачів, які не є авторизованими, та користувачів, котрі проходять авторизацію. Де для неавторизованих користувачів доступна лише частина сайту, на якій можна переглянути усю публічну інформацію, визначену вимогами до системи, а для авторизованих повинен бути створений особистий кабінет, в якому будуть здійснюватись процеси створення та керування направленнями. Тоді розробку інтерфейсу для частини сайту авторизованих користувачів потрібно почати з впровадження авторизації.

Для початку потрібно додати умову в відображення вмісту файлу layout.js, оскільки на сторінці авторизованого користувача не повинно бути навігаційне

меню в компоненті Header таке ж, як для неавторизованого користувача. Логіка процесу може виглядати таким чином: коли користувач потрапляє на сторінку localhost:3000/cabinet, то в layout.js потрібно відображати не компонент Header, а замінити його на компонент CabinetHeader, в якому здійснити перевірку, чи авторизований користувач, якщо користувач не авторизований, то відобразити поля введення даних для авторизації, а також кнопку «Увійти», якщо користувач авторизований – відобразити напис «Вітаємо, ім’я користувача», де «ім’я користувача» це “username” авторизованого користувача, і кнопку «Вийти», яка буде стирати дані авторизації.

Потрібно створити файл auth.js в директорії lib/ в корені проекту, де описати функції, необхідні для процесу авторизації. Процес авторизації буде побудований на збереженні отриманих даних користувача в файлах «Cookie» браузера. Блок схема алгоритму перевірки авторизації та процесу авторизації зображена на рисунку 3.25. В файлі auth.js потрібно описати такі функції:

- Функція setToken(). Функція повинна записувати отримані, при успішній авторизації JSON Web Token[62], username та id користувача з сервера системи контролю вмістом, в файли Cookie;
- Функція unsetToken(). Функція буде видаляти дані авторизації користувача з файлів Cookie;
- Функція getUserFromLocalCookie(). Функція повинна повертати інформацію про користувача з серверу, якщо дані авторизації містяться в файлах Cookie;
- Функція getIdFromLocalCookie(). Функція повинна повертати id користувача отриманий з серверу Strapi, якщо дані авторизації містяться в файлах Cookie;
- Функція getTokenFromLocalCookie(). Функція повинна повертати запис jwt (JSON Web Token) з файлів Cookie користувача;
- Функція getTokenFromServerCookie(). Функція повинна повертати новий JWT, якщо дані авторизації змінились та сервер надав відповідь з статусом 200, та новим JWT;

- Функція getIdFromServerCookie(). Функція повинна повертати новий id з серверу.

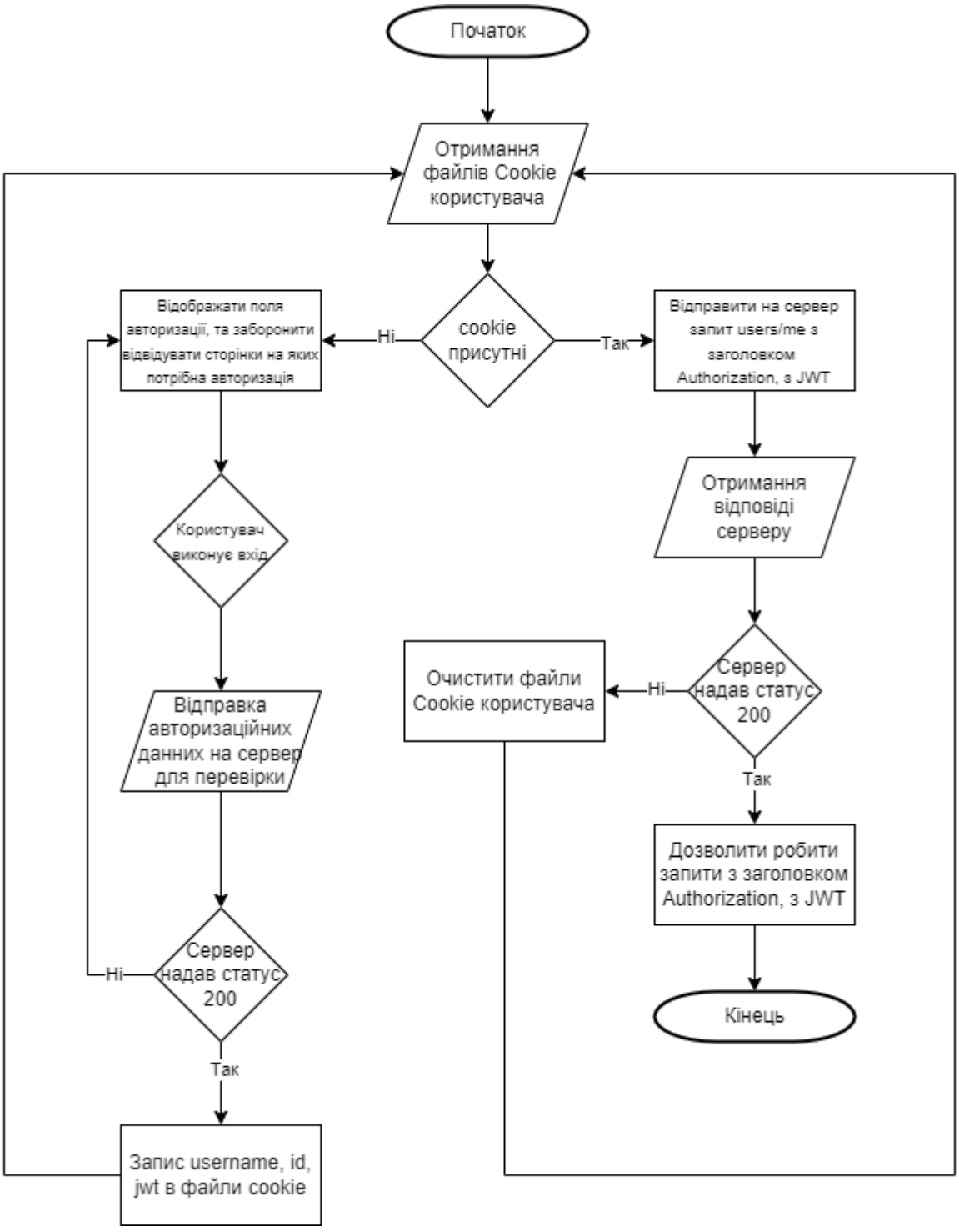


Рисунок 3.25 – Блок-схема алгоритму перевірки авторизації користувача.

Далі потрібно створити контекст React, який буде зберігати дані про користувача. Конктексти React створюються з метою доступу різних компонентів

до спільної інформації, необхідної для побудування логічного функціонування компонента та системи в цілому. В даному випадку, контекст буде потрібен для того, щоб брати інформацію про авторизованого користувача, етап завантаження даних з серверу, та в залежності від цих даних виводити вміст компонента, дозволяти чи забороняти робити запити, та будувати іншу логіку, в якій необхідне виконання умов авторизації. В директорії lib/ кореня проекту створимо файл authProvider.js, код якого зображений на рисунку 3.26.

lib > JS authProvider.js > useFetchUser

```
"use client"
import { createContext, useContext, useEffect, useState } from "react";
import { getUserFromLocalCookie } from "./auth";

let userState;
const User = createContext({user: null, loading: false});
export const UserProvider = ({value, children}) => {
    const {user} = value;
    useEffect(() => {
        if(!userState && user) {
            userState = user
        }
    }, [])
    return <User.Provider value={value}>{children}</User.Provider>;
}
export const useUser = () => useContext(User);
export const useFetchUser = () => {
    const [data, setUser] = useState({
        user: userState || null,
        loading: userState === undefined
    })
    useEffect(() => {
        if(userState !== undefined) {
            return
        }
        let isMounted = true;
        const resolveUser = async () => {
            const user = await getUserFromLocalCookie();
            if(isMounted) {
                setUser({user, loading: false})
            }
        }
        resolveUser();
        return () => {
            isMounted = false
        }
    }, [])
    return data;
}
```

Рисунок 3.26 – Програмний код контексту User.

В цьому коді створений контекст User, який повинен містити інформацію про користувача, та булеве значення loading яке вказує на те, чи завантажуються ще дані, чи вони вже завантажились повністю. Контекст має хук useFetchUser(), який встановлює значення user та loading в контекст User. Тільки як дані змінюються, хук useFetchUser() перезаписує дані user та loading.

Далі потрібно створити Server Actions[63]. Server Actions – це асинхронні функції, які виконуються на сервері. Їх можна використовувати в серверних і клієнтських компонентах для обробки надсилання форм і мутацій даних у програмах Next.js. Server Actions в Next.js це як посередник між користувачем, та сервером, на якому розміщена логіка фронт-енд частини додатку, суть цього посередництва заключається в тому, що запит на сервер системи контролю вмістом Strapi буде робитись не з браузера користувача, а з серверу Next.js. Таким чином, в браузері користувача не буде інформації про посилання на запит, що дасть додаткову безпеку. Створимо директорію actions, в якій будуть створюватись файли з функціями Server Actions. Першим файлом потрібно створити authorization.js з такими функціями:

- tryLogin(). Функція повинна приймати в параметри об’єкт з данними авторизації користувача, а саме username та password. Робити POST-запит з серверу Next.js на сервер системи контролю вмістом Strapi на кінцеву точку авторизації “auth/local”, в body якого потрібно помістити об’єкт з авторизаційними данними, переданими в параметр функції. Функція повертає відповідь сервера, при умові, що сервер не повернув помилку, інакше функція повертає помилку. Програмний код функції наведений на рисунку 3.27;
- authMeUsername(). Функція приймає в параметр JSON Web Token, виконує запит на сервер системи контролю вмістом Strapi на кінцеву точку “users/me” з URI-параметром “populate=*”, що включить в відповідь всі зв’язки запису User з іншими таблицями. Функція повертає username користувача, якщо сервер надав відповідь з статусом «200», інакше – функція повертає помилку;

- authMeId(). Функція приймає такий же параметр, та робить такий же запит як і функція authMeUsername(), але при успішній відповіді серверу, повертає id користувача;
- authMe(). Функція приймає в параметр JSON Web Token, робить такий же запит як і функції authMeUsername() та authMeId(), але при успішній відповіді серверу повертає всю відповідь серверу, при помилці – повертає помилку, та дублює цю помилку в консоль. Програмний код функції наведений на рисунку 3.28.

Також обов'язковою умовою створення Server Actions є використання директиви "use server" на першій лінії файлу, в якому будуть описані Server Actions.

```
export async function tryLogin (formData) {
    try {
        const response = await fetcher(
            `${process.env.STRAPI_API_URL}/auth/local`,
            {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                },
                body: JSON.stringify({
                    identifier: formData.login,
                    password: formData.password
                })
            }
        )
        const data = await response;

        return data;
    } catch (error) {
        console.error('Помилка при відправленні POST-запиту:', error);
        throw error;
    }

}
```

Рисунок 3.27 – Програмний код функції спроби відправки авторизаційних даних введених користувачем на сервер.

Далі, ці Server Actions потрібно використовувати в компонентах, де потрібно отримати чи перевірити статус авторизації користувача, змінити статус авторизації користувача, доповнити дані авторизації користувача. В моєму випадку слід використовувати функцію authMeId() в файлі auth.js, в функції getIdFromLocalCookie(), а функцію authMeUsername() в функції getUserFromLocalCookie(), при умовах, що присутній JSON Web Token, та з передачею його в параметри функцій.

```
export async function authMe(jwt) {
    try {
        return fetcher(
            `${process.env.STRAPI_API_URL}/users/me?populate=*`,
            {
                headers: {
                    'Content-Type' : 'application/json',
                    Authorization: `Bearer ${jwt}`,
                }
            }
        ).then((data) => {
            return data;
        }).catch((error) => {
            console.error(error)
        })
    } catch (error) {
        console.error('Помилка при відправленні POST-запиту:', error);
        throw error;
    }
}
```

Рисунок 3.28 – Програмний код функції authMe(), для отримання даних авторизованого користувача з серверу.

На цьому етапі, весь необхідний функціонал для створення компоненту, в якому буде здійнюватись авторизація користувача впроваджений, тому можна приступити до створення компоненту CabinetHeader, який буде відображатись лише в сторінках, що будуть відноситись до сторінок авторизованих користувачів. В директорії components/ створюємо папку CabinetHeader, а в ній файл

CabinetHeader.jsx, в якому оголошуємо функцію CabinetHeader, та експортуємо її за замовчуванням. Оскільки введення авторизаційних даних буде відбуватись на стороні клієнта, в браузері користувача, для чого буде використовуватись локальний стан useState, то потрібно впровадити директиву “use client” на першій стрічці файлу CabinetHeader.jsx, тому що хуки та контексти React працюють тільки на стороні користувача.

Компонент CabinetHeader повинен виводити вміст в залежності від статусу авторизації користувача. Якщо даних про авторизованого користувача не знайдено, то потрібно виводити текстові поля вводу логіну та паролю, також кнопку «Увійти». Для цього потрібно створити локальний стан formData – ініціалізаційними данними якого потрібно вказати об’єкт з пустими рядковими полями username та password:

```
const [formData, setFormData] = useState({username: ‘’, password: ‘’}); .
```

Також потрібно створити локальний стан з текстом помилки, який буде змінюватись, в випадку надходження помилки з серверу:

```
const [error, setError] = useState(undefined); .
```

Третій локальний стан буде зберігати булеве значення, яке буде використовуватися для того, щоб тримати відкритим вікно з помилкою авторизації, дамо йому назву isErrorDialogOpen, яке на момент ініціалізації буде мати значення false, оскільки потрібно відкривати вікно тільки в випадку, коли сервер повернув помилку:

```
const [isErrorDialogOpen, setIsErrorDialogOpen] = useState(false); .
```

Натискання на кнопку «Увійти» викличе внутрішню асинхронну функцію handleSubmit, яка візьме значення полів username та password з змінної локального стану formData, передасть їх як параметр до функції Server Action – tryLogin(), та викличе її. Якщо в результаті виклику функції tryLogin() повернеться помилка, потрібно вивести на екран модальне вікно, в якому буде вказаний текст помилки, та стерти значення з полів username та password.

Для цього потрібно визначити функцію loginError(), що буде приймати в параметр рядок – текст помилки авторизації, записувати до змінної локального

стану error текст помилки, змінювати змінну локального стану isErrorDialogOpen на true, що виконає умову відкриття модального вікна інформування про неуспішну авторизацію. Для модального вікна потрібно написати функцію closeErrorDialog(), що буде виконуватись в випадках закривання користувачем модального вікна інформування про помилку. Функція closeErrorDialog() повинна очищувати поля авторизації, шляхом обнулення змінної локального стану formData, також при закриванні модального вікна потрібно видалити текст помилки з локального стану, шляхом переприсвоєння значення на undefined для змінної error, та змінити умову відкритого стану модального вікна шляхом зміни значення true змінної локального стану isErrorDialogOpen на false. Функцію closeErrorDialog() слід додати до атрибуту onClose, та на обробник події onClick для кнопки «Закрити» в модальному вікні інформування про помилку. Модальне вікно буде побудоване з компонентів бібліотеки готових компонентів інтерфейсу користувача “Material UI/MUI”. JSX-розмітка модального вікна інформування про помилку наведена на рисунку 3.29.

```
<Dialog open={isErrorDialogOpen} onClose={closeErrorDialog}>
    <DialogTitle>Помилка авторизації</DialogTitle>
    <DialogContent>
        Помилка: <p className="text-red-600">{error?.message}</p><br />
        Введіть коректні данні!
    </DialogContent>
    <DialogActions>
        <Button onClick={closeErrorDialog}>
            Закрити
        </Button>
    </DialogActions>
</Dialog>
```

Рисунок 3.29 – Програмний код модального вікна інформування про помилку авторизації користувача.

У випадку, коли користувач введе правильні username та password, натисне на кнопку «Увійти», успішно виконається функція tryLogin(), а з серверу повернеться не помилка, а відповідь, то потрібно викликати функцію setToken(), та передати їй в параметр відповідь серверу, функція повинна встановити токени в

файли Cookie, та в разі успішного виконання повенути true. Тому це значення потрібно записати в константу isTokenSetted, та надалі виконати перевірку, якщо isTokenSetted має значення true, то перезавантажити сторінку. Перезавантаження сторінки потрібно для того, щоб отримати свіжі дані з Cookie з браузера користувача, оскільки вони не підтягуються автоматично. Програмний код функції handleSubmit наведений на рисунку 3.30.

```
const handleSubmit = async () => {
    const responseData = await tryLogin(formData)
    if(responseData.data === null) {
        loginError(responseData.error)
    } else {
        const isTokenSetted = setToken(responseData)
        if(isTokenSetted === true) {
            location.reload()

        }
    }
}
```

Рисунок 3.30 – Програмний код функції обробки надсилання користувачем авторизаційних даних на сервер в результаті натискання на кнопку «Увійти».

Також потрібно спростити зміну локального стану для змінної formData, для цього слід створити внутрішню функцію handleChange, що буде приймати в параметр елемент, який змінюють, брати з елемента атрибут name, в якому вказане значення ключа в formData, та значення елемента, так як функція буде застосовуватись до текстового поля, то буде братись значення, що введене в нього, та замінювати значення локального стану formData в полі, до якого застосовується зміна значення. Програмний код функції handleChange() наведений на рисунку 3.31.

