

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Система управління лікарнею»

Виконав студент групи ІПЗ-21-5
ТІМЧЕНКО Андрій Сергійович

Керівник роботи:
ВЛАСЕНКО Олег Васильович

Рецензент:
ЧИЖМОТРЯ Олексій Володимирович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна ВАКАЛЮК

«17» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ТІМЧЕНКО Андрій Сергійович**

Керівник роботи: **ВЛАСЕНКО Олег Васильович**

Тема роботи: **«Система управління лікарнею»,**

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи:

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Власенко О. В.	17.02.2025р.	17.02.2025р.
2	Власенко О. В.	17.02.2025р.	17.02.2025р.
3	Власенко О. В.	17.02.2025р.	17.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебсистеми керування лікарнею та пояснювальної записки. Пояснювальна записка до випускної роботи містить 61 сторінку, 40 ілюстрацій та 15 таблиць.

Розроблено інформаційну систему для управління роботою лікарні, яка дозволяє автоматизувати процеси обліку пацієнтів, лікарів, візитів, прийомів, спеціальностей, відділень, розкладу тощо. Система спрямована на підвищення ефективності медичних послуг, зменшення адміністративного навантаження та покращення взаємодії між медичним персоналом і пацієнтами.

Архітектура системи побудована за клієнт-серверною моделлю. Серверна частина реалізована з використанням PHP-фреймворку Symfony, який забезпечує гнучке налаштування, валідацію даних, REST API та взаємодію з базою даних. Для зберігання інформації використано СУБД MySQL. Клієнтська частина створена на основі React з компонентним підходом. Уся система контейнерізована та розгортається за допомогою Docker, що значно спрощує розробку, тестування та впровадження в різних середовищах.

У системі реалізовано багаторівневу авторизацію та автентифікацію, CRUD-операції для ключових сутностей, рольовий доступ, фільтрацію та пошук. Також підтримується управління розкладами, спеціальностями лікарів, запитами пацієнтів і історією візитів.

Система має зручний, адаптивний інтерфейс, що дозволяє легко орієнтуватися в функціоналі як медичному персоналу, так і адміністрації закладу.

КЛЮЧОВІ СЛОВА: ЛІКАРНЯ, СИСТЕМА КЕРУВАННЯ, API, REACT, SYMPHONY, MYSQL, DOCKER.

					ІПЗ.КР.Б-121-25-ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Система управління лікарнею	Лім.	Арк.	Аркушів
Розроб.		Тімченко А. С.						
Керівник		Власенко О. В.					3	61
Н. контр.		Тичина О.П.				ФІКТ Гр. ІПЗ-21-5		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's qualification thesis consists of a web-based hospital management system and an explanatory report. The explanatory report includes 61 pages, 40 illustrations, and 15 tables.

An information system was developed to manage hospital operations, enabling the automation of processes such as tracking patients, doctors, visits, appointments, specializations, departments, schedules, and more. The system is aimed at improving the efficiency of medical services, reducing administrative workload, and enhancing communication between medical staff and patients.

The system architecture is based on a client-server model. The backend is implemented using the Symfony PHP framework, which provides flexible configuration, data validation, a REST API, and database interaction. MySQL is used as the database management system. The frontend is built with React using a component-based approach. The entire system is containerized and deployed using Docker, which significantly simplifies development, testing, and deployment across different environments.

The system includes multi-level authentication and authorization, CRUD operations for key entities, role-based access control, filtering, and search functionality. It also supports schedule management, doctor specializations, patient requests, and visit history tracking.

The interface is user-friendly and responsive, allowing both medical staff and hospital administrators to easily navigate the system's functionality.

KEYWORDS: HOSPITAL, MANAGEMENT SYSTEM, API, REACT, SYMPHONY, MYSQL, DOCKER.

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ УПРАВЛІННЯ ЛІКАРНЕЮ.....	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту	10
1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	15
Висновки до першого розділу.....	18
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ЛІКАРНІ.....	19
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	19
2.2. Розробка бази даних системи	28
2.3. Проектування та реалізація алгоритмів роботи системи	35
2.4. Реалізація функціоналу системи	37
Висновки до другого розділу	41
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ УПРАВЛІННЯ ЛІКАРНІ.....	42
3.1. Порядок встановлення та налаштування параметрів системи	42
3.2. Загальна структура інтерфейсу системи	43
3.3. Тестування роботи системи управління лікарнею.....	51
Висновки до третього розділу	56
ВИСНОВКИ.....	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	58
ДОДАТКИ.....	61

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

ORM – Object-Relational Mapping

CRUD – Create, Read, Update, Delete

HTTP – HyperText Transfer Protocol

ID – Identifier

CSRF – Cross-Site Request Forgery

YAML – Yet Another Markup Language

XML – eXtensible Markup Language

JSX – javaScript XML

JSON – JavaScript Object Notation

БД – База даних

СУБД – Система управління базою даних

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми полягає у потребі впровадження сучасних цифрових рішень в організацію процесів лікування, обліку та планування у медичних закладах. Зростання обсягів інформації, збільшення кількості пацієнтів і високі вимоги до якості медичних послуг створюють потребу у впровадженні спеціалізованих інформаційних систем. Такі системи дозволяють централізовано керувати даними, забезпечують оперативність доступу до медичних записів, оптимізують адміністративні процеси, знижують навантаження на персонал та покращують взаємодію між пацієнтами, лікарями й адміністрацією. В умовах цифровізації галузі охорони здоров'я автоматизовані рішення сприяють зниженню кількості помилок, забезпечують точність у веденні історії хвороб і формуванні звітності, а також підвищують ефективність прийняття управлінських рішень.

Метою є розробка випускної кваліфікаційної роботи концепції і створення інформаційної системи керування лікарнею, яка забезпечить автоматизацію процесів обліку пацієнтів, управління медичними записами, реєстрації візитів і прийомів, адміністрування користувачів і ролей, формування розкладів лікарів та управління доступом до даних.

Для реалізації поставленої мети передбачено вирішення таких завдань:

- проаналізувати предметну область та виявити потреби медичного закладу в автоматизації;
- сформулювати вимоги до функціональності майбутньої системи;
- розробити архітектуру програмного комплексу;
- створити структуровану модель бази даних, що забезпечить логічні зв'язки між основними об'єктами системи;
- реалізувати інтерфейс користувача з урахуванням вимог до зручності та доступності;
- забезпечити авторизацію, захист персональних даних та контроль доступу до функцій системи;

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

- здійснити тестування системи для перевірки її функціональності та стійкості до помилок.

Об’єктом дослідження є процеси цифрового управління медичними закладами, які передбачають ведення обліку пацієнтів, призначення візитів, та організацію медичних процедур.

Предметом дослідження є функціональні, структурні та інтерфейсні аспекти інформаційної системи керування лікарнею, спрямованої на покращення взаємодії між пацієнтами, лікарями та адміністраторами.

У роботі використано методи системного аналізу, модульного проєктування та веброзробки.

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ УПРАВЛІННЯ ЛІКАРНЕЮ

1.1. Постановка задачі

Метою кваліфікаційної роботи бакалавра є розробка веборієнтованої інформаційної системи керування лікарнею, яка забезпечить комплексну автоматизацію адміністративних та медичних процесів, зручність у користуванні, відповідність сучасним вимогам до програмного забезпечення, а також можливість масштабування системи у майбутньому.

Передбачається реалізація комплексного рішення, яке дозволить вести облік пацієнтів, лікарів, організовувати розклад прийомів і процедур, забезпечувати доступ до статистики та звітності, а також адмініструвати доступ до системи на основі ролей. У системі мають бути передбачені різні типи, користувачів: пацієнти, лікарі, адміністратори. Кожен з яких має власний набір функцій і можливостей. Пацієнт зможе реєструватися переглядати свої прийоми, записуватися на візит або скасовувати його. Лікар матиме доступ до розкладу прийомів, встановлюватиме ціни за прийоми. Адміністратор, у свою чергу, буде керувати розкладом кабінетів, додаватиме нових лікарів, призначатиме ролі, переглядатиме загальну статистику та слідкуватиме за коректністю введених даних.

Однією з вимог до системи є зручний та інтуїтивний інтерфейс, який дозволить користувачам легко орієнтуватися у функціоналі, швидко знаходити потрібну інформацію, працювати з розкладом і картками пацієнтів. Інтерфейс повинен бути адаптивним, щоб забезпечити коректне відображення на різних пристроях: десктопах, ноутбуках, планшетах. Всі дії користувача мають супроводжуватись підтвердженням, валідацією і можливістю відкату у випадку помилки.

Ще одним важливим завданням є розмежування прав доступу, реалізоване на основі ролей. Лікарі не повинні мати доступ до адміністративних функцій, а пацієнти до даних інших користувачів. Реалізація авторизації та автентифікації

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		9

повинна гарантувати безпечну роботу із системою, виключаючи несанкціонований доступ до даних. Крім того, необхідне ведення журналу дій користувачів, що дозволяє у будь-який момент перевірити, хто, коли і які зміни вносив у систему.

Технічна реалізація має базуватися на модульному підході, який дозволяє розширювати функціонал без впливу на існуючу архітектуру. Це важливо як на етапі початкової розробки, так і під час майбутнього масштабування системи. У процесі розробки застосовуватимуться сучасні інструменти контейнеризації, зокрема Docker, що забезпечує розділення проєкту на незалежні сервіси: frontend, backend, база даних, phpMyAdmin, nginx. Це дозволить налаштовувати середовище швидко і безпомилково, а також легко відтворювати проєкт на новому обладнанні.

Для полегшення підтримки та модернізації всі етапи розробки системи мають бути задокументовані, включно з описом API, структур баз даних, прикладами запитів і логікою взаємодії між компонентами. Це дозволить у майбутньому залучати нових розробників до проєкту, адаптувати рішення під конкретні потреби закладу або інтегрувати його з іншими медичними сервісами.

У рамках цієї роботи окремі функції, як-от керування операціями або запис з дитиною, залишатимуться нереалізованими, але структура системи спроектована таким чином, щоб у майбутньому ці можливості можна було додати без необхідності глибокого переписування коду. Таким чином, система керування лікарнею буде масштабованим, гнучким і адаптивним рішенням для потреб сучасного медичного закладу, з урахуванням технічної надійності та зручності для кінцевого користувача.

1.2. Аналіз аналогів програмного продукту

Розробка системи керування лікарнею потребує детального аналізу існуючих рішень у галузі медичного програмного забезпечення. Оскільки системи такого типу активно впроваджуються в лікарнях різного рівня, важливо

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

дослідити вже реалізовані продукти, щоб виокремити їхні функціональні можливості, та врахувати ці аспекти під час проектування власного рішення.

Однією з відомих систем є Medics (рис. 1.1)

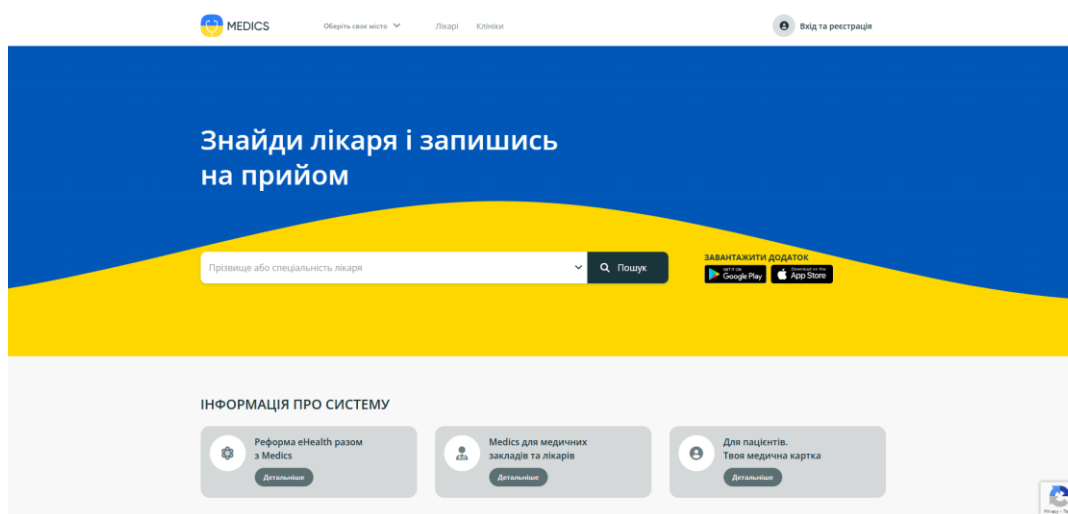


Рис. 1.1. Медична система Medics

Ця система дозволяє пацієнтам здійснювати запис на прийом до лікаря онлайн, переглядати інформацію про лікарів і заклади, а також завантажувати мобільний додаток. Medics забезпечує особистий кабінет пацієнта, де користувач може переглядати свою медичну картку. Також є функції для лікарів: ведення електронних записів, керування розкладами, перегляд історії візитів. Незважаючи на широкий функціонал, система не підтримує, наприклад, гнучкого розмежування ролей між адміністративним персоналом і лікарями, а також не має можливості повноцінного керування внутрішніми підрозділами медичного закладу.

Іншою популярною системою є OpenEMR (рис. 1.2), відкрите програмне забезпечення для електронного медичного обліку. Серед основних можливостей: ведення електронних медичних карток, планування візитів, виставлення рахунків, управління користувачами, ведення історії пацієнтів, підтримка стандартів FHIR та API-інтеграцій. OpenEMR орієнтована на повноцінну автоматизацію клінік і лікарських практик. Деякі з її можливостей, як-от підтримка стандартів HL7/FHIR або інтеграція зі сторонніми системами, не є обов'язковими для мінімального MVP, але можуть бути враховані у перспективі як додатковий функціонал.

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

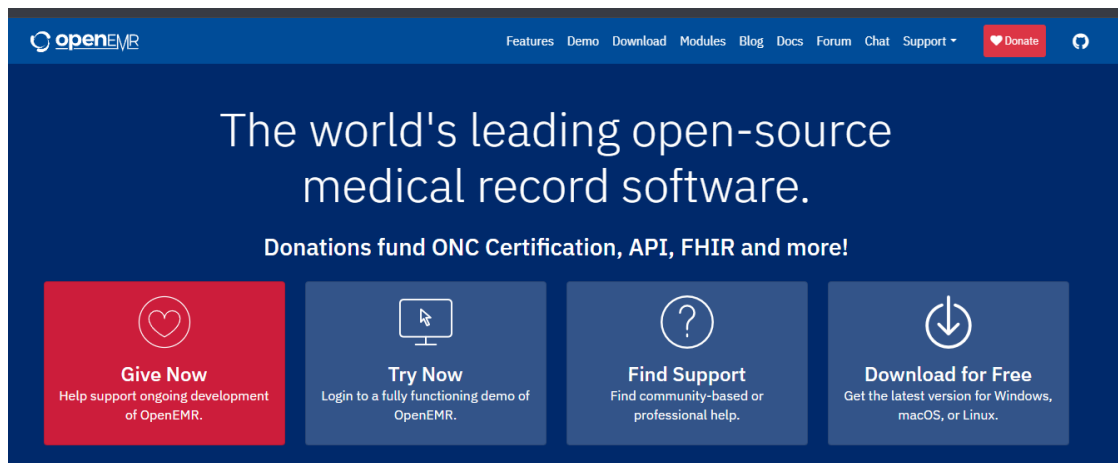


Рис. 1.2. Медична система OpenEMR

Наступною потужною медичною системою є Simplex HIMES (Рис. 1.3), яка орієнтована на управління лікарнями та клініками. Вона забезпечує широкий набір функцій для автоматизації адміністративних і клінічних процесів. Серед основних можливостей управління записами пацієнтів, електронна медична картка, планування прийомів, виставлення рахунків, облік страхування та лабораторні модулі. Система підтримує стандартні вимоги щодо безпеки та сумісності з регуляторними нормами охорони здоров'я. Особливістю Simplex HIMES є її адаптивність до потреб різних установ охорони здоров'я, включаючи державні й приватні лікарні, а також можливість інтеграції з іншими платформами через API. Інтерфейс системи оптимізований для різних ролей користувачів, зокрема лікарів, адміністраторів і бухгалтерів.

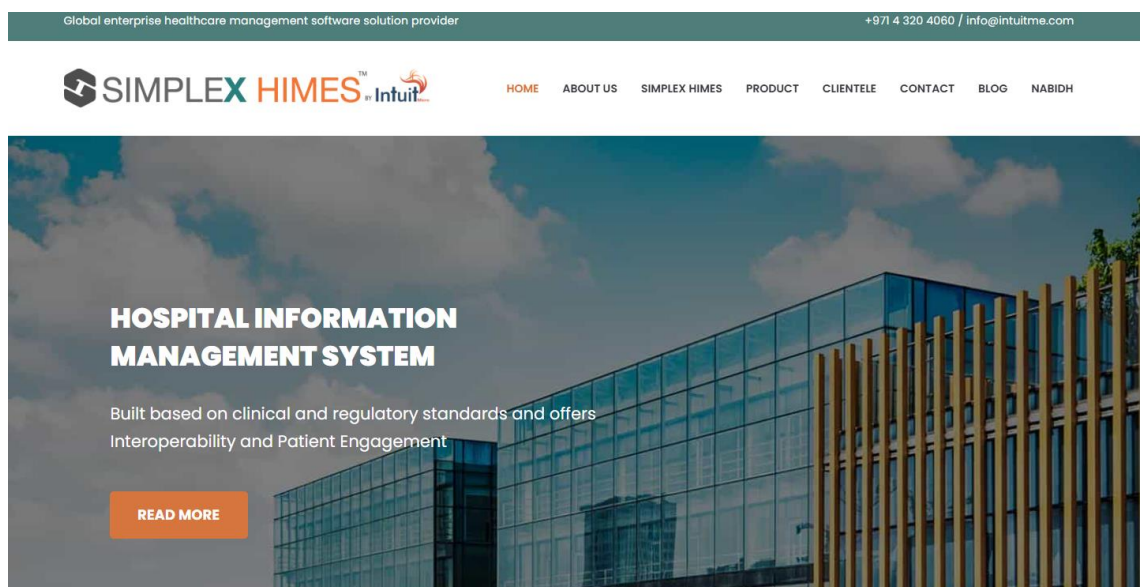


Рис. 1.3. Медична система Simplex HIMES

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

Helsi це популярна українська медична система, яка надає пацієнтам можливість онлайн-запису на прийом до лікаря, перегляду медичної інформації, історії візитів та призначень. Через особистий кабінет пацієнт може керувати записами, бачити рецепти, направлення, а також зберігати доступ до електронної медичної картки. Система також підтримує робоче середовище для лікарів, що включає ведення візитів, діагнозів, призначень і заповнення eHealth-документації. Helsi інтегрована з національною системою eHealth і відповідає українським нормативам у сфері охорони здоров'я. Окремою перевагою є простота інтерфейсу та адаптація до різних ролей користувачів, передусім лікарів і пацієнтів, проте система має обмежений функціонал для адміністраторів або повноцінного управління установою.