Після того як внутрішню логіку компонента CabinetHeader влаштовано, потрібно описати те, що компонент буде повертати, оскільки компонент повинен повертати JSX-розмітку, опишемо цю розмітку в return(). Так як компонент повинен

повертати поля для вводу авторизаційних даних в тому випадку, коли користувач не авторизовний, і напис «Вітаємо, ім'я користувача» та кнопку «Вийти» коли користувач авторизований, то слід застосувати умовний рендеринг (відображення).

```
const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value })
}
```

Рисунок 3.31 – Програмний код функції handleChange() для полів введення авторизаційних даних користувача.

Дані про користувача містяться в контексті User. Потрібно отримати user та loading з контексту User, використаємо хук useFetchUser(), описаний в файлі authProvider.js написавши в коді компонента CabinetHeader таку стрічку коду:

const {user, loading} = useFetchUser(); .

Далі в return() компонента огортаємо вміст компонентою <UserProvider> в атрибут value якого передаємо об'єкт з user та loading, отриманих з контексту User, далі виконуємо перевірку умов з допомогою тернарних операторів, оскільки виконання перевірки умов в return за допомогою конструкцій типу if-else не є можливим. Потрібно виконати перевірку чи завантались всі дані та чи відсутні дані користувача, якщо так, то відображаємо поля для введення username та password, кнопку «Увійти». Якщо умова перевірки – хиба, то відобразити кнопку виходу з облікового запису, що видалить авторизаційні дані користувача з файлів Cookie, та напис «Вітаємо, Ім'я Користувача». Програмний код перевірки умов в return компонента CabinetHeader наведений на рисунку 3.32.

Наступним, та завершальним кроком в впровадженні авторизації до системи, буде додавання умовного рендерингу для обгортки сторінок веб-застосунку, файлу layout.js. Умовний рендеринг – відображення елементів розмітки JSX, які відповідають умові. В цьому випадку, в додатку, потрібно відображати компоненту Header на всіх сторінках системи, окрім сторінки cabinet/, тобто, коли користувач вводить до адресного рядка браузера посилання localhost:3000/cabinet, потрібно

відображати не компоненту Header та Footer між якими вставлена сторінка, шо відповідає маршруту, як на всіх маршрутах додатку, а компоненту CabinetHeader та сторінку, що відповідає маршруту cabinet/, та всім дочірнім маршрутам.

```
<div className="flex flex-row gap-3">
    <div className="flex flex-row items-center gap-5">
        {!loading && (user ? (
            <li className="list-none">Вітаємо, {user}</li>
        ) : (''))}
        {!loading && (user ? (
            <Button variant="contained" onClick={handleLogout}>Вийти</Button>
        ) : (''))}
    </div>

    {!loading && !user ? (
        <>
            <TextField className="bg-[white] rounded" size="small" label='Логін' variant="filled"
                name="login" value={formData.login} onChange={handleChange} />
            <TextField className="bg-[white] rounded" type="password" size="small" label="Пароль"
                name="password" variant="filled" value={formData.password} onChange={handleChange} />
            <Button variant="contained" onClick={handleSubmit}>Увійти</Button>
        </>
    ) : ""}

</div>
```

Рисунок 3.32 – Умовний рендеринг (відображення) в компоненті CabinetHeader.

Для цього потрібно змінити файл layout.js, додавши в функцію перевірку маршруту адресного рядка. Щоб перевірити чи користувач знаходиться саме за маршрутом cabinet/ потрібно взяти поле params з об’єкту-параметру функції RootLayout, в нього взяти об’єкт props, з якого перейти до об’єкту childProps, та взяти в ньому значення з поля segment. Поле segment має тип даних – рядок, що є прийнятним, оскільки можна надати в перевірку рядкове значення. Для реалізації потрібно використати перевірку умови конструкцією if-else. Якщо сегмент адресного рядка відповідає рядку “cabinet”, то повернути JSX-розмітку з тегом html, в який вкладено тег body, в який вкладено компоненту <CabinetHeader> та {children} під нею, де children – сторінка, яка огортається обгорткою RootLayout.

У всіх інших випадках, користувачем буде відкрито не сторінку за маршрутом “cabinet/” тому потрібно відображати не компонент з авторизацією, а стандартний компонент для неавторизованого користувача з навігаційною панеллю сайту. Для цього потрібно повернути JSX розмітку з тегом html, в який

вкладено тег body, в який вкладено компоненти <Header> і <Footer>, та {children} між ними.

Програмний код файлу обгортки для сторінок веб-застосунку системи layout.js наведений на рисунку 3.33.

```
export default function RootLayout({children}) {
  if(children.props.childProp.segment !== "cabinet") {
    return (
      <html Lang="en">
        <body className={inter.className + " " + "min-h-screen flex flex-col
          <Header />
          {children}
          <Footer />
        </body>
      </html>
    )
  } else {
    return (
      <html Lang="en">
        <body className={inter.className + " " + "min-h-screen flex flex-col
          <CabinetHeader />
          {children}
        </body>
      </html>
    )
  }

}
```

Рисунок 3.33 – Програмний код файлу layout.js.

В браузері компонент CabinetHeader буде мати вигляд зображений на рисунку (3.34 – коли користувач не авторизований, 3.35 – коли користувач авторизований під обліковим записом з username expert).

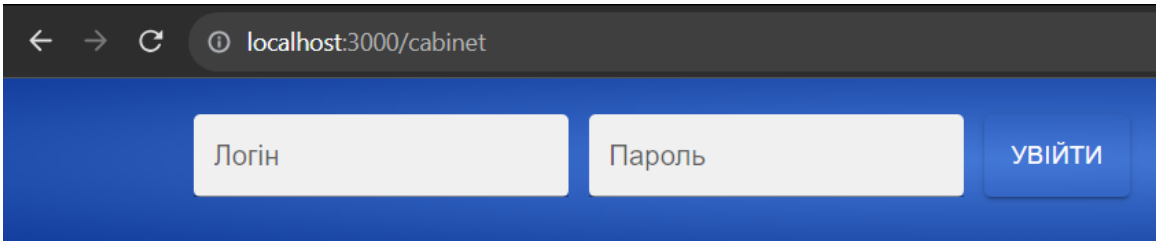


Рисунок 3.34 – Вигляд компоненту CabinetHeader при умові що користувач не авторизований.

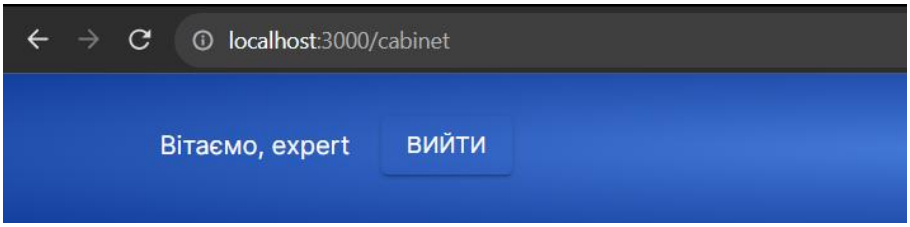


Рисунок 3.35 – Вигляд компоненту CabinetHeader при умові що користувач авторизований під обліковим записом expert.

3.2.6. Створення особистих кабінетів авторизованих користувачів.

Коли в систему впроваджено авторизацію слід створити кабінети для судово-медичного експерта та завідуючого відділенням/відділом. Першим кроком потрібно створити сторінку за маршрутом cabinet/. Для цього потрібно створити папку «cabinet» в директорії app/, а в ній файл page.jsx. В ній запровадити перевірку чи авторизований користувач, щоб унеможливити потрапляння неавторизованих користувачів до кабінету авторизованого користувача. Це можна зробити використовуючи хук useFetchUser(), беручи з об’єкта що повертається поля user та loading. Також для перевірки авторизованості користувача потрібно отримати дані про користувача, котрий переглядає сторінку. Для того щоб маніпулювати цими данними та будувати подальшу логіку, потрібно запровадити три нових локальних змінних:

```
const [userRole, setUserRole] = useState(undefined);
const [authUserData, setAuthUserData] = useState(undefined);
const [isRoleLoading, setIsRoleLoading] = useState(true); .
```

, де userRole – роль користувача, надана йому в системі контролю версій Strapi, можливі варіанти: “expert” або “headOfDepartment”, ініціалізаційне значення undefined, оскільки при вході на сторінку веб-застосунку ще невідомо чи користувач авторизувався, та яка в нього роль. authUserData – дані про авторизованого користувача, ініціалізаційне значення undefined, в процесі подальшої логіки сторінки ця змінна локального стану буде перевизначена. isRoleLoading – змінна локального стану, котра буде служити індикатором до

початку відрисовування вмісту компоненти, якщо isRoleLoading має значення true, то це означає що дані з серверу системи контролю вмістом не завантажились, тому не можна відрисовувати компоненту, оскільки це може потягнути за собою виникнення помилок. Коли змінні локального стану визначені потрібно отримати дані про користувача що переглядає сторінку, та вносити їх до локального стану.

Щоб розпочати першу ітерацію отримання даних потрібно використати хук useEffect, цей хук дозволяє виконувати побічні дії не порушуючи концепції чистих функцій. Коли користувач відкриває сторінку, отримується локальний стан, тобто всі локальні змінні, хук useEffect дозволяє відслідкувати цей запис, та виконати якісь дії коли відбувається зміна значень передана в масиві залежностей, які хук отримує в якості другого параметру до виклику. На даному етапі потрібно відслідковувати зміну оголошених для сторінки змінних локального стану. В тілі хука useEffect потрібно оголосити функцію fetchData, котра буде робити запит на отримання даних про користувача з допомогою Server action authMe, котрий знаходиться в особистому кабінеті, при умові що username користувача записаний до Cookie та в данний момент не відбувається процес авторизації, додатково перевірити чи в змінній userRole не записана роль користувача, відповідь від сервера занести в змінну локального стану authUserData, в змінну локального стану занести значення role з відповіді, та коли всі дані отримані та записані по відповідних місцях, змінити значення змінній isRoleLoading на false.

Оскільки процес завантаження даних завершився. Також після оголошення функції fetchData, її потрібно відразу викликати. В масив залежностей, в другий параметр функції хука useEffect потрібно передати змінні user, userRole, loading та jwt. Це означає, що перший параметр хука useEffect буде виконуватись в разі зміни значення однієї із переданих в масив залежностей змінних. Програмний код хука useEffect, що використовується для сторінки за маршрутом cabinet/ наведений на рисунку 3.36.

```
useEffect(() => {
    const fetchData = async () => {
        if (user && !loading) {
            if (!userRole) {
                const response = await authMe(jwt);
                setAuthUserData(response);
                setUserRole(response.role.name);
                setIsRoleLoading(false)
            }
        }
    };
    fetchData();
}, [user, userRole, loading, jwt]);
```

Рисунок 3.36 – Програмний код хука useEffect для сторінки Cabinet.

Наступним кроком буде запровадження логіки відображення JSX розмітки особистого кабінету. Оскільки за маршрутом cabinet/ може знаходитись особистий кабінет судово-медичного експерта або кабінет завідуючого відділенням, то потрібно запровадити умовне відображення (умовний рендеринг) JSX розмітки та компонент додатку. Зробимо перевірку чи не пусті дані користувача, що повернуті в результаті використання власного хуку useFetchUser(), та чи змінна loading з результату виклику цього ж хуку дорівнює false. Інакше вивести напис «Для доступу до сторінки потрібно авторизуватись!», що свідчить про те, що користувач, який відвідує сторінку не виконав авторизацію. Якщо ж змінна user має значення не null та не undefined, а змінній loading присвоєно значення false, то слід перейти до іншої перевірки, чи записана в змінну локального стану userRole роль користувача і чи змінній локального стану isRoleLoading присвоєно значення false, що свідчить про закінчення процесу завантаження даних про користувача з серверу та запису його ролі до даних компонента. Якщо умова хибна, то вивести напис «Завантаження даних...», якщо умова істина, перейти до фінальної умови, де буде перевірятись роль користувача, в залежності від якої потрібно виводити вміст. Для перевірки ролі користувача краще не використовувати конструкцію if-else, натомість краще використати конструкцію switch-case, де в параметри switch

передається змінна, очікуємі значення якої будуть міститись напроти ключового слова case. Так як вміст потрібно виводити в залежності від ролі користувача, в switch на перевірку передаємо змінну локального стану userRole, для case “expert” задаємо виведення компоненти <ExpertCabinet /> яка буде відображати кабінет судово-медичного експерта. Для case “headOfDepartment” задаємо виведення компоненти <HeadOfDepartmentCabinet />, в якій буде написано функціонал та JSX-розмітка для кабінету завідуючого відділом/відділенням.

На цьому етапі робота з файлом page.js в директорії app/cabinet/ завершена. Наступним кроком буде написання компонента <CabinetExpert />. В кабінеті судово-медичного експерта має бути такий функціонал та виведення такого вмісту:

1) Розділ отриманих направлень. Якщо судово-медичний експерт не отримував направлень, то потрібно вивести в цей розділ напис «Направлень не призначено!». В розділі повинні відображатись направлення, котрі надійшли експерту для обробки. Кожне направлення судово-медичний експерт може переглянути. Коли направлення тільки надходить до експерта, воно з’являється в розділі отриманих направлень, коли експерт обере це направлення натисканням миші на відповідну кнопку, повинно відкритись модальне (поверхневе) вікно в якому міститься інформація про назву направлення, файл-документ направлення, відділення/відділ з якого надійшло направлення. Також в модальному вікні є графа з станом направлення, воно може бути «надіслане», «отримане», «в обробці» та «з відповіддю». Стан направлення змінюється наступним чином: коли направлення надходить до експерта в кабінет, він також повинен отримати об’єкти для досліджень. Коли судово-медичний експерт фізично отримав об’єкти для досліджень, він відкриває електронне направлення та натискає кнопку «Об’єкт(и) отримано», відразу після цієї дії стан направлення змінюється на «отримане», а з модального вікна зникає кнопка «Об’єкт(и) отримано». Коли направлення має статус «отримане», то в модальному вікні з’являється кнопка «Почати роботу», що змінить стан направлення на «в

обробці». Це потрібно для того, щоб тому, хто направив направлення було видно, що робота за направленням вже почалась. При стані направлення «в обробці» зникає кнопка «Почати роботу» але з'являється кнопка «Додати висновок», натиснувши на яку, користувачеві буде запропоновано обрати файл з розширеннями ".pdf, .doc, ,docx", коли користувач обере файл-документ, в модальному вікні розблокується кнопка «Відправити», натискання на яку спричинить відправку документу висновку судово-медичного експерта на сервер системи контролю вмістом, та змінить стан направлення на «з відповіддю».

2) Розділ відправлених направлень. Якщо судово-медичний експерт не направляв направлень, то розділ має просто залишись пустим. Якщо користувач відправляв направлення, то вони виводяться в список розділу. В кожній стрічці міститься направлення з вказаною інформацією про номер направлення, відділ/відділення в яке надіслане направлення, стан (статус) направлення, та в випадку стану направлення «з відповіддю» - активна кнопка «Відкрити», натискання на яку спровокує відкривання модального вікна направлення, в якому можна завантажити документ-висновок, який є відповіддю по направленню. В правому куті розділу повинна бути кнопка «Створити нове +», натискання на яку спричинить відкривання модального вікна створення нового направлення. Модальне вікно повинно містити такі поля для введення даних: текстове поле направлення, в якому судово-медичний експерт повинен ввести номер направлення; поле вибору відділу/відділення на яке буде надіслане направлення; поле вибору файлу-документу направлення, дозволено обрати один документ з розширеннями ".pdf, .doc, .docx". Коли всі поля заповнені, повинна розблокуватись кнопка «Відправити», яка запустить функцію відправки відправлення до системи контролю вмістом Strapi. Також в модальному вікні є кнопка «Закрити», натискання на яку спричинить закривання модального вікна та очищення даних всіх полів.

Розробку функціоналу кабінету користувача-експерта слід розпочати з

отримання з серверу системи контролю вмістом Strapi даних про відправлені судово-медичним експертом направлення. На вхід, компонент отримує об’єкт з полями: jwt – JSON Web Token для того, щоб надсилати запити до серверу з підписом авторизації; user – ім’я корисувача авторизованого в кабінеті судово-медичного експерта. Також потрібно ініціалізувати змінні локального стану: authUserAllData – всі дані авторизованого користувача; departmentsList – список відділів/відділень; sendedReferralsList – список надісланих направлень; receivedReferralsList – список отриманих направлень. Як ініціалізаційне значення всім змінним локального стану потрібно присвоїти значення undefined. Наступним кроком буде застосування хуку useEffect, в перший параметр якого потрібно передати callback (функцію зворотнього виклику) в якій потрібно визначити та викликати функцію fetchData, яка буде перевіряти чи надійшли до компонента дані user та чи значення змінної authUserAllData дорівнює undefined. Якщо умова правдива, потрібно отримати з серверу системи керування вмістом Strapi дані про користувача, який користується кабінетом (для цього скористаємося server action getAllAuthUserData, який робить запит на адресу localhost:1337/api/cabinet-experts/${id}?populate=*, де id – id авторизованого користувача-експерта отримане з файлів Cookie), список відправлених направлень, список отриманих направлень та список відділень, та присвоїти їх відповідним змінним локального стану. В другий параметр хука useEffect потрібно передати масив залежностей, який повинен містити всі змінні локального стану. Таким чином буде забезпечене отримання всієї необхідної для компоненту інформації.