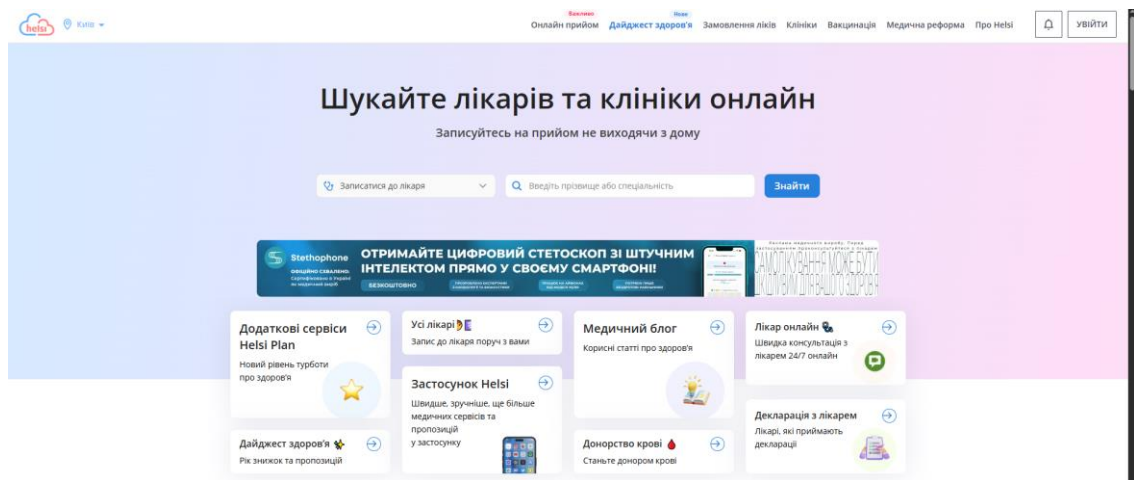


Рис. 1.4. Медична система Helsi

Виходячи з наявних функціоналів, складемо порівняльну таблицю вищезгаданих програмних аналогів (табл. 1.1).

Таблиця 1.1

Таблиця порівнянь медичних платформ

Характеристика	Medics	OpenEMR	Simplex Himes	Helsi
Електронна медкарта	+	+	+	+
Рівень безпеки даних	+	+	+	+
Кастомізація	—	+	—	—

Продовження таблиці 1.1

Хмарна архітектура	+	+	+	+
Відкритий код (Open Source)	—	+	—	—
Інтеграція з іншими системами	+	+	—	—
Підтримка eHealth	—	—	—	+
Підтримка мобільного додатку	+	—	—	+
Інтерфейс для лікарів	+	+	+	+
Інтерфейс для пацієнтів	+	—	—	+
Адміністративне управління	—	+	+	—

З аналізованих медичних платформ найбільш збалансованим за функціональністю є OpenEMR, оскільки вона пропонує відкритий код, підтримку стандартів FHIR, кастомізацію та повноцінне адміністрування. Simplex HIMES вирізняється широким функціоналом для лікарень і можливістю інтеграції з іншими системами, однак є складнішою у впровадженні. Medics забезпечує зручність для пацієнтів і наявність мобільного додатку, але має обмеження в адмініструванні. Helsi орієнтована на користувачів в Україні, має підтримку в eHealth і зручний інтерфейс, але не охоплює повноцінного управління установою.

При розробці власної системи керування лікарнею доцільно враховувати переваги OpenEMR та Helsi: відкритість, масштабованість, підтримку електронної медкарти та рольового доступу, зберігаючи при цьому простоту

інтерфейсу. Важливо також передбачити можливість інтеграції, багаторівневу авторизацію, а в майбутньому, розширення функціоналу під конкретні потреби медичного закладу.

1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Розробка веборієнтованої системи управління лікарнею потребує використання ефективного, масштабованого та безпечного технологічного стеку. На основі аналізу технічних вимог, функціонального навантаження та потреби у швидкому розгортанні системи було обрано набір інструментальних засобів, кожен із яких забезпечує надійність роботи окремих компонентів та всієї системи в цілому.

Серверна частина проєкту реалізована з використанням фреймворку Symfony, одного з найпотужніших та найпоширеніших PHP-фреймворків у світі. Symfony базується на архітектурі Model-View-Controller (MVC), що забезпечує логічне розділення функціональних частин додатку: бізнес-логіка, інтерфейс користувача та взаємодія з даними. Такий підхід не лише покращує підтримку та тестованість коду, а й дозволяє в майбутньому масштабувати або модифікувати окремі частини програми без ризику впливу на інші модулі. Однією з ключових переваг Symfony є його модульність, він складається з незалежних компонентів, які можуть використовуватись окремо або в комбінації з іншими. Завдяки цьому Symfony дозволяє точну адаптацію системи до специфіки медичної сфери, забезпечуючи контроль доступу, шифрування даних, захист від CSRF/XSS атак і підтримку багаторівневої авторизації.

Ще однією вагомою причиною обрання Symfony є його інтеграція з ORM Doctrine, що дозволяє працювати з базою даних через об'єктну модель. Це значно зменшує кількість ручних SQL-запитів, забезпечує автоматичну генерацію таблиць, обробку зв'язків між сутностями та вбудовану підтримку транзакцій. Doctrine дозволяє описати структуру таблиць у вигляді класів із анотаціями, а також легко мігрувати структуру бази в процесі розробки.

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

Для клієнтської частини обрано React, JavaScript-бібліотеку з відкритим кодом, створену компанією Meta. React дозволяє реалізувати сучасний, адаптивний та реактивний інтерфейс користувача. Основною концепцією бібліотеки є компонентність, що передбачає побудову інтерфейсу з незалежних елементів, які можна перевикористовувати, комбінувати та тестувати окремо. Це дозволяє розробляти масштабовані UI-рішення, які легко оновлюються та підтримуються. React використовує концепцію Virtual DOM, завдяки якій зміни у структурі відображення сторінки оновлюються лише частково, що значно покращує продуктивність застосунку. Крім того, підтримка JSX-синтаксису дозволяє зручно комбінувати HTML-розмітку з JavaScript-логікою, що прискорює розробку та робить код більш читабельним. Завдяки широкій екосистемі плагінів React легко інтегрується з іншими бібліотеками, наприклад, для роутингу (React Router), глобального стану (Redux) чи UI-компонентів (MUI або Bootstrap).

Для обробки вхідних HTTP-запитів, маршрутизації трафіку та забезпечення взаємодії між клієнтською і серверною частинами системи використовується nginx. Він працює як зворотній проксі, перенаправляючи запити з фронтенду до backend API, а також обслуговує статичні ресурси, що дозволяє зменшити навантаження на сервер додатка. Крім цього, nginx забезпечує базовий рівень безпеки, підтримує SSL-сертифікацію, кешування та балансування навантаження, що особливо важливо в умовах високої кількості запитів до системи.

Для зберігання інформації використовується MySQL, одна з найпопулярніших реляційних СУБД. MySQL забезпечує високу продуктивність, підтримує транзакції, індекси, зовнішні ключі та складні запити. У поєднанні з Doctrine вона дозволяє зберігати структури даних у вигляді класів, де кожна таблиця відповідає окремій сутності, а зв'язки між ними автоматично обробляються ORM-рівнем. Крім того, MySQL підтримує автоматичне створення резервних копій, що забезпечує збереження даних у разі збоїв або втрати доступу до основної бази. Також MySQL підтримує масштабування за

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

допомогою реплікації або шардінгу, що важливо при зростанні обсягів інформації.

Для візуального керування базою даних використовується phpMyAdmin вебінтерфейс для адміністрування MySQL. Він дозволяє переглядати структуру таблиць, створювати запити, переглядати журнал змін, імпортувати та експортувати дані. PhpMyAdmin зручний у використанні під час розробки та початкового етапу налаштування, оскільки дозволяє швидко коригувати вміст бази даних без використання командного рядка.

Розгортання всієї системи здійснюється за допомогою Docker платформи для контейнеризації застосунків. Docker дозволяє запускати кожен компонент проєкту в окремому контейнері: окремо frontend (React), backend (Symfony), база даних (MySQL), вебсервер (nginx), phpMyAdmin. Завдяки цьому розробник отримує незалежні середовища, що не конфліктують між собою, мають окремі залежності та легко масштабуються.

Docker дозволяє використовувати Docker Compose інструмент, який автоматично запускає всі сервіси на основі конфігураційного YAML-файлу. Це прискорює розгортання проєкту, спрощує CI/CD-процеси, дозволяє створювати однакові середовища на будь-якій машині та полегшує розгортання проєкту на сервері.

Загалом, обраний технологічний стек Symfony, React, MySQL, Docker, nginx, phpMyAdmin забезпечує відповідність сучасним вимогам до розробки вебзастосунків. Він поєднує в собі масштабованість, зручність, високу продуктивність, безпеку, а також простоту в налаштуванні та підтримці.

Висновки до першого розділу

У першому розділі було проведено аналіз предметної області системи. Визначено основні функціональні та нефункціональні вимоги до майбутньої системи, враховуючи специфіку медичної сфери та потреби користувачів.

У процесі дослідження були проаналізовані аналогічні рішення, що дозволило виявити їх сильні та слабкі сторони й сформулювати вимоги до розробки власної системи.

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

На основі отриманих вимог було обґрунтовано доцільність використання сучасних технологій, які забезпечують масштабованість, продуктивність і зручність у подальшій підтримці проєкту.

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ЛІКАРНІ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Описувана система керування лікарнею надає вебінтерфейс для адміністрації, лікарів і пацієнтів, забезпечуючи зручний доступ до функціональностей системи з різними рівнями прав доступу. Основні користувачі системи це адміністратори, які керують обліковими записами, розкладами та даними, лікарі, які ведуть запис на прийом, та пацієнти, які можуть переглядати особисту інформацію і записуватись на візити. Робота системи базується на взаємодії між цими трьома ролями, а її компоненти організовані у вигляді окремих модулів, що спрощує масштабування та підтримку. Для наочного представлення ролей та їх взаємодії була побудована діаграма варіантів використання (рис. 2.1).

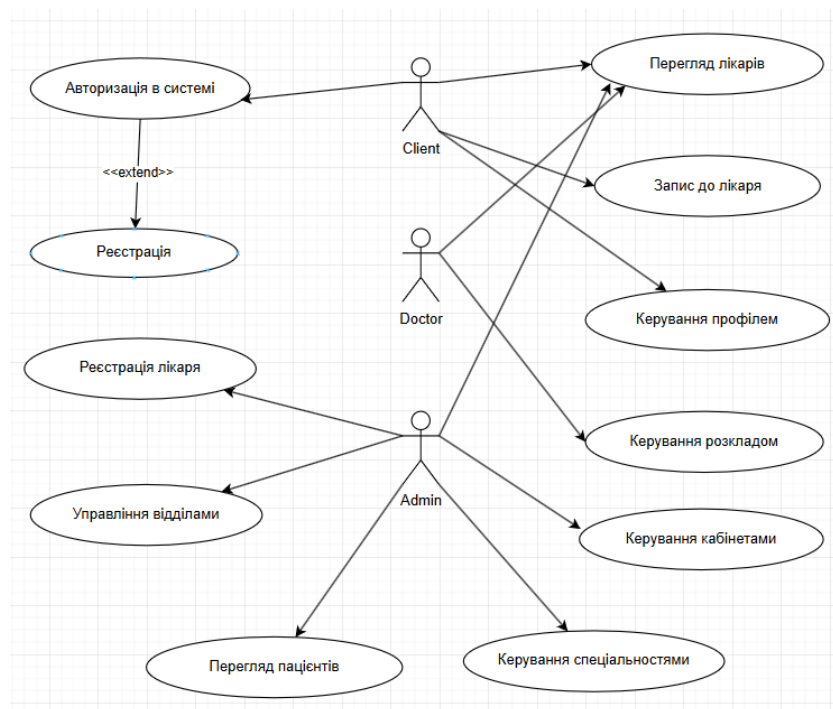


Рис. 2.1. Діаграма варіантів використання

Опис діаграми варіантів використання наведено в таблицях 2.1 – 2.6.

Таблиця 2.1

Опис прецеденту «Авторизація в системі»

Назва варіанту використання	Авторизація в системі
Основні актори	Client, Admin, Doctor
Короткий опис	Користувач вводить логін та пароль для доступу до системи та заходить на свій обліковий запис

Функціонал перегляду лікарів забезпечує користувачам зручний доступ до інформації про медичний персонал. Це дозволяє швидко знаходити потрібного лікаря за спеціальністю або відділенням та ініціювати запис на прийом. Така можливість покращує взаємодію пацієнтів із системою та сприяє ефективному управлінню медичними послугами.

Таблиця 2.2

Опис прецеденту «Перегляд лікарів»

Назва варіанту використання	Перегляд лікарів
Основні актори	Client, Doctor, Admin
Короткий опис	Користувачі мають можливість переглядати список лікарів, їх спеціальності, графіки прийому, ціни на прийоми.

Функціонал є важливою складовою системи, оскільки забезпечує користувачам зручний доступ до повної інформації про медичних спеціалістів. Завдяки цьому модулю пацієнти можуть переглядати перелік лікарів із зазначенням їхньої спеціалізації, місця прийому, графіка роботи та вартості послуг. Це дозволяє користувачам швидко знайти потрібного лікаря відповідно до власних потреб і здійснити подальші дії, зокрема запис на прийом. Для адміністратора та лікаря цей функціонал є корисним у частині перевірки актуальності інформації та координації записів пацієнтів.

Таблиця 2.3

Опис прецеденту «Керування профілем»

Назва варіанту використання	Управління профілем
Основні актори	Client
Короткий опис	Client має змогу переглядати й оновлювати власні дані профілю наприклад, ім'я, контактну інформацію пароль тощо.

Даний функціонал забезпечує користувачам персоналізований досвід взаємодії з системою. Можливість змінювати особисту інформацію, зокрема ім'я, контактні дані, пароль тощо, дозволяє підтримувати актуальність профілю та підвищує рівень безпеки облікового запису. Цей функціонал також сприяє зручності користування системою, оскільки користувач може самостійно вносити зміни без звернення до адміністратора.

Таблиця 2.4

Опис прецеденту «Керування розкладом»

Назва варіанту використання	Управління розкладом
Основні актори	Doctor
Короткий опис	Лікарі можуть налаштовувати або редагувати свій розклад прийому пацієнтів встановлювати доступні дні та години, тощо.

Керування розкладом дає можливість лікарям ефективно планувати свою роботу, визначаючи зручні для них години прийому пацієнтів. Це сприяє оптимальному розподілу навантаження, дозволяє уникати конфліктів у розкладі та забезпечує пацієнтам доступ до актуальної інформації щодо наявності вільного часу для запису. Такий підхід покращує організацію медичного процесу та підвищує якість обслуговування.

Таблиця 2.5

Опис прецеденту «Керування кабінетами»

Назва варіанту використання	Управління кабінетами
Основні актори	Admin
Короткий опис	Адміністратор має змогу додавати, змінювати або видаляти кабінети в системі, прив'язуючи їх до відділень та лікарів .

Керування кабінетами забезпечує адміністраторам централізоване управління фізичними ресурсами закладу. Можливість гнучкого додавання, редагування та видалення кабінетів дозволяє ефективно планувати розміщення лікарів, оптимізуючи використання простору. Прив'язка кабінетів до відділень і конкретних лікарів сприяє точному розподілу пацієнтів, зменшенню навантаження та покращенню організації внутрішніх процесів медичного закладу.

Таблиця 2.6

Опис прецеденту «Керування спеціальностями»

Назва варіанту використання	Управління спеціальностями
Основні актори	Admin
Короткий опис	Адміністратор керує списком медичних спеціальностей, додає нові, редагує існуючі або видаляє непотрібні, щоб їх могли обирати лікарі під час реєстрації чи оновлення профілю.

Можливість керування спеціальностями забезпечує актуальність і релевантність інформації в системі. Адміністратор може гнучко реагувати на зміну потреб закладу, додаючи нові напрямки або видаляючи застарілі. Це дозволяє лікарям точно вказувати свої компетенції, що, своєю чергою, полегшує

пацієнтам пошук фахівця відповідного профілю. Такий підхід сприяє покращенню сервісу та ефективному управлінню кадровим потенціалом.

Система управління призначена для автоматизації процесів управління медичним закладом, забезпечуючи зручний доступ до функціоналу як для пацієнтів, так і для лікарів та адміністраторів. У системі реалізовано багаторівневу модель доступу: користувачі поділяються на клієнтів (пацієнтів), лікарів та адміністраторів. Клієнти можуть переглядати доступних лікарів, їхні спеціальності, розклад прийому, а також записуватись на консультації. Лікарі мають змогу керувати своїм розкладом та переглядати пацієнтів, записаних на прийом. Адміністратори відповідають за реєстрацію лікарів, управління відділеннями, кабінетами, спеціальностями та повний контроль за інформацією, що стосується користувачів системи.