Оскільки перевірка актуальності авторизаційних даних проводилась в компоненті, з якого викликаний компонент ExpertCabinet, то проводити повторну таку перевірку в цьому компоненті не має сенсу. Але слід провести перевірку для виведення даних. Потрібно виводити дані в веб-застосунок тільки тоді, коли вони отримані. Тому потрібно ввести умову: якщо одна із змінних локального стану має значення undefined, то на сторінку кабінету судово-медичного експерта потрібно вивести напис «Завантаження даних...», для того, щоб проінформувати користувача про необхідність дочекатися повного завантаження (для випадків коли

користувач використовує повільний інтернет зв’язок). Якщо всі змінні локального стану мають правильні значення (не undefined), то відрисувати компоненти <ReceivedReferrals> та <SendedReferrals>.

Наступним кроком розробки кабінету судово-медичного експерта буде розробка компоненту ReceivedReferrals, який буде отримувати на вхід список відправлених експерту направлень, відображати їх, та дозволяти редагувати їхній стан (статус), а в разі стану «в обробці» надати змогу додати документ-відповідь на направлення. Щоб список отриманих направлень міг коригуватися без перезавантаження сторінки, коли судово-медичний експерт змінить стан направлення, потрібно запровадити змінну локального стану referralList, ініціалізаційним значенням якого буде список направлень, що переданий параметром вхідних даних до компонента. Вивести список направлень потрібно саме з змінної локального стану, а не з змінної вхідних даних. В разі зміни експертом стану направлення, буде змінюватись змінна локального стану referralList, а не вхідна змінна receivedReferralList, так бібліотека React зможе відслідкувати момент, коли потрібно перерисувати вміст компоненти, який залежить від даних що змінились, без перезавантаження сторінки.

Стан направлення повинен змінюватись в модальному вікні. Умова відкриття модального вікна – натиснута кнопка «Відкрити» напроти направлення. Умова доступності кнопки (розблокований стан) – стан має значення всі крім «з відповіддю». В модальне вікно виводиться: номер направлення; прізвище, ім’я, по-батькові та відділ/відділення експерта, який відправив направлення; стан направлення, від якого залежить функція та напис кнопки зміни стану; якщо стан направлення «в обробці», то – в модальному вікні з’являється поле вводу з типом «файл», дозволені розширення – “.pdf, .doc, .docx”. Модальне вікно для направлення з станом «в обробці» зображене на рисунку 3.37.

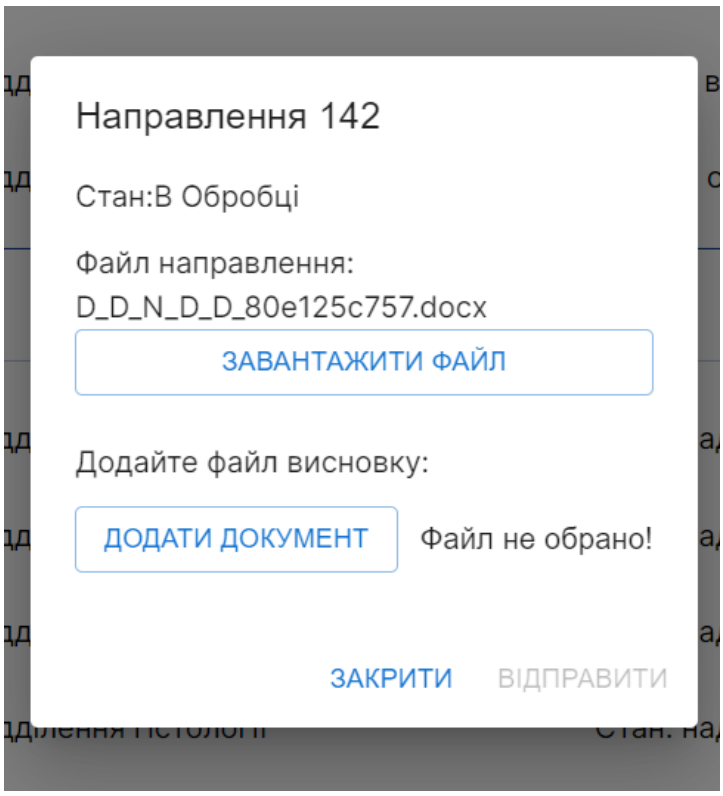


Рисунок 3.37 – Модальне вікно отриманого направлення з станом «в обробці»

Кнопка «Відправити» з’являється тільки тоді, коли направленню присвоєний статус (стан) «в обробці», і стає активна тоді, коли користувач завантажив документ-відповідь для відправки. Відправлення статусу відбувається з допомогою Server Action, котрий робить PUT-запит на сервер системи контролю вмістом Strapi, у випадку зміни стану направлення на «з відповіддю», використовується Server Action, який виконує POST-запит на кінцеву точку плагіну Uploads, що встановлений в систему керування вмістом, з відправленням прикріпленого файлу, і в разі успішного відправлення, повертається посилання на файл на сервері. Далі це посилання вставляється в поле responseDocument для направлення та відправляється PUT-запитом на кінцеву точку направлення до якого застосовується зміна. Компонент ReceivedReferrals відображений в кабінеті судово-медичного експерта має вигляд зображений на рисунку 3.38.

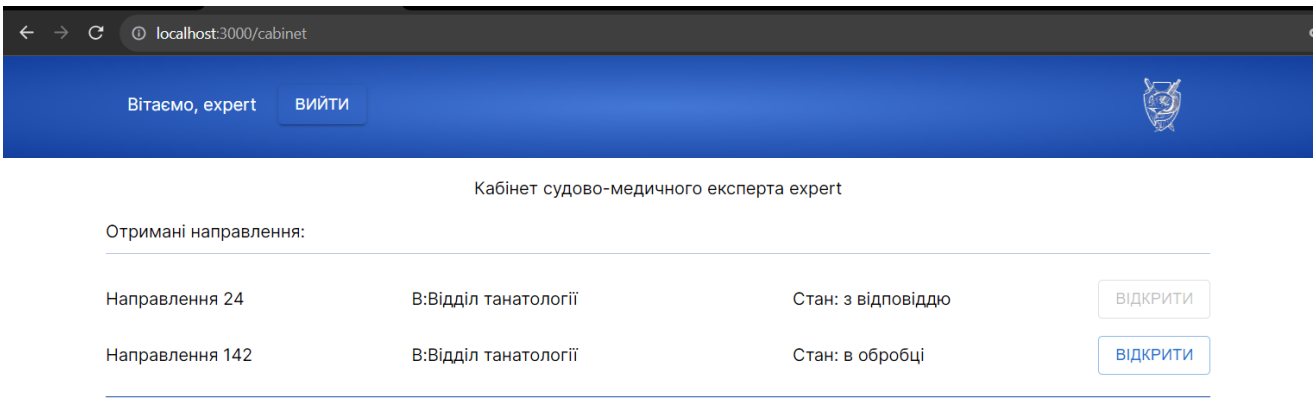


Рисунок 3.38 – Компонент ReceivedReferrals відображений в веб-застосунку, в кабінеті користувача.

Наступним кроком розробки кабінету судово-медичного експерта повинен бути крок розробки компонента SendedReferrals. Компонент повинен приймати в вхідні параметри список направлень, список відділів/відділень, дані про користувача, JSON Web Token користувача. Задача компонента – відображати список направлень, що надіслав користувач. Також компонент повинен мати функцію створення нового направлення, для якого повинен бути вказаний номер, прикріплений файл-документ, зі списку обране відділення/відділ, в який потрібно надіслати направлення. Створення направлення буде відбуватись в модальному вікні. Дані з модального вікна повинні відправлятись за допомогою Server Actions, при успішній перевірці на стороні клієнту, та змінюватись в списку направлень без перезавантаження сторінки. Ще однією функцією компонента буде можливість відкривання модального вікна для відправленого направлення, на яке прийшла відповідь, де користувач може переглянути стан направлення, та завантажити документ-відповідь Потрібно створити всі необхідні змінні локального стану та внутрішні функції компонента, для його правильного функціонування.

Компонент повинен мати такі змінні локального стану:

- sendedReferralsList – список направлень. Ініціалізаційне значення – список направлень переданий в компоненту як аргумент;
- departmentSelectedToSend – вибране відділення для відправки направлення. Стартове значення – undefined;

- isNewReferralDialogOpen – прапор відкривання модального вікна. Стартове значення – false;
- referralTitleData – значення поля введення номеру направлення. Ініціалізаційне значення – пустий рядок;
- DEFAULT_FILE_NAME – константа, що використовується для відображення коли не обрано файл для направлення, також використовується при перевірці умови чи вибрав користувач файл. Стартове значення – рядок: “Файл не обрано”;
- file – обраний файл для відправки. Ініціалізаційне значення – об’єкт з полями file та filename з значеннями null;
- isSendedReferralDialogOpen – прапор відкривання модального вікна відправленого раніше направлення. Ініціалізаційне значення – false;
- selectedSendedReferral – вибране направлення, на яке надійшла відповідь. Стартове значення – undefined.

Також наведу перелік функцій, які повинен містити компонент:

- setFileData – функція зміни даних про вибраний файл;
- closeAddDialog – функція, що викликається для закривання модального вікна для створення нового направлення;
- handleOpenSendedDialog – функція для обробки відкриття модального вікна раніше надісланого направлення;
- closeSendedDialog – функція, що обробляє закривання модального вікна раніше надісланого направлення;
- handleSubmitReferral – функція, що обробляє відправку новоствореного направлення.

На рисунку 3.39 в нижній частині екрану наведений компонент SendedReferrals, що відображений в кабінеті користувача.

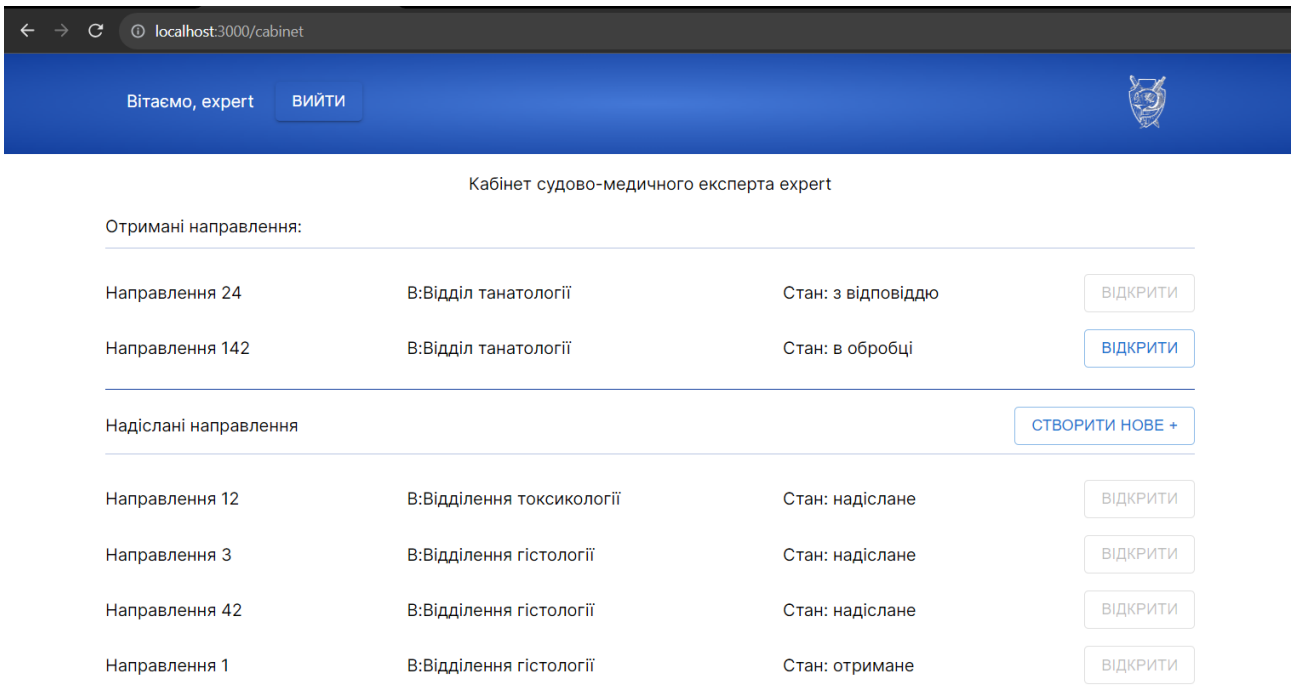


Рисунок 3.39 – Компонент SendedReferrals, відображений в кабінеті користувача.

Результат розробки модальних вікон для роботи з направленнями наведений на рисунках 3.40 – 3.41.

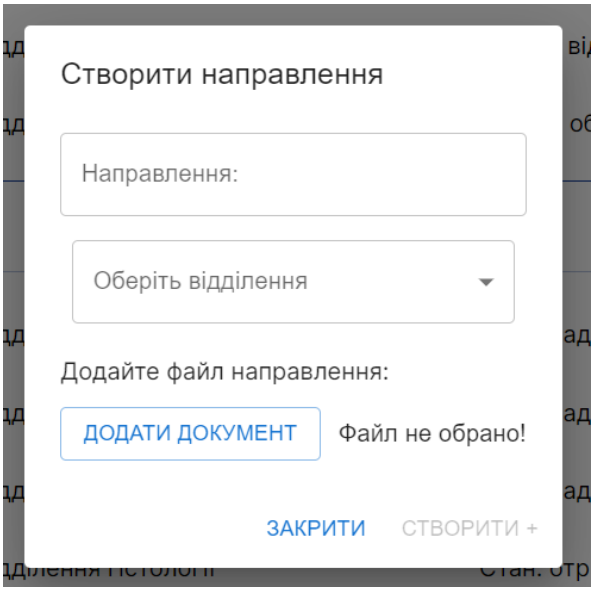


Рисунок 3.40 – Модальне вікно створення нового направлення з пустими полями введення.

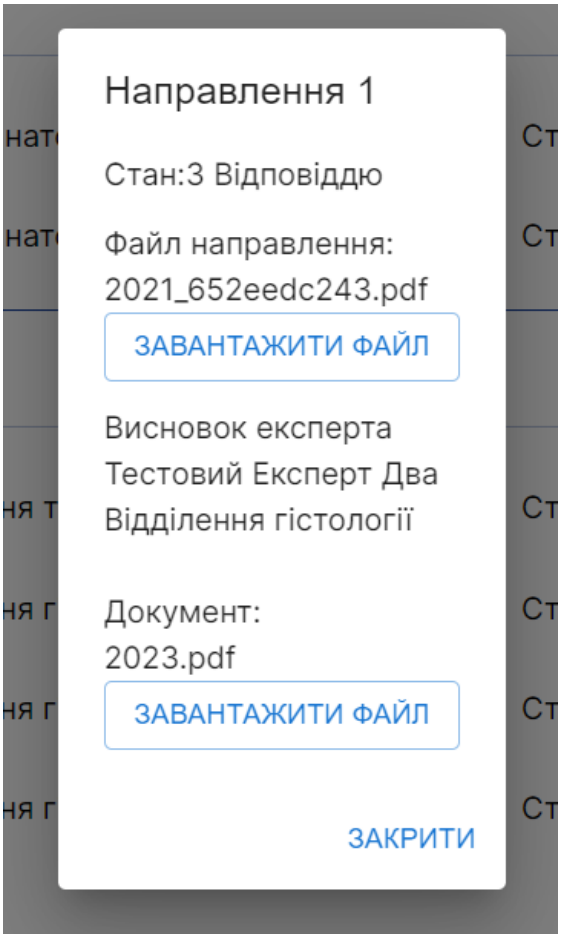


Рисунок 3.41 – Модальне вікно перегляду відповіді на направлення.

Фінальним етапом розробки системи буде впровадження кабінету завідуючого відділом/відділенням. Для реалізації потрібно створити компонент HeadOfDepartmentCabinet. Компонент буде приймати в вхідні параметри user та jwt (JSON Web Token) з компонента чи сторінки, де викликається. Функціоналом головної компоненти завідуючого відділом/відділенням буде розподілення направлень між експертами, які йому підпорядковуються, та можливість перегляду стану направлень, над якими працює відділ/відділення.

Компонент буде мати одне модальне вікно – вікно вибору судово-медичного експерта з списку відділу/відділення, котрий буде працювати над направленням. Для цього компонент має отримати список експертів з системи керування вмістом Strapi та записати їх в змінну локального стану departmentExperts. Пошук експертів які належать до відділення буде відбуватись шляхом відправлення GET-запиту на

кінцеву точку cabinet-experts з вказанням фільтру по полю “department”, значення якого повинно відповідати значенню id відділу/відділення, яким керує завідувач. Для цієї цілі, в інформації про користувача з роллю headOfDepartment є поле department, звідки і треба отримати значення. Щоб його отримати, потрібно зробити запит на повну інформацію про авторизованого в кабінеті користувача завідувача відділом/відділенням, та отримати поле id з об’єкту department.