Основна ціль розробки інформаційної системи керування лікарнею полягає у створенні сучасного цифрового середовища для автоматизації основних процесів медичного закладу. Система покликана забезпечити ефективний облік пацієнтів, лікарів, спеціальностей, розкладів прийому та інших ресурсів. Вона сприяє оптимізації щоденних операцій лікарні, покращенню координації між персоналом і підвищенню загальної якості надання медичних послуг. Головна мета полягає у підвищенні точності ведення даних, зменшенні кількості паперової роботи та покращенні доступу до медичної інформації. Система дозволяє лікарям зручно працювати з розкладом, пацієнтами переглядати інформацію про прийоми, а адміністраторам керувати структурою закладу. Такий підхід створює надійний фундамент для цифрової трансформації медичного закладу та його подальшого розвитку.

Неzareєстровані користувачі мають обмежений доступ до функціональності системи. Вони можуть переглядати базову інформацію про лікарів прізвище, спеціальність, розклад прийому, наявність вільних слотів, вартість послуг. Також доступний пошук за відділеннями та спеціальностями. Створення облікового запису потрібне для подальшої взаємодії з системою запису на прийом, редагування профілю тощо.

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

Зареєстровані клієнти мають змогу проходити авторизацію, переглядати та оновлювати профіль, записуватись на прийом до лікаря, переглядати історію своїх звернень та візитів. Вони можуть обирати лікарів за фільтрами, бачити доступні дні та години прийому, скасовувати записи.

Користувач з роллю лікаря має доступ до власного розкладу, який він може редагувати. Лікар бачить список призначених візитів, може відкоригувати ціну візитів та їх тривалість. Деякі дії, як-от зміна переведення до іншого відділення, можливі лише через адміністратора.

Адміністратор володіє розширеними правами доступу. Він може реєструвати нових лікарів, керувати розкладами, кабінетами, відділеннями та спеціальностями. Також адміністратор переглядає інформацію про пацієнтів і лікарів, має можливість редагувати або блокувати облікові записи лікарів, слідує за цілісністю структури медичного закладу та відповідає за підтримку коректної роботи системи.

Функціональні вимоги до системи передбачають реалізацію процесів реєстрації та авторизації для різних ролей користувачів (клієнт, лікар, адміністратор), створення та редагування профілів, перегляд списку лікарів, їх профілів, спеціальностей і розкладу прийому, можливість запису на прийом, скасування або редагування візиту. Крім того, система підтримує управління розкладом лікарів, адміністрування кабінетів, відділень і спеціальностей, перегляд списку пацієнтів, отримання системних повідомлень (наприклад, про успішну реєстрацію або запис), а також підтримку різних рівнів доступу до функцій залежно від ролі.

Нефункціональні вимоги системи охоплюють зручність, швидкодію, доступність. Інтерфейс має бути інтуїтивно зрозумілим, дозволяти новим користувачам повністю опанувати систему за 5-10 хвилин, а досвідченим менш ніж за 3 хвилини. Середній час відгуку на дії користувача не повинен перевищувати 5 секунди, а у складних операціях 10 секунд. Інтерфейс має бути адаптивний.

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

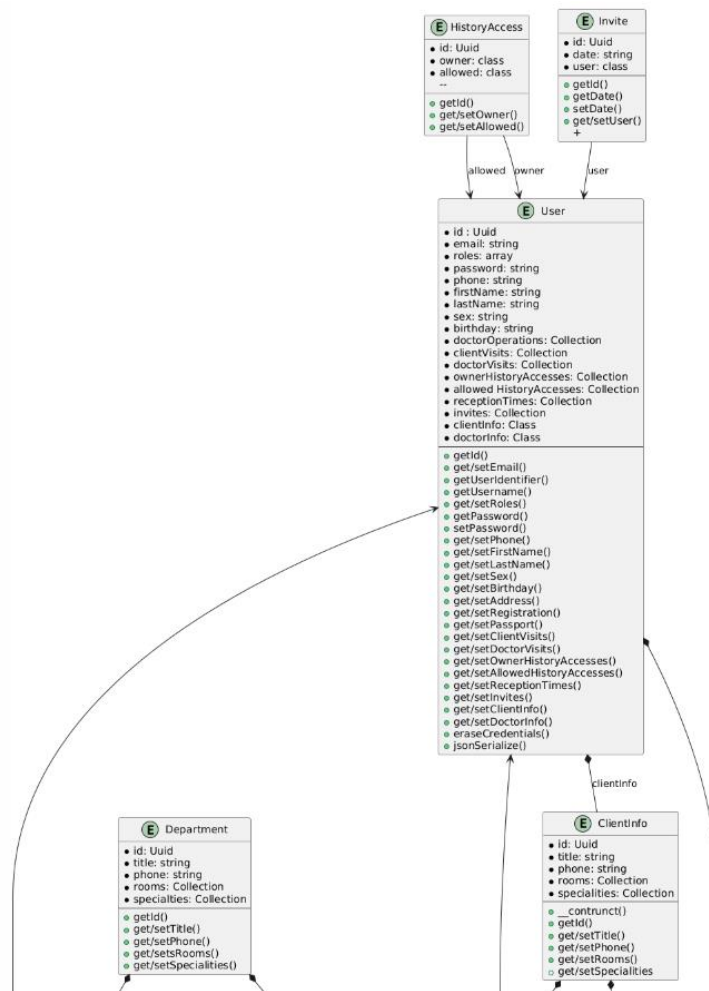


Рис. 2.2. Діаграма класів (частина 1)

Клас User призначений для зберігання та обробки основної інформації про будь-якого користувача системи пацієнта, лікаря чи адміністратора. Він реалізує методи getEmail() і setEmail() для доступу до електронної пошти, getPassword() і setPassword() для керування паролем, а також getRoles() і setRoles() для отримання і встановлення ролей користувача. Особисті дані обробляються через getFirstName(), setFirstName(), getLastName(), setLastName(), getSex(), setSex(), getBirthday(), setBirthday(), getPhone() та setPhone(). Для взаємодії з пов'язаними сутностями система використовує getClientInfo() і getDoctorInfo(), а також методи для отримання списків прийомів (getClientVisits(), getDoctorVisits()), графіка (getReceptionTimes()), історії доступів (getOwnerHistoryAccesses(), getAllowedHistoryAccesses()). Метод jsonSerialize() визначає, які саме поля об'єкта User будуть включені до JSON-представлення, приховуючи чутливу

інформацію. Метод eraseCredentials() очищає тимчасові або конфіденційні дані (наприклад, пароль) після автентифікації для підвищення безпеки.

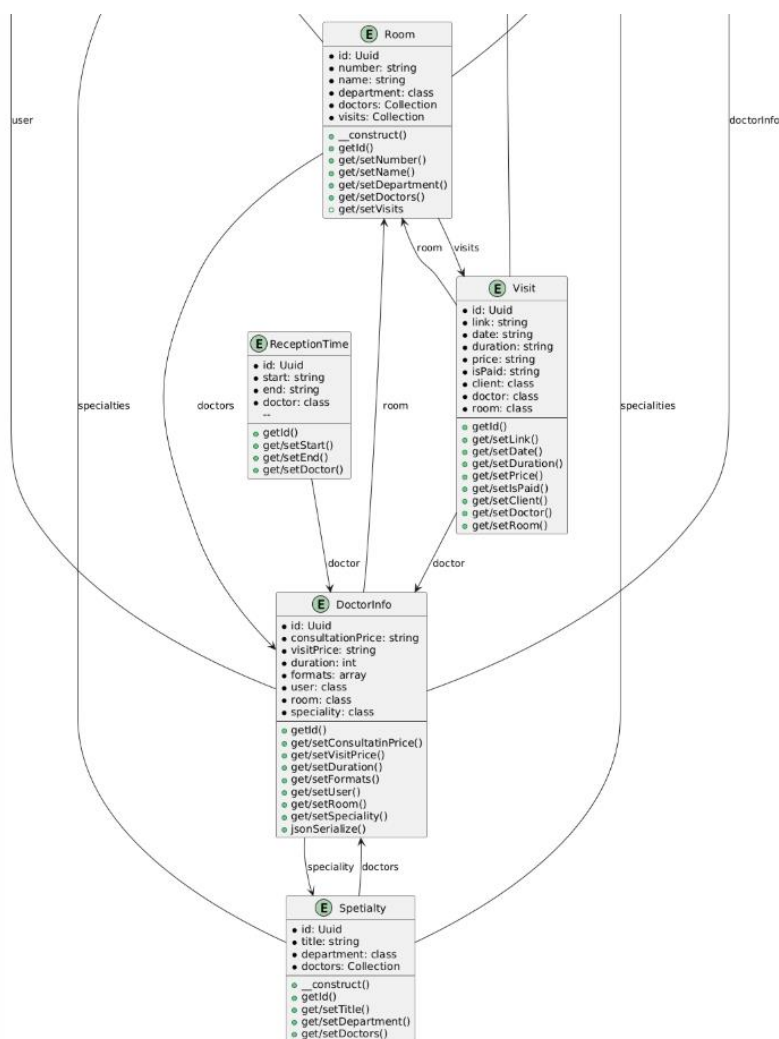


Рис. 2.3. Діаграма класів (частина 2)

Клас DoctorInfo зберігає службову інформацію про лікаря, яка використовується для розрахунку вартості, тривалості прийомів і зв'язку з профілем користувача. Методи getConsultationPrice() і setConsultationPrice() встановлюють ціну консультації, getVisitPrice() і setVisitPrice() вартість повного візиту, getDuration() і setDuration() тривалість прийому. Формат візитів задається методами getFormats() і setFormats(). Для встановлення зв'язків із зовнішніми сутностями використовуються getUser() / setUser(), getRoom() / setRoom() та getSpeciality() / setSpeciality().