Щоб отримати список направлень, які отримувало відділення, потрібно зробити запит на кінцеву точку referrals API Strapi з використанням фільтру recipient_department, значення якого повинно братись з authUserAllData.department.id та відповідати відділу, для якого надіслані направлення.

Надсилання направлення для роботи судово-медичному експерту буде відбуватись методом зміни полів для referrals, а саме з використанням PUT-запиту, що буде додавати id експерта, котрому буде доручена робота в поле recipient.
В списку направлень повинна бути індикація невизначених направлень, тобто тих, які не розподілені між експертами, тому при умові, що направлення не має даних в полі recipient, лінія з ним в списку буде підсвічуватись світло-червоним кольором. І тільки при умові, що направлення не має прямого отримувача, а має тільки відділення, котре отримало це направлення, буде доступна для використання кнопка «Відкрити», яка спровокує відкривання модального вікна призначення експерта. Готове модальне вікно для додавання направлення в роботу експерта зображене на рисунку 3.42.

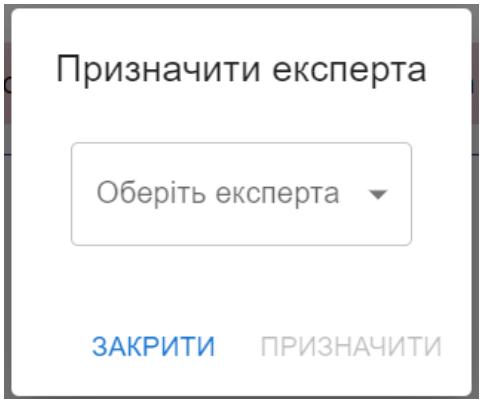


Рисунок 3.42 – Модальне вікно призначення направлення для експерта.

Готовий та відображений на сторінці компонент HeadOfDepartmentCabinet зображений на рисунку 3.43.

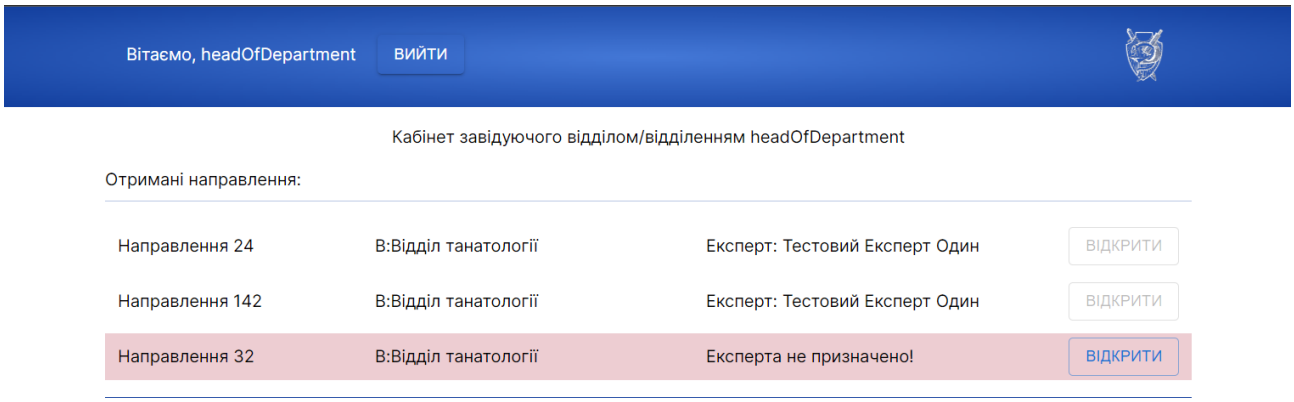


Рисунок 3.43 – Компонент HeadOfDepartmentCabinet на сторінці кабінету в обліковому записі завідуючого відділом/відділенням.

3.3. Тестування роботи системи.

Оскільки основою будь-якої автоматизованої системи є база даних, потрібно протестувати доступ до неї, щоб унеможливити витік даних в невизначені для цього джерела. В системі керування вмістом Strapi створені такі ролі:

- Public. Роль яка надається будь-якому неавторизованому користувачеві. Її призначення – перегляд загальнодоступних даних системи, таких як публічна інформація про діяльність відділів та відділень, керівників цих відділень, години роботи установи, контактна інформація для населення, публічна інформація про фінансову діяльність установи, яка повинна публікуватись згідно Господарського кодексу України;
- Expert. Роль, яка надається користувачеві, що авторизований під обліковим записом судово-медичного експерта. Використання цієї ролі передбачує доступ до всіх даних, що може отримати роль Public, але додається доступ на перегляд інформації типу колекцій cabinetExpert, cabinetHeadOfDepartment, cabinetDepartment, доступ на перегляд, створення та оновлення типу колекцій referrals, для роботи з

направленями, також передбачений доступ до створення uploads, для розміщення файлів-документів;

- HeadOfDepartment. Роль, яка надається користувачеві, який авторизований під обліковим записом завідуючого відділом/відділенням. Використання цієї ролі дає доступ до всіх даних, що можуть отримувати ролі Public та Expert, але завідувач відділом/відділенням має доступ до модифікації типу колекції cabinet-expert.

Для тестування ролі Public, потрібно надсилати GET-запити без заголовку Authorization, в який передається Bearer-токен (JSON Web Token). Слід зробити запити[64] на кінцеві точки:

- /api/slugify/slugs/:modelName/:slug – для пошуку по базі даних з плагіном Slugify,
- /api/bureau-chief,
- /api/departments,
- /api/departments/:id,
- /api/fininfos,
- /api/fininfos/:id,
- /api/head-of-departments,
- /api/head-of-departments/:id,
- /api/idepartments,
- /api/idepartments/:id,
- /api/newss,
- /api/newss/:id.

Тестові запити можна виконувати, використовуючи безкоштовний інструмент «ReqBin»[65], що зображений на рисунку 3.44. Слід вказати посилання запиту, кінцеву точку, обрати тип запиту (в цьому випадку GET), та в пункті Authorization обрати No Auth, оскільки звичайний відвідувач не повинен мати даних авторизації. Після введення всіх даних натискаємо на кнопку «Send», далі змінюємо кінцеву точку, на наступну з списку, і так для кожної кінцевої точки.

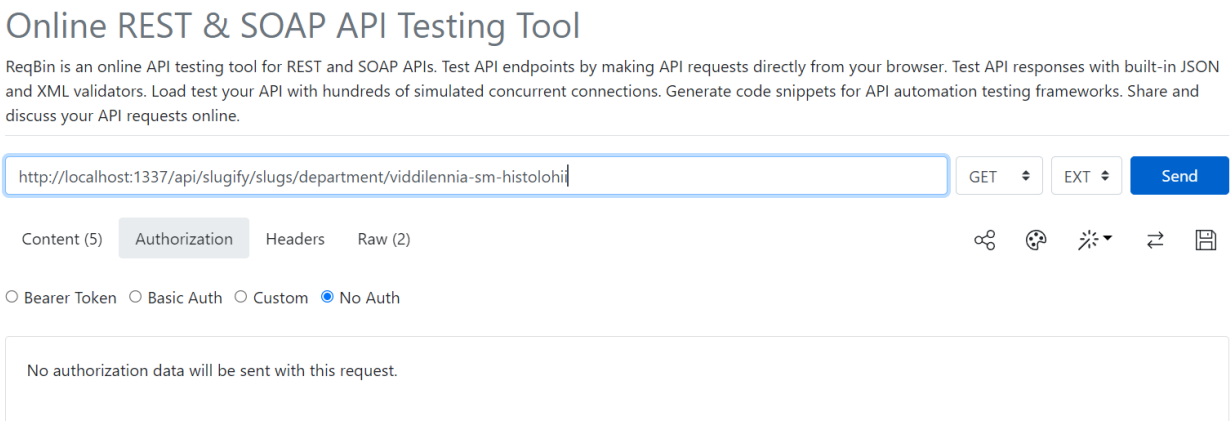


Рисунок 3.44 – Інструмент тестування запитів «ReqBin»

Зробивши GET-запити на ці кінцеві точки, неавторизований користувач має отримати відповідь «200», але «:id» потрібно замінити цілочисельним значенням, для прикладу взято – 1. Бо в кожній таблиці є запис з ідентифікатором, який відповідає одиниці. Для плагіну Slugify потрібно зробити тестовий запит, де замість :modelName підставлено “department”, а замість :slug – “ viddilennia-sm-histolohii”. Результати відповідей серверу можна побачити в терміналі системи контролю версій Strapi, зображено на рисунку 3.45. Наші тестові запити знаходяться на сьомій стрічці зверху, починаючи з часу 09:31:28.270. На всі запити сервер надав відповідь 200.

```
09:11:28.691] http: GET /admin/project-settings (17 ms) 200
09:11:28.709] http: GET /uploads/photo_5438468047997360077_x_918b7d884c.jpg (3 ms) 200
09:11:29.709] http: GET /users-permissions/roles (21 ms) 200
09:11:30.623] http: GET /users-permissions/permissions (24 ms) 200
09:11:30.645] http: GET /users-permissions/roles/2 (16 ms) 200
09:11:30.663] http: GET /users-permissions/routes (14 ms) 200
09:31:28.270] http: GET /api/slugify/slugs/department/viddilennia-sm-histolohii (16 ms) 200
09:31:32.033] http: GET /api/bureau-chief (6 ms) 200
09:31:42.816] http: GET /api/departments (8 ms) 200
09:31:49.821] http: GET /api/departments/1 (5 ms) 200
09:31:58.543] http: GET /api/fininfos (19 ms) 200
09:32:02.366] http: GET /api/fininfos/1 (5 ms) 200
09:32:10.626] http: GET /api/head-of-departments (13 ms) 200
09:32:13.967] http: GET /api/head-of-departments/1 (4 ms) 200
09:32:26.837] http: GET /api/idepartments (10 ms) 200
09:32:31.616] http: GET /api/idepartments/1 (6 ms) 200
09:32:36.623] http: GET /api/idepartments/1 (4 ms) 200
09:32:46.286] http: GET /api/newss (18 ms) 200
09:32:59.115] http: GET /api/newss/1 (5 ms) 200
```

Рисунок 3.45 – Результати тестування необхідних GET-запитів для неавторизованого користувача.

Далі потрібно протестувати, чи в неавторизованого користувача є доступ до типів колекцій, до яких в нього не повинно бути доступів. Потрібно виконати запити цих типів:

GET:

- /api/cabinet-departments,
- /api/cabinet-departments/:id,
- /api/cabinet-experts,
- /api/cabinet-experts/:id,
- /api/cabinet-head-of-departments,
- /api/cabinet-head-of-departments/:id,
- /api/referrals,
- /api/referrals/:id,

POST:

- /api/cabinet-departments,
- /api/cabinet-experts,
- /api/cabinet-head-of-departments,
- /api/departments,
- /api/fininfos,
- /api/head-of-departments,
- /api/idepartments,
- /api/newss,
- /api/referrals.

PUT та DELETE:

- /api/bureau-chief,
- /api/cabinet-departments/:id,
- /api/cabinet-experts/:id,
- /api/cabinet-head-of-departments/:id,
- /api/departments/:id,
- /api/fininfos/:id,
- /api/head-of-departments/:id,

- /api/idepartments/:id,
- /api/newss/:id,
- /api/referrals/:id.

DELETE: /api/upload/files/:id.

Для POST та PUT запитів в якості тесту додається body: {“data”: {“id”:88}, замість “:id” також буде підставлено цифру 1. Зробимо GET-запити по списку, очікуємий результат – 403 (Forbidden), що означає «Відмовлено в доступі». Результат GET-запитів неавторизованого користувача на заборонені адреси зображений на рисунку 3.46 починаючи з часу 09:55:15.234. Тестування POST-запитів на кінцеві точки з списку, дали очікуваний результат – відмовлено в доступі.

Результат POST-запитів неавторизованого користувача на заборонені кінцеві точки зображений на рисунку 3.47 починаючи з часу 10:02:01.141. Результат PUT-запитів неавторизованого користувача на заборонені кінцеві точки зображений на рисунку 3.48 починаючи з часу 10:14:33.073.

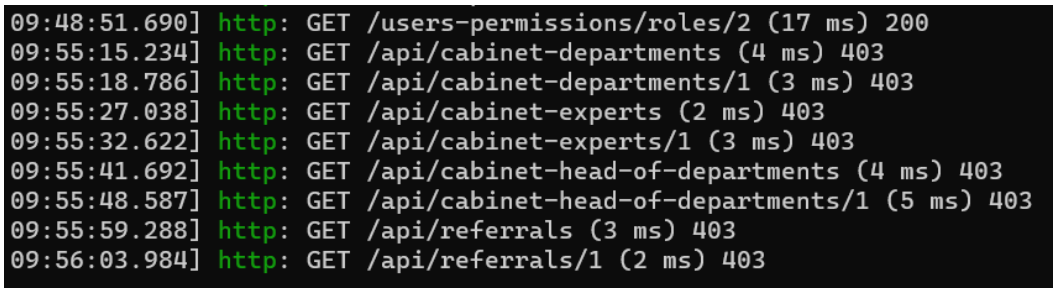


Рисунок 3.46 – Відповіді серверу на GET-запити неавторизованого користувача на заборонені кінцеві точки.

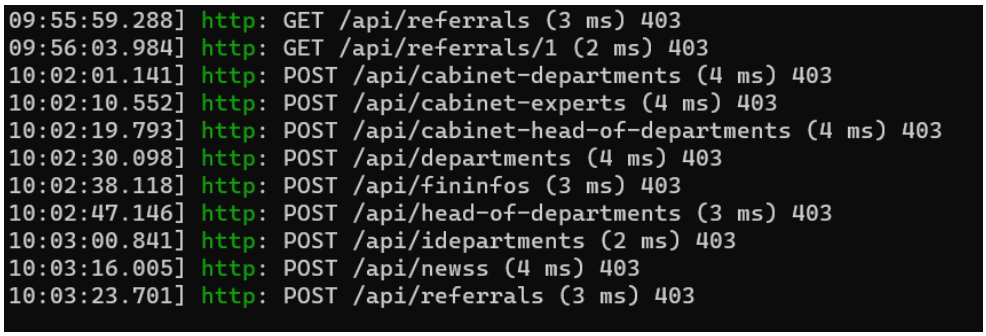


Рисунок 3.47 – Відповіді серверу на POST-запити неавторизованого користувача

на заборонені кінцеві точки.

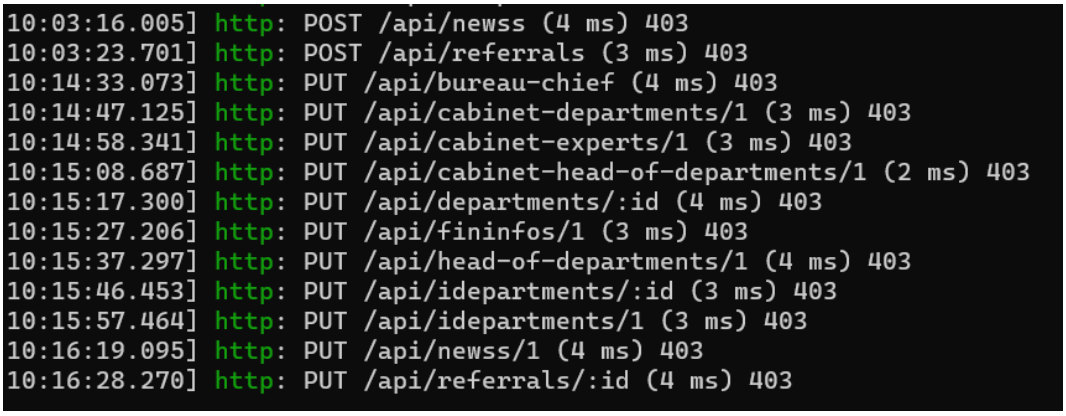


Рисунок 3.48 – Відповіді серверу на PUT-запити неавторизованого користувача на заборонені кінцеві точки.

Залишилось зробити останній тип запиту, це DELETE-запит. DELETE-запит не потребує body, натомість слід передати ідентифікатор запису, або просто застосувати його до колекції, якщо її тип – Single Type. Результат тестування DELETE-запитами наведений на рисунку 3.49, починаючи з часу 10:21:08.613.