Клас Visit потрібен для представлення одного конкретного запису на прийом між лікарем і пацієнтом. Він включає методи getDate() і setDate() для задання дати візиту, getDuration()/setDuration() для його тривалості, getPrice()

/setPrice() для вартості, а також getIsPaid() / setIsPaid() для статусу оплати. Зв'язки з пацієнтом і лікарем реалізуються через getClient() / setClient() та getDoctor() / setDoctor(), а кабінет візиту визначається через getRoom() / setRoom().

Клас ReceptionTime описує графік доступності лікаря, тобто проміжки часу, коли він може приймати пацієнтів. Методи getStart() і setStart() задають початок прийому, getEnd() і setEnd() кінець. Метод getDoctor() повертає лікаря, а setDoctor() дозволяє його встановити.

Клас Room використовується для опису медичних кабінетів або інших приміщень у лікарні. Він включає методи getNumber() і setNumber() для номера кімнати, getName() / setName() для назви. Зв'язок із відділенням реалізується через getDepartment() / setDepartment(). Для перегляду лікарів, які працюють у цій кімнаті, та візитів, що там проходили, використовуються методи getDoctors() і getVisits(). Метод __construct ініціалізує об'єкт, створюючи порожні колекції для пов'язаних лікарів і візитів.

Клас Department представляє медичне відділення та відповідає за логічне групування кімнат і спеціальностей. Методи getTitle() і setTitle() керують назвою відділення, getPhone() і setPhone() його контактним номером. Також передбачено getRooms() і getSpecialities() для доступу до пов'язаних приміщень і спеціалізацій.

Клас Specialty визначає медичну спеціалізацію та пов'язаний з відповідним відділенням і лікарями. Його методи getTitle() / setTitle() дозволяють керувати назвою, getDepartment() / setDepartment() відділенням, а getDoctors() лікарями, які мають цю спеціальність. Метод __construct ініціалізує об'єкт, створюючи порожню колекцію для пов'язаних лікарів.

Клас ClientInfo зберігає додаткову інформацію про пацієнта, яка не входить до основного профілю. За допомогою методів getTitle() і setTitle() задається опис, getPhone() і setPhone() номер телефону, getRooms() і getSpecialities() закріплені кімнати та спеціальності. Метод getUser() використовується для зв'язку з основним обліковим записом користувача. Метод

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

__construct ініціалізує об'єкт, створюючи порожні колекції для пов'язаних кімнат і спеціальностей.

Клас HistoryAccess забезпечує механізм делегування доступу до медичних даних одним користувачем іншому. Методи getOwner() і setOwner() встановлюють власника інформації, а getAllowed() і setAllowed() того, хто отримує дозвіл на перегляд чи використання цієї інформації.

2.2. Розробка бази даних системи

Для збереження файлів та інформації у додатку використовується реляційна СУБД MySQL. Вона забезпечує ефективну організацію, надійне збереження та швидкий доступ до даних, що стосуються користувачів, лікарів, спеціальностей, відділів, кабінетів, а також клієнтів, візитів та розкладів приймань. Таблиці пов'язанні зовнішніми ключами для їх інтеграції.

Основні таблиці описані на таблицях 2.7 – 2.15.

Таблиця 2.7

Таблиця User

User		
Поле	Тип	Призначення
id	binary(16)	Ідентифікатор
email	varchar(180)	Email користувача
roles	json	Ролі користувача
password	varchar(255)	Пароль
phone	varchar(20)	Телефон
first_name	varchar(45)	Ім'я
last_name	varchar(45)	Прізвище
sex	varchar(45)	Стать
birthday	bigint	Дата народження

Таблиця User призначена для зберігання інформації про користувачів, які взаємодіють із системою управління лікарнею. Вона включає основні дані, необхідні для реєстрації, авторизації, визначення ролей користувача, а також підтримки персоналізованого доступу до функцій системи.

У структурі таблиці містяться такі ключові поля, як ідентифікатор id, email, пароль, телефон, ім'я, прізвище, стать та дата народження користувача. Поле roles, реалізоване у форматі JSON, дозволяє зберігати кілька ролей для одного користувача. Ця таблиця є базовою в системі, оскільки забезпечує ідентифікацію та керування правами доступу для всіх учасників процесу — як пацієнтів, так і медичного персоналу.

Таблиця 2.8

Таблиця doctor_info

doctor_info		
Поле	Тип	Призначення
id	binary(16)	Ідентифікатор
user_id	binary(16)	ID користувача
room_id	binary(16)	ID кабінету
speciality_id	binary(16)	ID спеціальності
consultation_price	decimal(7,2)	Ціна консультації
visit_price	decimal(7,2)	Ціна візиту
duration	int	Тривалість

Таблиця doctor_info призначена для зберігання детальної інформації про лікарів, зареєстрованих у системі управління лікарнею. Вона доповнює дані користувача (user) медичними параметрами, що стосуються професійної діяльності лікаря та його доступності для пацієнтів.

У таблиці зберігається зв'язок із загальним профілем користувача через поле user_id, а також зазначається кабінет (room_id) та спеціальність (speciality_id). Окрім цього, таблиця містить важливі поля для фінансової та організаційної взаємодії вартість консультації (consultation_price), вартість візиту (visit_price) та середня тривалість прийому (duration). Це дозволяє системі

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

точно керувати записами на прийом, розраховувати вартість послуг і будувати гнучкий графік роботи для кожного лікаря.

Таблиця 2.9

Таблиця Speciality

Spetialty		
Поле	Тип	Призначення
id	binary(16)	Ідентифікатор спеціальності
department_id	binary(16)	Зовнішній ключ до таблиці department
title	varchar(255)	Назва спеціальності

Speciality призначена для зберігання інформації про медичні спеціальності, що існують у межах лікарні. Вона є довідниковою і використовується для класифікації лікарів за напрямками їхньої професійної діяльності. Кожен запис у таблиці містить унікальний ідентифікатор спеціальності (id), її назву (title) та зв'язок із відповідним відділенням лікарні через поле department_id. Таким чином, таблиця забезпечує узгодженість між структурними підрозділами медичного закладу та спеціалізацією персоналу, що дає змогу системі коректно формувати графіки, а маршрутизацію пацієнтів.

Таблиця 2.10

Таблиця Department

Department		
Поле	Тип	Призначення
id	binary(16)	Ідентифікатор відділення
title	varchar(255)	Назва відділення
phone	varchar(20)	Контактний номер телефону

Таблиця department використовується для зберігання інформації про структурні підрозділи лікарні, такі як кардіологічне, терапевтичне чи хірургічне відділення. До таблиці входять поля з унікальним ідентифікатором відділення (id), його назвою (title) та контактним номером телефону (phone), що забезпечує швидкий доступ до необхідної адміністративної чи медичної інформації. Таблиця відіграє важливу роль у формуванні зв'язків із таблицями speciality, room.

Таблиця 2.11

Таблиця Room

room		
Поле	Тип	Призначення
id	binary(16)	Ідентифікатор кімнати
department_id	binary(16)	Зовнішній ключ на відділення
number	varchar(10)	Номер кімнати
name	varchar(255)	Назва або опис кімнати

Таблиця room призначена для зберігання інформації про кабінети та приміщення, які використовуються в межах лікарняного закладу. Вона дає змогу структурувати фізичний простір установи, пов'язуючи кімнати з відповідними відділеннями. Основними полями таблиці є унікальний ідентифікатор кімнати (id), зовнішній ключ на відповідне відділення (department_id), номер кімнати (number) і її назва або функціональне призначення (name).

Таблиця 2.12

Таблиця client_info

client_info		
Поле	Тип	Призначення
id	binary(16)	Унікальний ідентифікатор клієнта

Продовження таблиці 2.12

user_id	binary(16)	Зовнішній ключ на таблицю user, що вказує на обліковий запис
first_name	varchar(255)	Ім'я клієнта
last_name	varchar(255)	Прізвище клієнта
gender	varchar(10)	Стать клієнта (male, female, other)
birth_date	date	Дата народження клієнта
phone_number	varchar(20)	Номер телефону

Таблиця client_info використовується для зберігання персональних даних клієнтів, які взаємодіють із системою медичного закладу. Вона дозволяє фіксувати ключову інформацію про осіб, що отримують медичні послуги. Основними полями таблиці є унікальний ідентифікатор клієнта (id), зовнішній ключ на обліковий запис (user_id), а також поля, що зберігають ім'я (first_name), прізвище (last_name), стать (gender), дату народження (birth_date) та контактний номер телефону (phone_number).