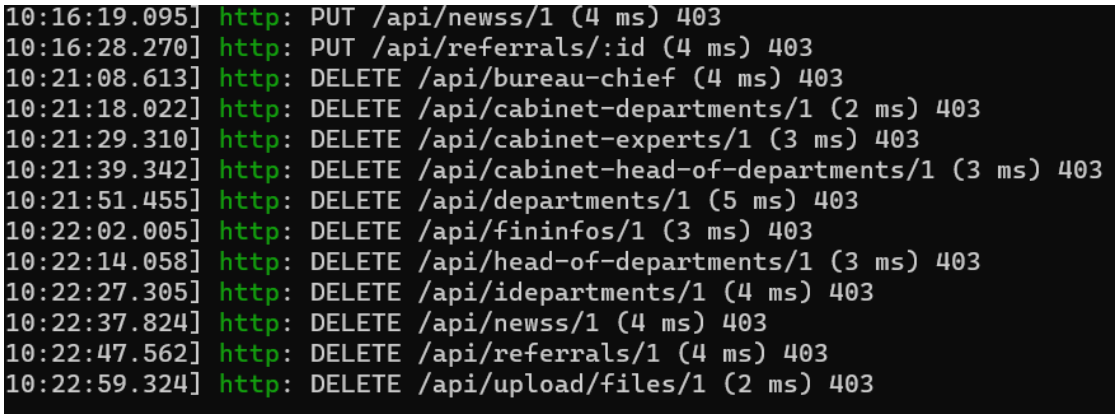


Рисунок 3.49 – Відповіді серверу на DELETE-запити неавторизованого користувача на заборонені кінцеві точки.

4. ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка автоматизованої системи бюро судово-медичної експертизи» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1.

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	3	4	3
2. Ринкові переваги (наявність аналогів)	1	2	1
3. Ринкові переваги (ціна продукту)	3	2	2
4. Ринкові переваги (технічні властивості)	3	2	2
5. Ринкові переваги (експлуатаційні витрати)	2	2	2
6. Ринкові перспективи (розмір ринку)	1	2	2
7. Ринкові перспективи (конкуренція)	2	2	2
8. Практична здійсненність (наявність фахівців)	3	2	2
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	3	2	3
11. Практична здійсненність (термін реалізації)	3	4	4
12. Практична здійсненність (розробка документів)	4	4	4
Сума балів	31	31	30
Середньоарифметична сума балів $СБ_c$	**(31+31+30)/3 = 30,6**		

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [66].

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка автоматизованої системи бюро судово-медичної експертизи» становить 30,6 бала, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

4.2 Розрахунок витрат на здійснення розробки

4.2.1 Основна заробітна плата дослідників, яка розраховується за формулою:

$$З_o = \sum_{i=1}^{k} \frac{M_{ni} * t_i}{T_p} \quad (4.1)$$

де k – кількість посад дослідників, залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – кількість днів роботи конкретного дослідника, дн.;

T_p – середня кількість робочих днів в місяці, T_p=*21…23* дні. Проведені розрахунки занесено до таблиці – 4.4.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Кількість днів роботи	Витрати на заробітну плату, грн
Керівник проекту	18550	742	25	20163,04
Інженер-програміст	17330	693,2	25	18836,95
Інженер-електронік	8500	472,2	18	6652,17
Всього				45652,12

4.2.2 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_р = \sum_{i=1}^{n} C_i * t_i \quad (4.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M * K_i * K_c}{T_p * t_{зм}} \quad (4.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи або мінімальної

місячної заробітної плати (залежно від діючого законодавства), – 6700 грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б);

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мініма- ної заробітної плати. (табл. Б.1, додаток Б)

$Т_р$ – середня кількість робочих днів в місяці, приблизно $Т_р = 21…23$ *дні*;

$t_{зм}$ – тривалість зміни, 8 год.

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
CMS	43	5	1,7	111,42	4791,06
Інтерфейси користувача	181	5	1,7	111,42	20167,02
Всього					24958,08

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

4.2.3 Додаткова заробітна плата дослідників та робітників

Додаткова заробітна плата розраховується як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (4.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати.

$$З_{дод} = (45652{,}12 + 24958{,}08) * \frac{11\%}{100\%} = 7767{,}12(грн.)$$

4.2.4 Відрахування та нарахування на заробітну плату дослідників та робітників

До статті «Відрахування на соціальні заходи» належать відрахування внеску на загальнообов'язкове державне соціальне страхування та для здійснення заходів щодо соціального захисту населення (ЄСВ – єдиний соціальний внесок).

Нарахування на заробітну плату дослідників та робітників розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) * \frac{Н_{зп}}{100\%} \quad (4.5)$$

$$З_н = (45652{,}12 + 24958{,}08 + 7767{,}12) * \frac{22\%}{100\%} = 17243{,}01 \text{ (грн.)}$$

де $H_{зп}$ – норма нарахування на заробітну плату.

4.2.5 Розрахунок витрат на матеріали та комплектуючі

Оскільки для розроблювального програмного засобу не потрібно витрачати матеріали та комплектуючі, то витрати на матеріали і комплектуючі дорівнюють нулю.

4.2.6 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування, верстатів, пристроїв, інструментів, приладів, стендів, апаратів, механізмів, іншого спецобладнання, необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними цінами. До балансової вартості устаткування окрім прейскурантної вартості входять витрати

на його транспортування і монтаж, тому ці витрати беруться додатково в розмірі 10…12% від вартості устаткування.

Балансову вартість спецустаткування розраховують за формулою:

$$В_{спец} = \sum_{i=1}^{k} Ц_i * С_{пр.i} * K_i \quad (4.6)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$С_{пр.i}$ – кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, (Кі = 1,10…1,12);

k – кількість найменувань устаткування. Отримані результати занесено до таблиці – 4.6.

Таблиця 4.6 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
БФП Canon PIXMA Ink Efficiency E414	1	3499	3918,88
Всього			3918,88

4.2.7 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення.

Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{i=1}^{k} Ц_{iпрг} * С_{прг.i} * К_i \quad (4.7)$$

де $Ц_{іпрг}$ – ціна придбання одиниці програмного засобу цього виду, грн;

$С_{прг.і}$. – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (Кі = 1,10…1,12);

k – кількість найменувань програмних засобів.

Отримані результати занесено до таблиці – 4.7.

Таблиця 4.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Windows 11 Pro	1	7899	8688,9
ESET Home Security Premium	1	1668	1834,8
Всього			10523,7

4.2.8 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо можуть бути розраховані з використанням прямолінійного методу амортизації за формулою:

$$А_{обл} = \frac{Ц_б}{Т_в} * \frac{t_{вик}}{12} \quad (4.8)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$Т_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Розрахуємо, для прикладу, амортизаційні витрати на ноутбук балансова вартість якого становить 37599 грн., термін його корисного використання – 4 роки,

а термін його фактичного використання – 1 міс.

$$А_{обл} = \frac{37599}{4} * \frac{1}{12} = 783,28 \text{ (грн)}$$

Аналогічно визначаємо амортизаційні витрати на додаткове програмне забезпечення, такі як: ліцензійний ключ Windows 11, ліцензійний ключ антивірусу ESET Home Security Premium, також розраховуємо обладнання: принтер, робота виконується в приміщенні, що надане державою, оскільки підприємство - державне. Розрахунки заносимо до таблиці 4.8.

Таблиця 4.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук Acer Aspire 7	37599	4	1	783,28
Додаткове програмне забезпечення	10523,7	1	1	876,93
Офісне обладнання	3918,88	5	1	65,31
Всього				1725,52

4.2.9 Витрати на електроенергію

До статті «Паливо та енергія для науково-виробничих цілей» належать витрати на придбання у сторонніх підприємств, установ і організацій будь-якого палива, що витрачається з технологічною метою на проведення досліджень.

Витрати на силову електроенергію (Ве) розраховують за формулою:

$$В_е = \sum_{i=1}^{n} \frac{W_{yi} * t_i * Ц_е * K_{впі}}{\eta_i} \quad (4.9)$$

де W_{yi} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії);

$K_{\text{впі}}$ – коефіцієнт, що враховує використання потужності, $K_{\text{впі}} < 1$;

$\eta_{\text{і}}$ – коефіцієнт корисної дії обладнання, ηі < 1;

Проведені розрахунки занесено до таблиці.

Таблиця 4.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук Acer Aspire 7	0,13	194	74,9
БФП Canon PIXMA Ink Efficiency E414	0,3	15	11,88
Всього			86,78

4.2.10 Службові відрядження

До статті «Службові відрядження» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{В}_{\text{св}} = \left(\text{З}_{\text{о}} + \text{З}_{\text{р}}\right) * \frac{\text{Н}_{\text{св}}}{100\%} \qquad (4.10)$$

де $\text{Н}_{\text{св}}$ – норма нарахування за статтею «Службові відрядження».

$$\text{В}_{\text{св}} = (45652{,}12 + 24958{,}08) * \frac{25\%}{100\%} = 17652{,}55 \text{ (грн)}$$

4.2.11 Витрати на роботи, які виконують сторонні підприємства, установи і організації

До статті «Витрати на роботи, які виконують сторонні підприємства,

установи і організації» належать витрати на проведення досліджень, що не можуть бути виконані штатними працівниками або наявним обладнанням організації, а виконуються на договірній основі іншими підприємствами, установами і організаціями незалежно від форм власності та позаштатними працівниками.

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуються як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{сп} = (З_о + З_р) * \frac{Н_{нсп}}{100\%} \quad (4.11)$$

де $Н_{нсп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації».

$$В_{сп} = (45652{,}12 + 24958{,}08) * \frac{40\%}{100\%} = 28244{,}08 \text{ (грн)}$$

4.2.12 Інші виробничі витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) * \frac{Н_{ів}}{100\%} \quad (4.12)$$

де $Н_{ів}$– норма нарахування за статтею «Інші витрати».

$$І_в = (45652{,}12 + 24958{,}08) * \frac{50\%}{100\%} = 35305{,}1 \text{(грн)}$$

4.2.13 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) * \frac{Н_{нзв}}{100\%} \quad (4.13)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$В_{нзв} = (45652{,}12 + 24958{,}08) * \frac{100\%}{100\%} = 70610{,}2 \text{ (грн)}$$

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_є + В_{сп} + І_в + В_{нзв} \quad (4.14)$$

$$В_{заг} = 45652{,}12 + 24958{,}08 + 7767{,}12 + 17243{,}01 + 6700 + 0 + 3918{,}88 + 10523{,}7 + 1725{,}52 + 86{,}78 + 28244{,}08 + 35305{,}1 + 70610{,}2 = 252734{,}59 \text{ (грн)}$$

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \quad (4.15)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9.

Оберемо η = 0,7, так як розробка, на даний момент, знаходиться на стадії розробки промислового зразка.

$$ЗВ = \frac{252734,59}{0,7} = 361049,41 \text{ (грн)}$$

4.3 Розрахунок економічної ефективності науково-технічної розробки від її впровадження безпосередньо розробником (замовником)

При виконанні даної роботи за темою «Розробка автоматизованої системи бюро судово-медичної експертизи» розглядається ситуація, коли замовник певної науково-технічної розробки використовує її тільки на своєму підприємстві (чи в організації) і не виводить її на ринок. У цьому випадку позитивним результатом від впровадження цієї науково-технічної розробки може бути покращення певних економічних та фінансових показників діяльності підприємства.

Для визначення величини майбутнього економічного ефекту та ефективності розробки визначимо певні характеристики підприємства.

Таблиця 4.10 – Вихідні дані замовника

Показник	Рік розробки	1-й рік	2-й рік	3-й рік	4-й рік
Чисельність працівників, які виконують визначені функції вручну, осіб	2	-	-	-	-

Середня заробітна плата працівника, який виконує відповідну функцію вручну, грн	18150,00	-	-	-	-
Приблизні витрати на розробку автоматизованої системи управління, грн	361049,41	-	-	-	-
Економія чисельності працівників виконання виробничої чи управлінської функції яких було автоматизовано у році що аналізується, осіб	-	1	1	0	0
Кількість функцій, які виконуються вручну у році до впровадження результатів нової науково-технічної розробки, шт	5	-	-	-	-
Прогнозоване зростання кількості виробничих чи інформаційно-технічних управлінських функцій, виконання яких автоматизується, у році що аналізується (відносно року до впровадження даної розробки), шт	-	3	2	0	0

В даному випадку майбутній економічний ефект та ефективність буде формуватися на основі використання таких показників: $\Delta П_я$ – зростання прибутку підприємства внаслідок зниження витрат на оплату праці працівників, які виконують окремі виробничі чи інформаційно-технічні управлінські функції, грн. Причому $\Delta П_я$ може бути визначено як:

$$\Delta П_я = \frac{ЧП \cdot ЗП \cdot 12}{N} - \frac{(0,2...0,6) \cdot ЗВ}{\Delta N_i}, \qquad (4.21)$$

де $ЧП$ – чисельність працівників, які виконують визначені функції вручну, приймемо 2 особи; $ЗП$ – середня заробітна плата працівника, який виконує відповідну функцію вручну, приймемо 10050,00 грн; $ЗВ$ – приблизні витрати на розробку автоматизованої системи управління, приймемо 361049,41 грн; N – кількість функцій, які виконуються вручну у році до впровадження результатів нової науково-технічної розробки, 5 шт; ΔN_i – прогнозоване зростання кількості виробничих чи інформаційно-технічних управлінських функцій, виконання яких автоматизується, у році що аналізується (відносно року до впровадження даної розробки), шт.

Зростання прибутку підприємства в 1-й рік впровадження розробки

$\Delta П_я$ =2·18150,00·12/5 - 0,4·361049,41/3 = 38980,08 грн/функц.

Зростання прибутку підприємства в 2-й рік впровадження розробки

$\Delta П_я$ =2·18150,00·12/5 - 0,4·361049,41/5 = 58236,05 грн/функц.

Зростання прибутку підприємства в 3-й рік впровадження розробки

$\Delta П_я$ =2·18150,00·12/5 - 0,4·361049,41/5 = 58236,05 грн/функц.

Зростання прибутку підприємства в 4-й рік впровадження розробки

$\Delta П_я$ =2·18150,00·12/5 - 0,4·361049,41/5 = 58236,05 грн/функц.

$П_я$ – прибуток, який отримує підприємство від автоматизації виконання окремої виробничої чи інформаційно-технічної управлінської функції у кожному із років після впровадження науково-технічної розробки, грн. Даний прибуток можна приблизно оцінити виходячи з формули:

$$П_я = \frac{\Delta ЧП \cdot ЗП \cdot 12}{N}, \qquad (4.22)$$

де $\Delta ЧП$ – економія чисельності працівників виконання виробничої чи управлінської функції яких було автоматизовано у році що аналізується, осіб;

Прибуток який отримує підприємство від автоматизації функції в 1-й рік

$П_я$ =1·18150,00·12/5 = 43560 грн/функц.

Прибуток який отримує підприємство від автоматизації функції в 2-й рік

$П_я$ =1·18150,00·12/5 = 43560 грн/функц.

Прибуток який отримує підприємство від автоматизації функції в 3-й рік

$П_я$ =0·18150,00·12/5 = 0 грн/функц.

Прибуток який отримує підприємство від автоматизації функції в 4-й рік

$П_я$ =0·18150,00·12/5 = 0 грн/функц.

Збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження науково-технічної розробки, розраховуємо за формулою:

$$\Delta П_i = (\Delta П_я \cdot N + П_я \cdot \Delta N)_i, \quad (4.23)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження на підприємстві результатів науково-технічної розробки у році що аналізується;

N – основний кількісний показник, який визначає обсяг діяльності підприємства у році до впровадження результатів нової науково-технічної розробки;

$П_я$ – основний якісний показник, який визначає результати діяльності підприємства у кожному із років після впровадження науково-технічної розробки;

ΔN – зміна основного кількісного показника діяльності підприємства в результаті впровадження науково-технічної розробки у році що аналізується.

Збільшення чистого прибутку підприємства в 1-й рік впровадження $\Delta П_i$ =38980,08·5+43560·3= 325580,4 грн.

Збільшення чистого прибутку підприємства в 2-й рік впровадження $\Delta П_i$ =58236,05 5+43560·(3+2)= 412700,4 грн.

Збільшення чистого прибутку підприємства в 3-й рік впровадження $\Delta П_i$ =58236,05·5+0·(3+2+0)= 194900,4 грн.

Збільшення чистого прибутку підприємства в 4-й рік впровадження $\Delta П_i$ =58236,05·5+0·(3+2+0+0)= 194900,4 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати замовник від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.24)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів

від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,13;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання замовником додаткових чистих прибутків у цьому році.

$ПП$ =325580,4/(1+0,13)^1+412700,4/(1+0,13)^2+ 194900,4/(1+0,13)^3+

+194900,4/(1+0,13)^4=367905,85+526977,14+281221,20+317779,95= 1493884,14 грн.

Величина початкових інвестицій PV, які замовник має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{4.25}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 361049,41 грн.

$PV = k_{інв} \cdot ЗВ$ = 2 361049,41= 722098,82 грн.

Абсолютний економічний ефект $E_{абс}$ для розробника від можливого впровадження науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.26}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 1493884,14 грн;

PV – теперішня вартість початкових інвестицій, 722098,82грн.

$E_{абс} = ПП - PV$ = 1493884,14 - 722098,82 = 771785,32 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені розробником у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.27)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 487905,97 грн;

PV – теперішня вартість початкових інвестицій, 198423,3 грн;

$T_{ж}$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+771785,32/722098,82)^{1/4} = 0,517.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (4.28)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,22.

$\tau_{мін}$ = 0,1+0,22 = 0,32 < 0,517 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені розробником у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка автоматизованої системи бюро судово-медичної експертизи» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені розробником у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \quad (4.29)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,517= 1,93р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати розробника профінансувати впровадження даної розробки для застосування в діяльності підприємства.