Таблиця 2.13

Таблиця visit

visit		
Поле	Тип	Призначення
id	binary(16)	Унікальний ідентифікатор візиту
client_id	binary(16)	Зовнішній ключ на таблицю client_info, що вказує на клієнта

Продовження таблиці 2.13

doctor_id	binary(16)	Зовнішній ключ на таблицю doctor, що вказує на лікаря
scheduled_time	datetime	Запланований час візиту
status	varchar(50)	Статус візиту
client_id	binary(16)	Зовнішній ключ на таблицю client_info, що вказує на клієнта

Таблиця visit призначена для зберігання інформації про заплановані візити клієнтів до лікарів у межах медичної системи. У таблиці містяться ключові поля, що дозволяють встановити зв'язок між пацієнтом і лікарем, а також фіксується дата, час і статус кожного прийому. Поле id виступає унікальним ідентифікатором візиту. Поле client_id є зовнішнім ключем на таблицю client_info, що визначає, який саме клієнт записаний на візит. Аналогічно, doctor_id встановлює зв'язок з лікарем у таблиці doctor_info, що дозволяє визначити спеціаліста, який здійснює прийом. Поле scheduled_time зберігає дату та час запланованого візиту. Поле status призначене для фіксації поточного стану візиту наприклад, scheduled, completed або cancelled, що дозволяє відстежувати історію відвідувань та планування прийомів. Ця таблиця є критичною у побудові логіки взаємодії між пацієнтами та лікарями, оскільки забезпечує повний облік всіх запланованих або завершених візитів у системі.

Таблиця 2.14

Таблиця reception_time

reception_time		
Поле	Тип	Призначення
id	binary(16)	Унікальний ідентифікатор запису про прийом

doctor_id	binary(16)	Зовнішній ключ на таблицю doctor, що вказує на лікаря
weekday	int	День тижня (1 понеділок, 7 неділя), коли лікар веде прийом
start_time	time	Час початку прийому (наприклад, 09:00:00)
end_time	time	Час завершення прийому (наприклад, 13:00:00)

Таблиця `reception_time` призначена для зберігання розкладу прийому лікарів у медичній системі. Вона містить інформацію про доступні дні та години, коли конкретний лікар здійснює прийом пацієнтів. Поле `id` є унікальним ідентифікатором запису розкладу. Поле `doctor_id` виступає зовнішнім ключем, який пов'язує кожен запис із лікарем у таблиці `doctor_info`, вказуючи, хто саме веде прийом у зазначений день. Поле `weekday` зберігає числове значення дня тижня (від понеділок до неділя), що дозволяє фіксувати розклад у прив'язці до конкретного дня. Поля `start_time` та `end_time` визначають часовий інтервал початку та завершення прийому, наприклад з 09:00 до 13:00. Завдяки цій таблиці система може формувати розклад для лікарів, враховуючи доступні години, що дозволяє ефективно планувати візити клієнтів.

Коли адміністратор або користувач створює новий запис, наприклад, додає лікаря, пацієнта чи спеціальність, система зберігає відповідні дані в таблицях `User`, `DoctorInfo`, `ClientInfo`, `Specialty`, `Department` тощо. Кожна з таблиць пов'язана зовнішніми ключами для забезпечення цілісності структури, наприклад, `doctor_info` містить посилання на `user`, `room`, `specialty`, що дозволяє відображати повну інформацію про лікаря. Після створення записів дані стають доступними для перегляду, призначення візитів та формування розкладів прийомів.

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

При додаванні візиту в таблицю Visit система автоматично зберігає зв'язок між клієнтом та лікарем через поля client_id та doctor_id, а також фіксує дату, час та статус візиту. Таблиця ReceptionTime дозволяє зберігати розклад роботи кожного лікаря, включаючи день тижня та години прийому, що забезпечує зручне планування для адміністраторів та користувачів.

Для захисту даних система обмежує доступ до конфіденційної інформації. Наприклад, поля password та roles в таблиці User не доступні для перегляду іншими користувачами та зберігаються у зашифрованому вигляді. Система реалізована таким чином, що тільки адміністратор може призначати ролі та змінювати критичні параметри користувачів.

Зчитування, фільтрація та обробка даних у системі здійснюються за допомогою SQL-запитів, які реалізуються через фреймворк Symfony. Наприклад, при побудові розкладу система виконує запит до таблиці reception_time, фільтруючи записи за doctor_id та weekday. Для визначення доступних лікарів виконується об'єднання таблиць doctor_info, specialty, department та user, що дозволяє отримати повну картину розподілу лікарів по відділеннях.

Оновлення записів (наприклад, зміна статусу візиту або оновлення профілю клієнта) також виконуються SQL-запитами з контролем прав доступу. Таким чином, система забезпечує цілісне управління всіма об'єктами лікарні з урахуванням ролей, зв'язків між таблицями та вимог безпеки.

2.3. Проектування та реалізація алгоритмів роботи системи

Інформаційна система управління лікарнею функціонує як централізована платформа для обліку, планування та взаємодії між користувачами різних ролей. Усі дії системи реалізовані у вигляді послідовностей операцій над пов'язаними сутностями, такими як користувачі, лікарі, пацієнти, відділення, спеціальності, графіки прийому, візити тощо.

У межах архітектури клієнт-сервер застосунок забезпечує логічну цілісність даних, а контроль доступу ґрунтується на перевірці ролей, закладених при реєстрації користувачів. Завдяки цьому вдається уникнути несанкціонованих змін, а всі взаємодії проходять через перевірені сценарії

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		35

використання. Крім того, система створена таким чином, щоб усі дії користувача супроводжувалися відповідними запитами до API та фіксувалися у базі даних, що забезпечує повну прозорість та відстежуваність.

Незареєстровані користувачі мають доступ до загальнодоступної інформації. Вони можуть переглядати базову структуру медичного закладу, включаючи список відділень, кабінетів, лікарів та їх спеціальностей. Наприклад, користувач може зайти на головну сторінку платформи та ознайомитись із профілями лікарів, спеціалізацію, та вартість прийому. При цьому функції запису на прийом або доступу до адміністративних розділів для них недоступні. Це дозволяє ознайомитись з можливостями платформи без створення облікового запису.

Користувачі з роллю «Client» після проходження процедури реєстрації та авторизації отримують доступ до особистого кабінету. У ньому вони можуть редагувати власну контактну інформацію, переглядати історію своїх візитів, а також заплановані прийоми. Клієнт має змогу самостійно обрати спеціальність, переглянути доступних лікарів у відповідному відділенні, ознайомитися з графіком їх роботи та обрати зручний час. Також існує можливість записатися як на фізичний прийом у лікарні, так і на онлайн консультацію. У разі вибору онлайн візиту, клієнт отримує посилання на відеоконференцію після здійснення попередньої оплати. Кожен візит супроводжується зазначенням лікаря, часу, вартості, типу консультації та статусу виконання. Усі ці дані зберігаються у відповідних сутностях бази даних і відображаються в інтерфейсі клієнта. Також клієнт може скасувати запис до певного моменту. Усі дії клієнта викликають відповідні API-запити, які обробляються бекендом, перевіряючи доступність часу, статус користувача та актуальність графіка. Це дозволяє мінімізувати ймовірність дублювання записів чи інших конфліктів.

Лікарі в системі мають доступ до функціоналу перегляду своїх прийомів. Після входу в систему їм відкривається перелік запланованих візитів, де вони можуть переглядати інформацію про пацієнта, час прийому, спосіб зв'язку та статус оплати. У своєму кабінеті лікар може також редагувати графік прийомів,

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

тобто вказувати дні тижня, години початку та завершення робочого дня, та редагувати ціни на свої послуги, та їх тривалість.

Адміністратор системи має змогу встановлювати загальні правила наприклад, тривалість консультації або вартість за одиницю часу, які відображаються в інтерфейсі лікаря та клієнта. Адміністратор володіє найбільшим обсягом прав. Він може створювати, редагувати та видаляти відділення, кабінети, спеціальності, додавати нових лікарів. При цьому адміністратор не може змінювати персональні дані клієнтів, лише переглядати їх у межах, необхідних для супроводу. Усі адміністративні дії також зберігаються в системі та можуть бути переглянуті для аудиту або у випадку спірних ситуацій.

Таким чином, система реалізує сценарії для кожного типу користувача, забезпечуючи їхні потреби, зберігаючи захист даних і дозволяючи масштабувати функціональність у майбутньому. Усі інтерфейси побудовані таким чином, щоб забезпечити просту та зрозумілу взаємодію, а технічна реалізація ґрунтується на розподілі ролей і відповідному контролі доступу.