Висновки до розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка автоматизованої системи бюро судово-медичної експертизи» становить 30.6 бала, що свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

Також термін окупності становить 1,93р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати розробника до впровадження даної розробки при отриманні ефекту в розмірі 771785,32 грн.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка автоматизованої системи судово-медичної експертизи».

ВИСНОВКИ

Отже, під час розробки автоматизованої системи бюро судово-медичної експертизи було оглянуто сучасний технологічний та методичний стан розробки автоматизованих систем для підприємств та установ. Проаналізовано аналогічні системи, та системи з схожим функціоналом. Проаналізовані способи та методи вирішення поставлених задач, підібрані технології та методики розробки системи. Розроблена логіка серверної частини сайту, за яку відповідає система контролю вмістом Strapi, створено типи колекцій, з мінімально достатньою кількістю полів та зв’язків між типами колекцій. Налаштовані ролі користувачів, та права доступу до типів колекцій в залежності від ролі користувача, що користується системою. Створені діаграми таблиць бази даних для ролей «Експерт» та «Завідувач відділом/відділенням». Використано плагіни: «Slugify» для застосування пошуку по таблицям за полем «slug»; «Users & Permissions» для впровадження до системи ролей та прав доступу; «Uploads» для зберігання та обміну документами та медіаданими.

Розроблений front-end додаток, з використанням фреймворку Next.js, з системою маршрутизації App Router. Умовно front-end було розділено на дві частини: частина для неавторизованих користувачів – частина системи, яка є веб-сайтом, на котрому користувач зможе знайти публічну інформацію про діяльність організації, контактні дані установи, адреси, список відділень та напрямки їх роботи, інформацію про завідувачів відділення, публічну інформацію про фінансову діяльність установи згідно Господарського кодексу України, та інші сторінки описані завданням; Друга умовна частина – особистий кабінет працівника установи, де судово-медичні експерти надсилають електронні направлення між відділеннями для дослідження об’єктів чи проведення експертиз іншого роду.
Система протестована списком запитів при різних умовах для різних ролей, результати тестування наведені в підрозділі 3.3.

Було автоматизовано процес обміну направленнями, та надання відповідей по ним.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Shopify. Wikipedia. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Shopify
2. Prozorro. Wikipedia. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Prozorro
3. НАУКОВО-ПРАКТИЧНИЙ КОМЕНТАР до статті 78 Господарського кодексу України. Liga 360. [Електронний ресурс] – Режим доступу: https://ips.ligazakon.net/document/KK003844
4. Наказ №6 від 17.01.95. Про розвиток та вдосконалення судово-медичної служби України. Міністерство охорони здоров'я. Електронний ресурс] – Режим доступу: https://zakononline.com.ua/documents/show/160027___160027
5. Welcome to DocuWare. About DocuWare. [Електронний ресурс] – Режим доступу: https://start.docuware.com/about
6. Початок роботи зі службою SharePoint. Документація Microsoft. [Електронний ресурс] – Режим доступу: https://support.microsoft.com/uk-ua/office/%D0%BF%D0%BE%D1%87%D0%B0%D1%82%D0%BE%D0%BA-%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B8-%D0%B7%D1%96-%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%BE%D1%8E-sharepoint-909ec2f0-05c8-4e92-8ad3-3f8b0b6cf261
7. Elektronischer Leitz-Ordner. Wikipedia. [Електронний ресурс] – Режим доступу: https://de.wikipedia.org/wiki/Elektronischer_Leitz-Ordner
8. Documentum. Wikipedia. [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Documentum
9. FileNet. Wikipedia. [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/FileNet
10. Hyland Software. Wikipedia. [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Hyland_Software
11. M-Files. Wikipedia. [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/M-Files

12. WordPress. Wikipedia. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/WordPress
13. Joomla! Wikipedia. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Joomla!
14. Drupal. Wikipedia. [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Drupal
15. React: Making faster, smoother UIs for data-driven Web apps. InfoWorld. [Електронний ресурс] – Режим доступу: https://web.archive.org/web/20151215102758/http://www.infoworld.com/article/2608181/javascript/react--making-faster--smoother-uis-for-data-driven-web-apps.html
16. Facebook's React JavaScript User Interfaces Library Receives Mixed Reviews. InfoQ. [Електронний ресурс] – Режим доступу: https://web.archive.org/web/20160308144447/http://www.infoq.com/news/2013/06/facebook-react
17. JavaScript’s History and How it Led To ReactJS. The New Stack. [Електронний ресурс] – Режим доступу: https://web.archive.org/web/20160316235033/http://thenewstack.io/javascripts-history-and-how-it-led-to-reactjs/
18. "Next.js, Strapi, and Tailwind CSS Blog Tutorial" (Medium стаття). [Електронний ресурс] – Режим доступу: https://medium.com/fullstacked/next-js-strapi-and-tailwind-css-blog-tutorial-4053828f4c36
19. "What Is AngularJS? Meaning, Working, Benefits and Challenges". Spiceworks. [Електронний ресурс] – Режим доступу: https://www.spiceworks.com/tech/devops/articles/what-is-angular-js/
20. VueJS. Simplified JavaScript Jargon. [Електронний ресурс] – Режим доступу: https://web.archive.org/web/20170211081024/http://jargon.js.org/_glossary/VUEJS.md
21. First Week of Launching Vue.js. Evan You. [Електронний ресурс] – Режим доступу:

https://web.archive.org/web/20170412110312/http://blog.evanyou.me/2014/02/11/first-week-of-launching-an-oss-project/index.html

22. Template Syntax — Vue.js. [Електронний ресурс] – Режим доступу: https://web.archive.org/web/20211104004717/https://vuejs.org/v2/guide/syntax.html
23. Vue 2.0 is Here!. The Vue Point. [Електронний ресурс] – Режим доступу: https://web.archive.org/web/20170312072632/https://medium.com/the-vue-point/vue-2-0-is-here-ef1f26acf4b8
24. Recommended Toolchains (HTML). React documentation. [Електронний ресурс] – Режим доступу: https://reactjs.org/docs/create-a-new-react-app.html#recommended-toolchains
25. Next.js Brand Guidelines. [Електронний ресурс] – Режим доступу: https://vercel.com/design/brands#next-js
26. Develop. Preview. Ship. For the best frontend teams – Vercel (HTML). vercel.com [Електронний ресурс] – Режим доступу: https://vercel.com/home
27. Differences Between Static Generated Sites And Server-Side Rendered Apps. Smashing Magazine. [Електронний ресурс] – Режим доступу: https://www.smashingmagazine.com/2020/07/differences-static-generated-sites-server-side-rendered-apps/
28. Thakkar, Mohit (2020). Adding Server-Side Rendering to Your React Application. У Thakkar, Mohit. Building React Apps with Server-Side Rendering: Use React, Redux, and Next to Build Full Server-Side Rendering Applications. [Електронний ресурс] – Режим доступу: https://doi.org/10.1007/978-1-4842-5869-9_4
29. Vue vs React у 2022 році - який фреймворк вибрати і коли? [Електронний ресурс] – Режим доступу: https://wezom.com.ua/ua/blog/vue-vs-react
30. Netlify CMS. Helpip.ua. [Електронний ресурс] – Режим доступу: https://hellip.com/ua/product/netlifycms.html
31. Netlify CMS Documentation. [Електронний ресурс] – Режим доступу: https://decapcms.org/docs/intro/
32. Contentful. Managing Content. [Електронний ресурс] – Режим доступу:

https://www.contentful.com/help/content-tab/

33. Про Strapi: Це те, що слухають мій робот і мої друзі. [Електронний ресурс] – Режим доступу: https://strapi.io/about-us
34. Функціональна нефункціональна CMS і вбудований API | Страпі. [Електронний ресурс] – Режим доступу: https://strapi.io/features
35. Бази даних | Документація розробника Strapi. [Електронний ресурс] – Режим доступу: https://strapi.io/documentation/developer-docs/latest/guides/databases.html
36. Моделі | Документація розробника Strapi. [Електронний ресурс] – Режим доступу: https://strapi.io/documentation/developer-docs/latest/concepts/models.html#concept
37. Документація API | Документація розробника Strapi. [Електронний ресурс] – Режим доступу: https://strapi.io/documentation/developer-docs/latest/plugins/documentation.html#api-documentation
38. Strapi - аналіз і порівняння | Bluepick. [Електронний ресурс] – Режим доступу: https://www.bluepick.de/cms/strapi
39. Проміжне програмне забезпечення | Документація розробника Strapi. [Електронний ресурс] – Режим доступу: https://strapi.io/documentation/developer-docs/latest/concepts/middlewares.html#structure
40. Стрічки | Чому Koa? [Електронний ресурс] – Режим доступу: https://strapi.io/blog/why-koa
41. Introduction to users, roles & permissions. Strapi. [Електронний ресурс] – Режим доступу: https://docs.strapi.io/user-docs/users-roles-permissions
42. What Is SEO – Search Engine Optimization? Search Engine Land. [Електронний ресурс] – Режим доступу: https://searchengineland.com/guide/what-is-seo
43. What is Babel? Babel.js. [Електронний ресурс] – Режим доступу: https://babeljs.io/docs/
44. Is React SEO-Friendly? Key React Search Engine Optimization Tips. RubyGarage. [Електронний ресурс] – Режим доступу:

https://rubygarage.org/blog/seo-for-react-websites

45. Headless CMS explained in 1 minute. Contentful. [Електронний ресурс] – Режим доступу: https://www.contentful.com/headless-cms/
46. Understanding and Using Relations in Strapi. Strapi. [Електронний ресурс] – Режим доступу: https://strapi.io/blog/understanding-and-using-relations-in-strapi
47. Quick Start Guide. Strapi. [Електронний ресурс] – Режим доступу: https://docs.strapi.io/dev-docs/quick-start
48. Introduction to the Content Manager. Strapi. [Електронний ресурс] – Режим доступу: https://docs.strapi.io/user-docs/content-manager
49. Introduction to the Content-type Builder. Strapi. [Електронний ресурс] – Режим доступу: https://docs.strapi.io/user-docs/content-type-builder
50. Media Library Feature. Strapi. [Електронний ресурс] – Режим доступу: https://strapi.io/features/media-library
51. Models. Strapi. [Електронний ресурс] – Режим доступу: https://docs.strapi.io/dev-docs/backend-customization/models
52. What Is a Database Diagram? Vertabelo.com. [Електронний ресурс] – Режим доступу: https://vertabelo.com/blog/what-is-database-diagram/
53. Upload plugin. Strapi. [Електронний ресурс] – Режим доступу: https://docs.strapi.io/dev-docs/plugins/upload
54. API Reference. create-next-app. Nextjs.org. [Електронний ресурс] – Режим доступу: https://nextjs.org/docs/pages/api-reference/create-next-app
55. ESLint: What, Why, When, How. DEV Community. [Електронний ресурс] – Режим доступу: https://dev.to/shivambmgupta/eslint-what-why-when-how-5f1d
56. "Tailwind CSS - From Zero to Production" by Academind - Відеокурс на YouTube. [Електронний ресурс] – Режим доступу: https://www.youtube.com/watch?v=Mo4lEbbuW3I
57. Що таке npm і навіщо він потрібен. R_d media. [Електронний ресурс] – Режим доступу: https://robotdreams.cc/uk/blog/271-chto-takoe-npm-i-zachem-on-nuzhen
58. Pages and Layouts. Next.js. [Електронний ресурс] – Режим доступу:

https://nextjs.org/docs/pages/building-your-application/routing/pages-and-layouts

59. What is DRY Development? Digital Ocean. [Електронний ресурс] – Режим доступу: https://www.digitalocean.com/community/tutorials/what-is-dry-development
60. Routing Fundamentals. Next.js. [Електронний ресурс] – Режим доступу: https://nextjs.org/docs/app/building-your-application/routing
61. When to use dangerouslySetInnerHTML in React? Refine. [Електронний ресурс] – Режим доступу: https://refine.dev/blog/use-react-dangerouslysetinnerhtml/
62. Introduction to JSON Web Tokens. Jwt.io. [Електронний ресурс] – Режим доступу: https://jwt.io/introduction
63. Data Fetching: Server Actions and Mutations. Next.js. [Електронний ресурс] – Режим доступу: https://nextjs.org/docs/app/building-your-application/data-fetching/server-actions-and-mutations
64. Тестування баз данних: На що звернути увагу? QaTestLab. [Електронний ресурс] – Режим доступу: https://training.qatestlab.com/blog/technical-articles/database-testing-what-to-look-for/
65. About ReqBin. ReqBin. [Електронний ресурс] – Режим доступу: https://reqbin.com/about
66. МЕТОДИЧНІ ВКАЗІВКИ до виконання економічної частини магістерських кваліфікаційних робіт. В.О. Козловський, О.Й. Лесько В.В. Кавецький. 2021р.

ДОДАТКИ

Додаток А

(обов'язковий)

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: **« Розробка автоматизованої системи бюро судово-медичної експертизи»**

Тип роботи: Магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ КСУ, ФІІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 96,4% Схожість 3,6%

Аналіз звіту подібності (відмітити потрібне)

ˇ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Володимир ДУБОВОЙ
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek щодо роботи.

Автор роботи _____ Олександр ПОБЕРЕЖНЯК
(підпис) (прізвище, ініціали)

Керівник роботи _____ Олег КОВАЛЮК
(підпис) (прізвище, ініціали)

Додаток Б

(обов'язковий)

ВНТУ

ЗАТВЕРДЖЕНО
Т.в.о.Зав. кафедри КСУ ВНТУ,
д.т.н., доцент
_____Марія ЮХИМЧУК
" 6 "жовтня 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

Розробка автоматизованої системи бюро судово-медичної експертизи

(тема)

08-33.МКР.10.00.000 ТЗ
номер

Студент групи Олександр ПОБЕРЕЖНЯК

Підпис *Ім'я ПРІЗВИЩЕ*

Керівник _____ Олег КОВАЛЮК

Підпис *Ім'я ПРІЗВИЩЕ*

Вінниця 2023

1. Назва та галузь застосування

1.1. Назва – Автоматизована система бюро судово-медичної експертизи.

1.2. Галузь застосування – документообіг, звітність, інтернет-ресурс.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від “18” вересня 2023 року №247

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є автоматизація подання потрібної звітності організації для держави та населення відповідно Господарського кодексу України, впровадження інтернет-порталу, автоматизація процесу обміну направленнями та відповідями на них між судово-медичними експертами.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1."Next.js in Action" by Adam Boduch. 2022.

2."Practical Next.js" by Frank Taillandier. 2021.

3. "Next.js, Strapi, and Tailwind CSS Blog Tutorial" (Medium стаття). [Електронний ресурс]. Режим доступу до ресурсу: https://medium.com/fullstacked/next-js-strapi-and-tailwind-css-blog-tutorial-4053828f4c365.

Вимоги до розробки.

5.1. Перелік головних функцій:

- можливість авторизації в особистий кабінет експерта або завідуючого відділом;
- можливість відправки та отримання направлень між судово-

медичними експертами;

- можливість надавати відповідь на отримане направлення з прикріпленим до відповіді файлом-документом;
- можливість завідуючого відділом призначити підлеглому судово-медичному експерту направлення, що отримане в відділ;
- можливість перегляду сторінок з публічною інформацією для неавторизованих користувачів.

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- WINDOWS 10;
- Google Chrome, Mozilla Firefox, Opera, Microsoft Edge;
- Visual Studio Code.

5.2.2. Умови експлуатації системи:

- робота на мобільних та веб-додатках;
- можливість цілодобового функціонування системи;
- дані оновлюються і є актуальними.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Розробка технічного завдання «28»_09_ 2023 р.
2. Аналіз автоматизованих систем підприємств й установ та огляд аналогів. «15»_10_ 2023 р.
3. Проєктування автоматизованої системи бюро судово-медичної експертизи «25»_10 _ 2023 р.
4. Розробка та тестування автоматизованої системи «29»_10_ 2023 р.
5. Оформлення пояснювальної записки і графічного матеріалу «30»_11_ 2023 р.

6.2 Графічні матеріали:

1. Розробка UML-діаграм системи «26»_11_ 2023 р.
2. Розробка моделі бази даних системи «27»_11_ 2023 р.
3. Розробка демонстраційних плакатів «28»_11_ 2023 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «11»__12_ 2023 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «11»__12_ 2023 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засідані ЕК. Захист магістерської кваліфікаційної роботи провести до «19»__12_ 2023 р.

Додаток В

(довідниковий)

Програмний код функцій авторизації

```
import Cookies from "js-cookie";
import { authMeId, authMeUsername } from "./actions/authorisation";

export const setToken = (data) => {
    if(typeof window === 'undefined') {
        return;
    }
    Cookies.set('id', data.user.id);
    Cookies.set('username', data.user.username);
    Cookies.set('jwt', data.jwt)

    if(Cookies.get('username') && Cookies.get('id') && Cookies.get('jwt')) {
        return true;
    }
}

export const unsetToken = () => {
    if(typeof window === 'undefined') {
        return;
    }
    Cookies.remove('id')
    Cookies.remove('username')
    Cookies.remove('jwt')
    return true;
}

export const getUserFromLocalCookie = async () => {
```

```
    const jwt = getTokenFromLocalCookie();

    if (jwt) {
        const response = await authMeUsername(jwt)
        return await response
    } else {
        return;
    }
}

export const getIdFromLocalCookie = async () => {
    const jwt = getTokenFromLocalCookie();

    if (jwt) {
        return await authMeId(jwt)
    } else {
        return;
    }
}


export const getTokenFromLocalCookie = () => {
    return Cookies.get('jwt');
}

export const getTokenFromServerCookie = (req) => {
    if(!req.headers.cookie || '') {
        return undefined;
    }
    const jwtCookie = req.headers.cookie
```

```
        .split(';')
        .find((c) => c.trim().startsWith('jwt='));
    if (!jwtCookie) {
        return undefined;
    }
    const jwt = jwtCookie.split('=')[1];
    return jwt;
}

export const getIdFromServerCookie = (req) => {
    if(!req.headers.cookie || '') {
        return undefined;
    }
    const idCookie = req.headers.cookie
        .split(';')
        .find((c) => c.trim().startsWith('id='));
    if (!idCookie) {
        return undefined;
    }
    const id = idCookie.split('=')[1];
    return id;
}
```

Додаток Г

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

Розробка автоматизованої системи бюро судово-медичної експертизи
(тема)

1. Аналіз схожих автоматизованих систем, вибір технологій;
2. UML-діаграма варіантів використання;
3. Діаграма бази данних для типів колекцій неавторизованого користувача;
4. Діаграма бази данних для типів колекцій авторизованого користувача;
5. Вигляд навігаційного меню сторінок для неавторизованого користувача з відкритою вкладкою «ВОБ СМЕ»;
6. Вигляд головної сторінки для неавторизованого користувача;
7. Вигляд компоненту перегляду PDF-документів для неавторизованого користувача;
8. Вигляд кабінету судово-медичного експерта;
9. Вигляд кабінету завідуючого відділом/відділенням;
10. Вигляд модального вікна отриманого направлення в залежності від стану направлення;
11. Вигляд модального вікна створення направлення та відправленого направлення в стані «З відповіддю»;
12. Висновки.

Студент групи 2АКІТ-22м

Олександр ПОБЕРЕЖНЯК
Підпис *Ім'я ПРІЗВИЩЕ*

Керівник

Олег КОВАЛЮК
Підпис *Ім'я ПРІЗВИЩЕ*

Магістерська кваліфікаційна робота

на тему:

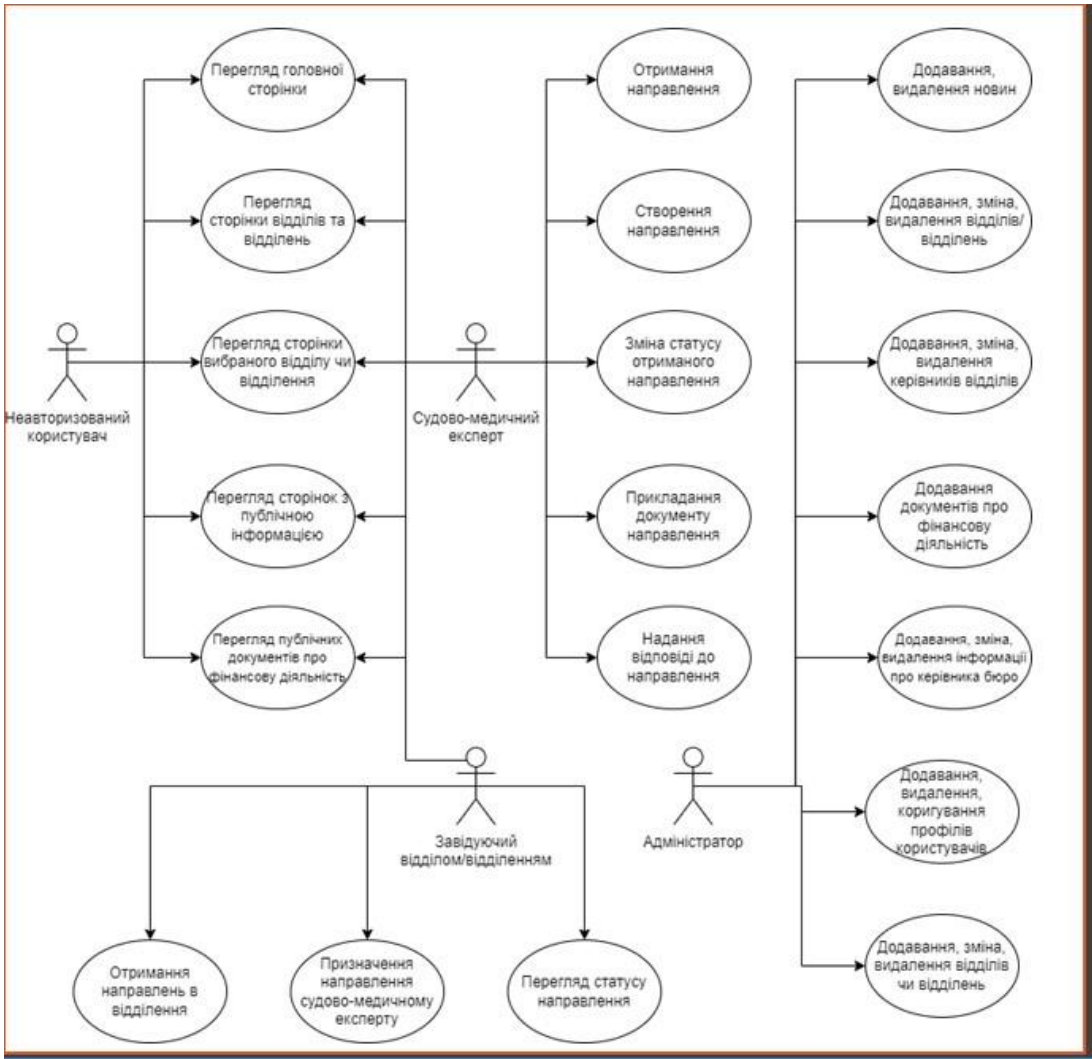
«Розробка автоматизованої системи бюро судово-медичної експертизи»

ВИКОНАВ: СТУДЕНТ ГРУПИ 2АКІТ-22М
О.Р. ПОБЕРЕЖНЯК

КЕРІВНИК: К.Т.Н., ДОЦЕНТ, ДОЦЕНТ КАФЕДРИ КСУ

О.О.КОВАЛЮК

Аналіз схожих автоматизованих систем, вибір технологій

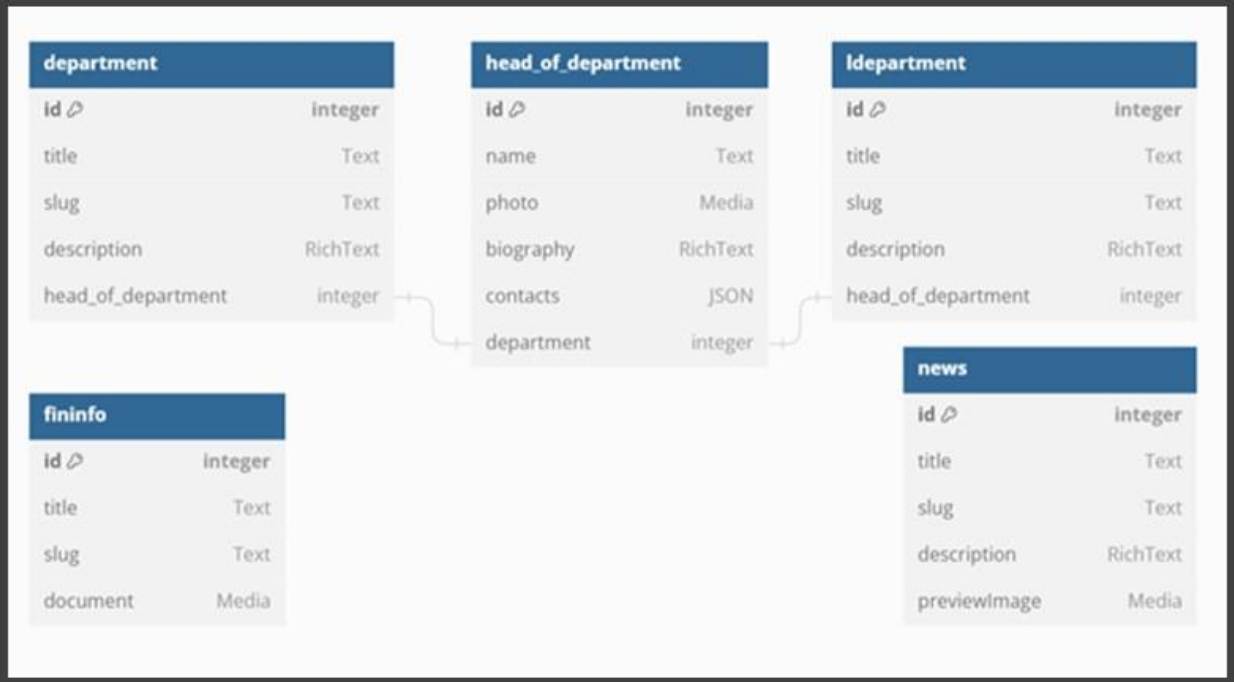


UML-діаграма варіантів використання

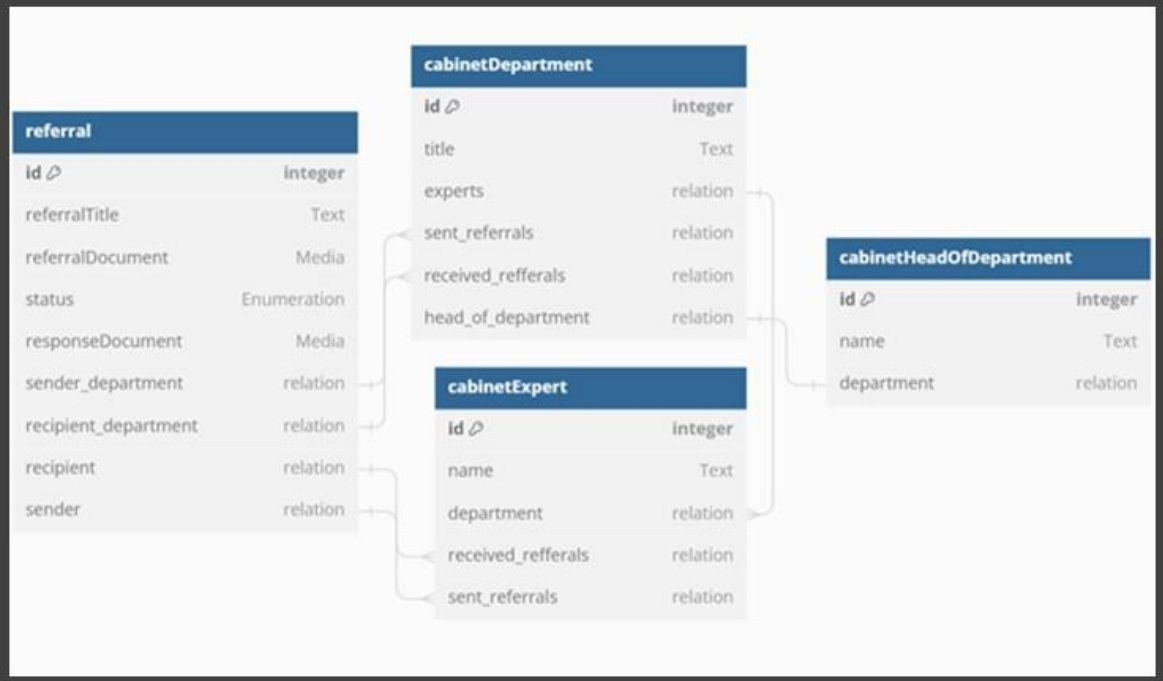
Ролі:

1. Неавторизований користувач
2. Судово-медичний експерт
3. Завідуючий відділом/відділенням
4. Адміністратор

Діаграма бази данних для типів колекцій неавторизованого користувача

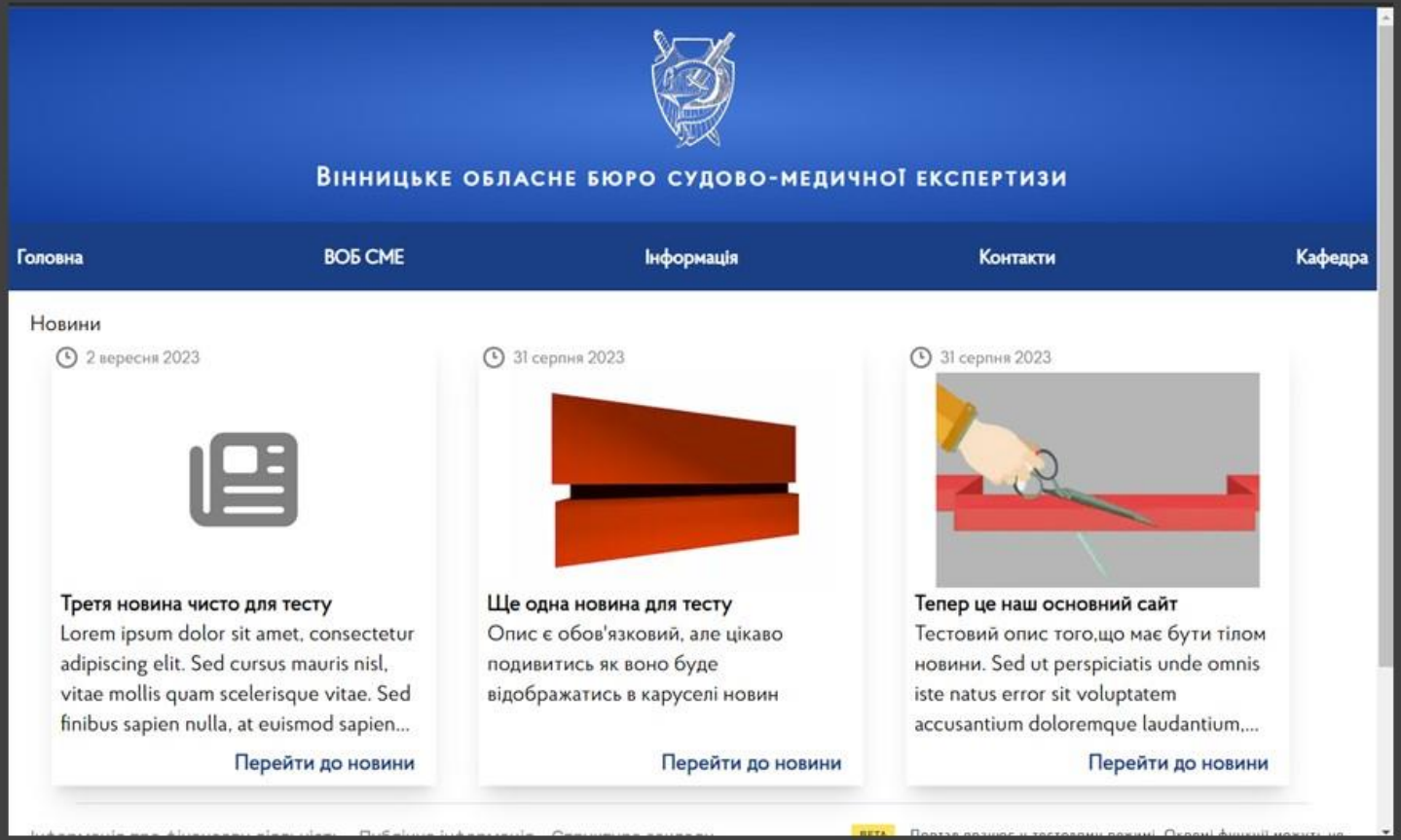


Діаграма бази данних для типів колекцій авторизованого користувача

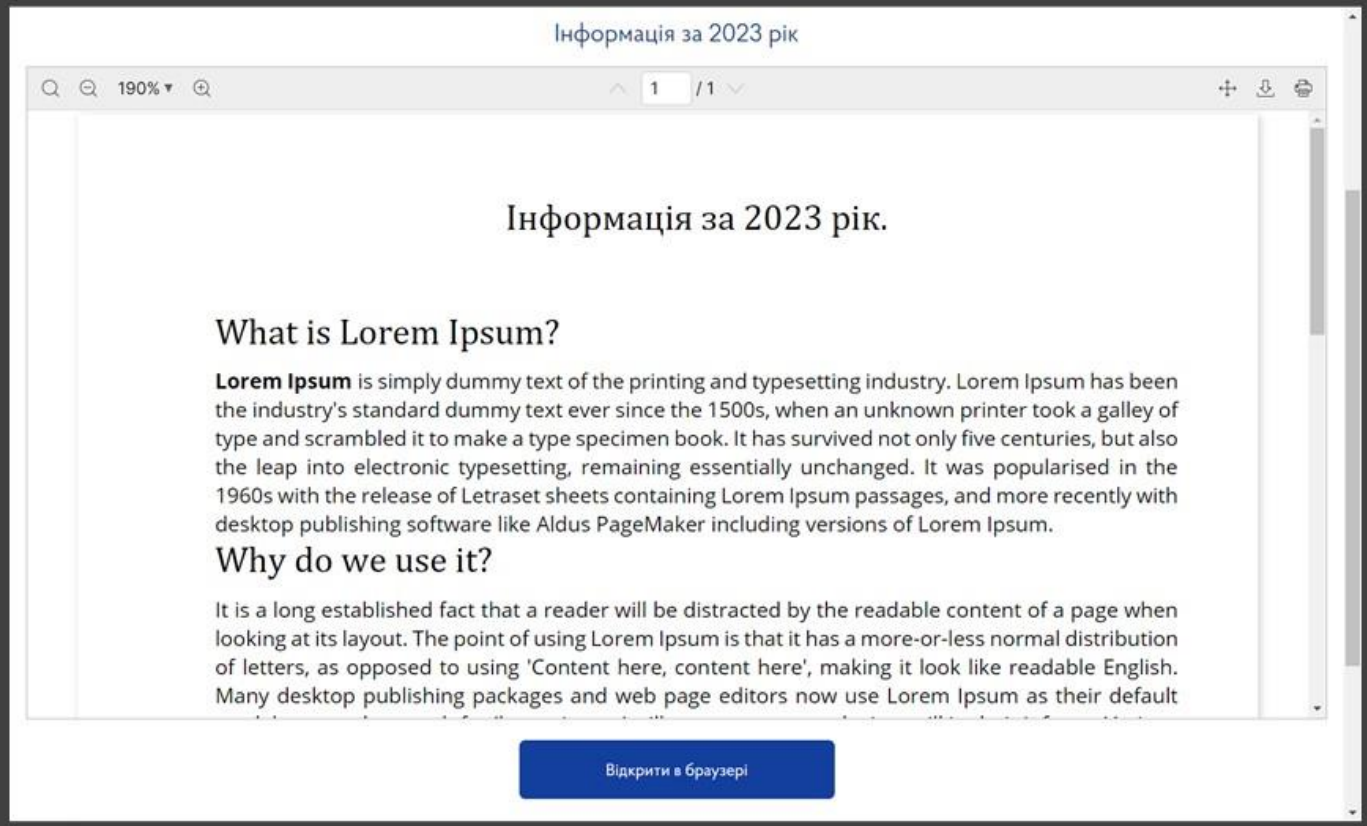


Вигляд навігаційного меню сторінок для неавторизованого користувача з відкритою вкладкою «ВОБ СМЕ»

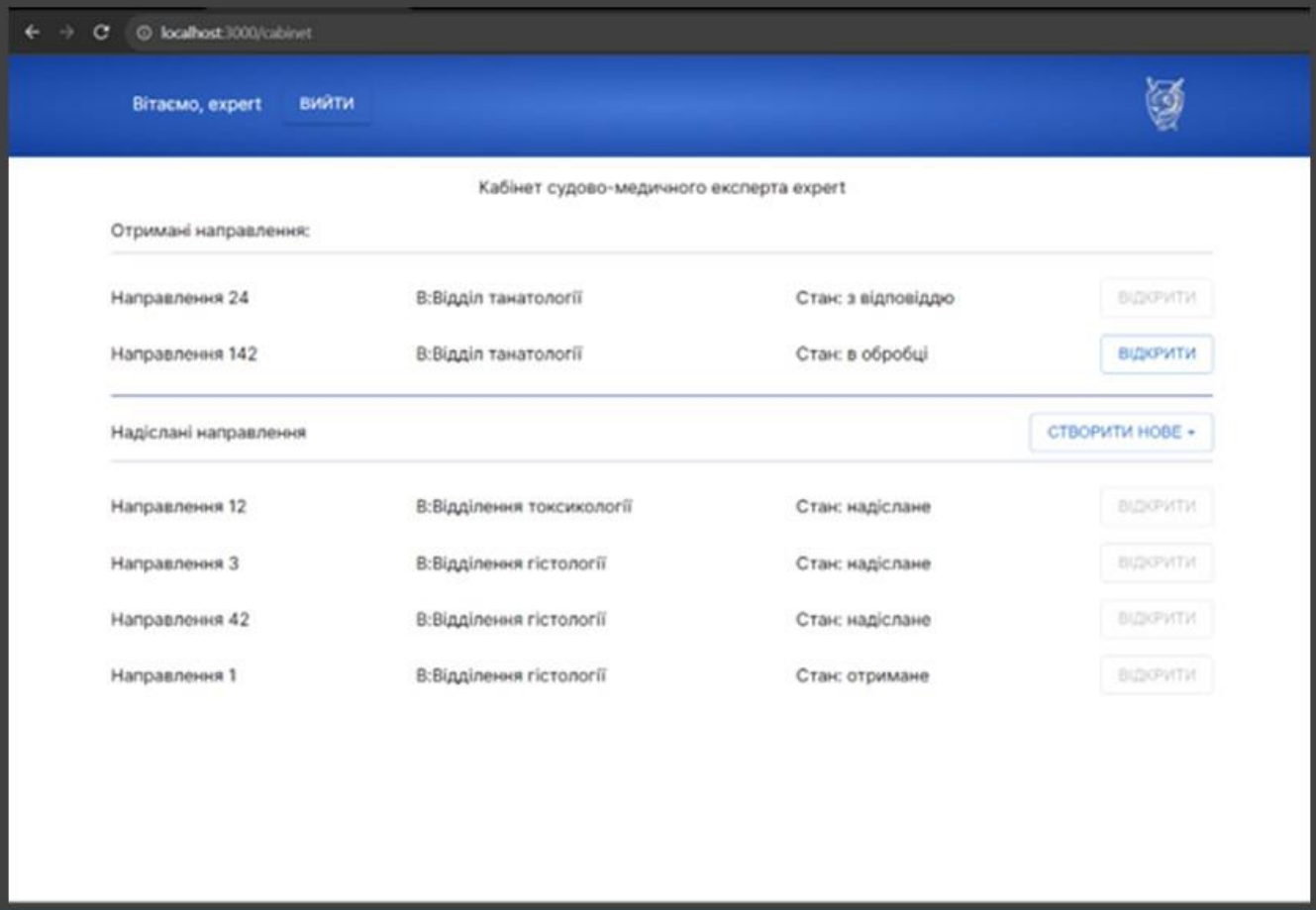
Вигляд головної сторінки для неавторизованого користувача



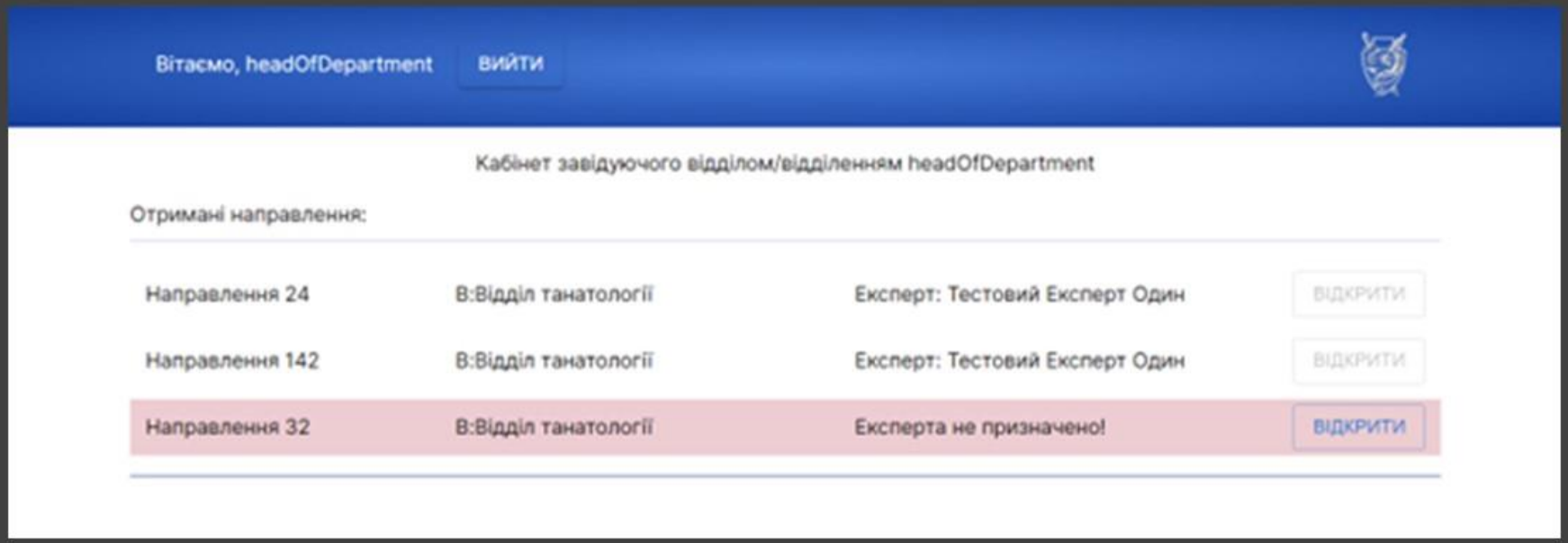
Вигляд компоненту перегляду PDF-документів для неавторизованого користувача



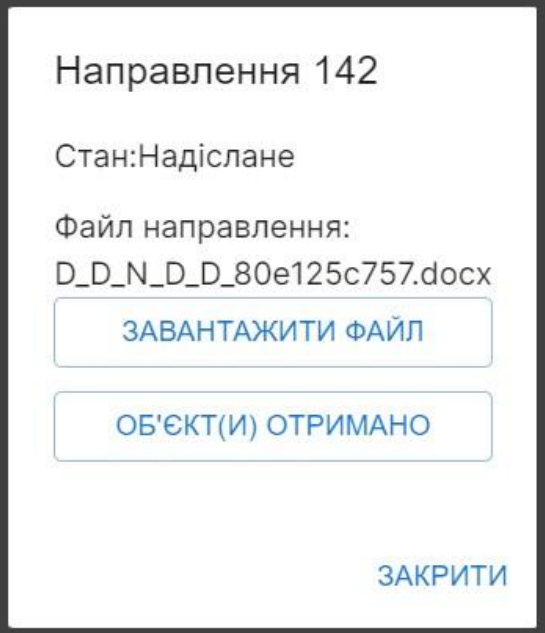
Вигляд кабінету судово-медичного експерта

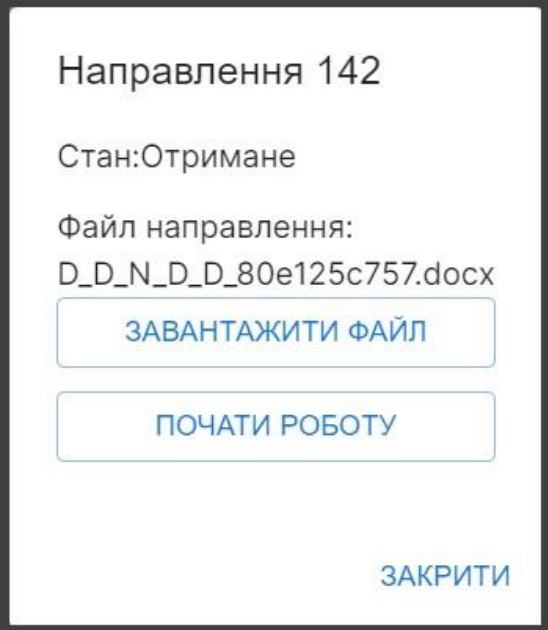


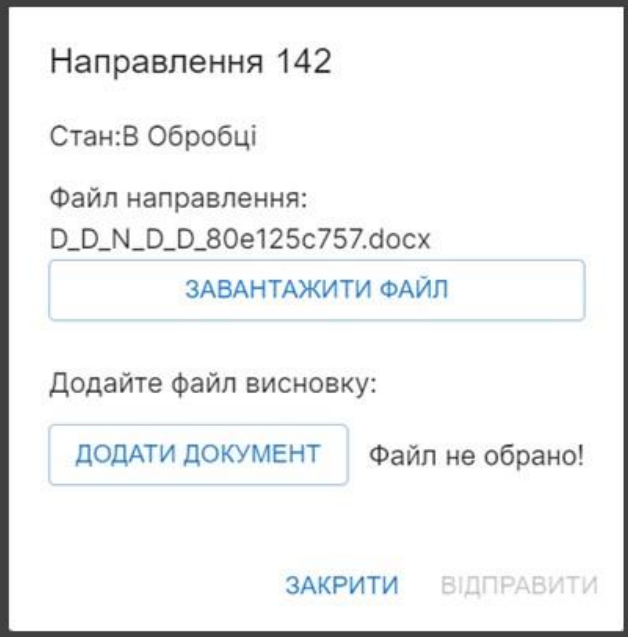
Вигляд кабінету завідуючого відділом/відділенням



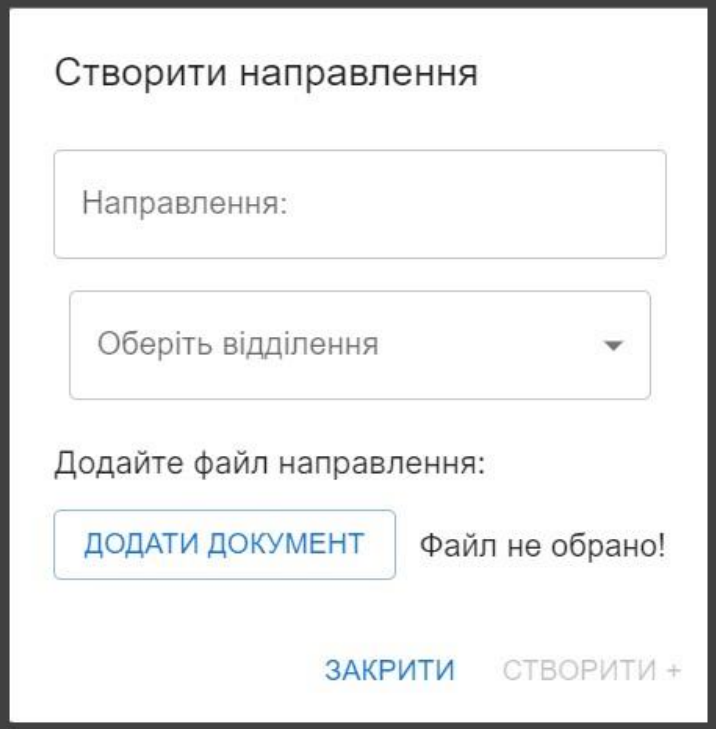
Вигляд модального вікна отриманого направлення в залежності від стану направлення

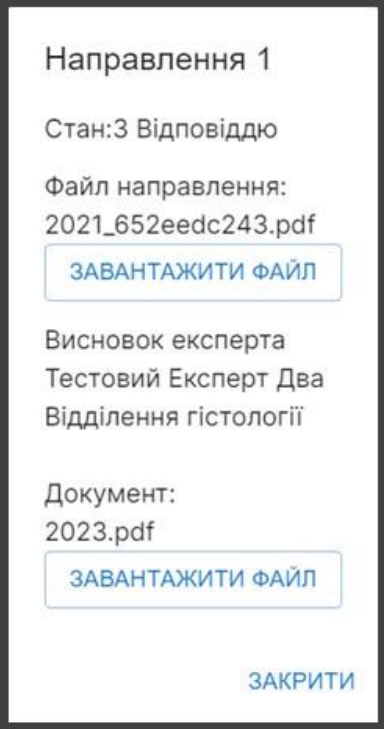






Вигляд модального вікна створення направлення та відправленого направлення в стані «З відповіддю»





Висновки

Отже, під час розробки автоматизованої системи бюро судово-медичної експертизи було оглянуто сучасний технологічний та методичний стан розробки автоматизованих систем для підприємств та установ. Проаналізовано аналогічні системи, та системи з схожим функціоналом. Проаналізовані способи та методи вирішення поставлених задач, підібрані технології та методики розробки системи.

Розроблена логіка серверної частини сайту, за яку відповідає система контролю вмістом Strapi, створено типи колекцій, з мінімально достатньою кількістю полів та зв'язків між типами колекцій. Налаштовані ролі користувачів, та права доступу до типів колекцій в залежності від ролі користувача, що користується системою. Створені діаграми таблиць бази даних для ролей «Експерт» та «Завідувач відділом/відділенням».

Розроблений front-end додаток, з використанням фреймворку Next.js. Умовно front-end було розділено на дві частини: частина для неавторизованих користувачів – частина системи, яка є веб-сайтом, на котрому користувач зможе знайти публічну інформацію про діяльність організації, контактні дані установи, адреси, список відділень та напрямки їх роботи, інформацію про завідувачів відділення, публічну інформацію про фінансову діяльність установи згідно Господарського кодексу України, та інші сторінки описані завданням; Друга умовна частина – особистий кабінет працівника установи, де судово-медичні експерти надсилають електронні направлення між відділеннями для дослідження об'єктів чи проведення експертиз іншого роду.

Система протестована списком запитів при різних умовах для різних ролей.

Було автоматизовано процес обміну направленнями, та надання відповідей по ним.