2.4. Реалізація функціоналу системи управління лікарні

Після того, як користувач авторизувався, перед цим зареєструвавшись, як клієнт, настає напевно що, найважливіша для усієї системи, це запис до лікаря. Відповідний код розміщено у класі `CreateVisitAction`, де метод `__invoke` обробляє POST-запит, перевіряючи автентифікацію користувача, шукає об'єкти `ClientInfo` та `DoctorInfo`, створює новий екземпляр класу `Visit`, виконує валідацію й зберігає його в базу даних.

```
$visit = new Visit();  
$visit  
    ->setDate($data->getDate())  
    ->setDuration($doctorInfo->getDuration())  
    ->setPrice($doctorInfo->getVisitPrice() / (60 /  
$doctorInfo->getDuration()))  
    ->setClient($client)  
    ->setDoctor($doctor)
```

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

```

->setRoom($data->getRoom());
return $visit;

```

Для коректного запису, у лікаря має бути можливість оновлення їх графіку. Це реалізується в у методі `__invoke` класу `UpdateDoctorReceptionTimesAction`. Перед оновленням відбувається перевірка прав доступу лікаря, потім здійснюється пошук відповідного запису `ReceptionTime` і його оновлення. Це дозволяє лікареві редагувати час прийому в інтерфейсі без перезавантаження графіка повністю:

```

/** @var User $doctor */
$doctor = $this->security->getUser();
if (!in_array("ROLE_DOCTOR", $doctor->getRoles())
    || $doctor->getId() !== $data->getDoctor()->getId()) {
    throw new AccessDeniedException();
}
/** @var ReceptionTime $receptionTime */
$receptionTime = $this->entityManager
    ->getRepository(ReceptionTime::class)
    ->findOneBy(["id" => $id]);
if (!$receptionTime) {
    throw new NotFoundHttpException();
}
$receptionTime
    ->setStart($data->getStart())
    ->setEnd($data->getEnd());

```

У процесі побудови розкладу лікаря система перевіряє, чи існує розширена інформація про нього в таблиці `doctorInfo`. Це реалізується за допомогою запиту до репозиторію `DoctorInfo`, де здійснюється пошук запису, що відповідає поточному користувачу-лікарю. У випадку, якщо відповідні дані не знайдено, система генерує виключення з повідомленням про відсутність даних, необхідних для подальшої роботи. Такий механізм дозволяє запобігти помилкам, пов'язаним із неповною інформацією, і забезпечує цілісність даних у системі.

```

$doctorInfo= $this->entityManager->getRepository(DoctorInfo::class)
    ->findOneBy(['user' => $doctor->getId()]);

```

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		38

```

if (!$doctorInfo) {
    throw new UnprocessableEntityHttpException("No doctorInfo found
for given user");
}

```

Окрему увагу в реалізації приділено процесу створення нового користувача. В методі `__invoke` визначено основну логіку ініціалізації даних для новоствореного облікового запису.

```

public function __invoke(User $data): User
{
    if (!$data->getPlainPassword()) {
        return $data;
    }

    $data = $this->hashUserPassword($data);

    if (empty($data->getRoles())) {
        $data->setRoles([User::ROLE_CLIENT]);
    }

    if (empty($data->getClientInfo())) {
        $clientInfo = new ClientInfo();
        $clientInfo->setUser($data->setFamily(null)-
>setIsParent(false);
        $this->entityManager->persist($clientInfo);
    }

    return $data;
}

```

Спочатку перевіряється, чи задано пароль, далі виконується хешування. У випадку відсутності ролі, за замовчуванням призначається роль клієнта. Якщо до користувача ще не прив'язано об'єкт `ClientInfo`, він створюється й пов'язується з користувачем, після чого зберігається в базі.

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

Якщо звертатися до frontend, то важливою частиною є функціонал оновлення особистої інформації користувача в особистому кабінеті. Він реалізований у методі PatchUserInfo, що міститься у файлі CabinetContainer.jsx. Цей метод дозволяє користувачеві змінити свої дані та надіслати оновлення на сервер без необхідності перезавантаження сторінки.

```
const patchUserInfo = () => {
  setLoading(true); // Активується індикатор завантаження

  axios
    .patch("/api/users/" + userId, editedUserInfo,
      userAuthenticationConfig())
    .then(response => {
      setNotification({
        ...notification,
        visible: true,
        type: "success",
        message: "Ваші дані успішно оновлені!",
      });
    })
    .catch(error => {
      setNotification({
        ...notification,
        visible: true,
        type: "error",
        message: "Помилка при зміні даних!",
      });
    })
    .finally(() => {
      setLoading(false); // Завантаження завершується
    });
};
```

На початку методу встановлюється індикатор завантаження, який сигналізує про початок процесу обробки запиту. Далі за допомогою бібліотеки Axios відправляється HTTP-запит методом PATCH, у якому передаються нові

					ІПЗ.КР.Б-121-25-ІЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

значення властивостей користувача. Запит надсилається до відповідного маршруту бекенда, що обробляє зміну даних. У разі успішного оновлення користувачу відображається повідомлення про успіх, яке підтверджує збереження нових даних. Якщо виникає помилка (наприклад, через помилку валідації або проблеми з підключенням), виводиться повідомлення про помилку з відповідним типом "error". Усі сповіщення формуються динамічно через локальний стан компонента.

Висновки до другого розділу

У межах розробки системи керування лікарнею проаналізовано вимоги до системи та спроектовано об'єктно-орієнтовану структуру з описом сутностей, їх зв'язків і взаємодій. Розроблено реляційну модель бази даних на основі MySQL для зберігання інформації про користувачів, лікарів, візити, спеціалізації, графіки прийому тощо. Описано алгоритми роботи для кожної ролі користувача та розглянуто реалізацію ключового функціоналу: створення й обробку записів на прийом, керування графіком лікаря, реєстрацію користувачів, контроль доступу до ресурсів в залежності від ролі.

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС КОРИСТУВАЧА ТА ПОРЯДОК РОБОТИ З ХМАРНИМ СХОВИЩЕМ З НАСКРІЗНИМ ШИФРУВАННЯМ

3.1. Порядок встановлення та налаштування параметрів застосунку

Встановлення та налаштування системи управління лікарнею передбачає розгортання всіх її компонентів у контейнеризованому середовищі Docker. Це включає налаштування мережі між сервісами та забезпечення коректної взаємодії між клієнтською частиною, серверною логікою та базою даних. Архітектура системи охоплює кілька ізольованих сервісів: фронтенд-частина реалізована за допомогою React, серверна частина написана на Symfony (PHP), база даних працює на MySQL, а phpMyAdmin використовується як зручний вебінтерфейс для керування нею. Всі ці компоненти пов'язані між собою через реверс-проксі сервер Nginx, який виконує маршрутизацію запитів до відповідних контейнерів.

Усі сервіси розгортаються автоматизовано за допомогою docker-compose. Після клонування коду з Git-репозиторію виконується налаштування конфігураційних файлів середовища .env як для серверної, так і клієнтської частин. Тут вказуються параметри для підключення до MySQL, секрети додатку, дозволені CORS-адреси та інші змінні. Запуск усіх сервісів здійснюється однією командою з автоматичною побудовою контейнерів.

База даних ініціалізується за допомогою Doctrine Migrations, які створюють необхідні таблиці (користувачі, лікарі, візити, розклади, консультації тощо). За потреби застосовується попереднє наповнення таблиць тестовими даними через Doctrine Fixtures.

Система підтримує кілька рівнів доступу: пацієнти можуть переглядати лікарів і записуватись на візити, лікарі керують власним розкладом і бачать прийоми, адміністратори мають повний контроль над усіма сутностями. Доступ захищено через middleware та механізми авторизації, а безпека запитів гарантується через JWT-токени. Система гнучко масштабується, легко

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

налаштовується й готова до розгортання як у локальному середовищі, так і на віддалених хостингах.

3.2. Загальна структура інтерфейсу користувача

Система керування лікарнею реалізована як багаторівневий вебдодаток, що забезпечує ефективну взаємодію між різними типами користувачів: адміністраторами, лікарями та пацієнтами. Інтерфейс користувача побудований з використанням бібліотеки React, що дозволяє створити динамічний, адаптивний та зручний у користуванні інтерфейс. Адміністративна частина системи реалізована через спеціальний набір компонентів, інтегрованих у загальний React-додаток, що дозволяє зручно керувати користувачами, записами, та розкладами. Серверна логіка розроблена з використанням фреймворку Symfony, що забезпечує надійність, розширюваність і захист даних. Присутня система авторизації та доступу в залежності до ролі користувача.

Взаємодія з додатком у користувача розпочинається з вхідної домашньої сторінки (рис. 3.1). У верхній частині зліва знаходиться логотип, що повертає до домашньої сторінки, а зліва кнопка входу. У головній частині зліва знаходиться панель фільтрації, над якою розміщується пошук лікарів та вибір відділення. Незареєстрований користувач може переглядати лікарів та їх графіки, але не записуватися.

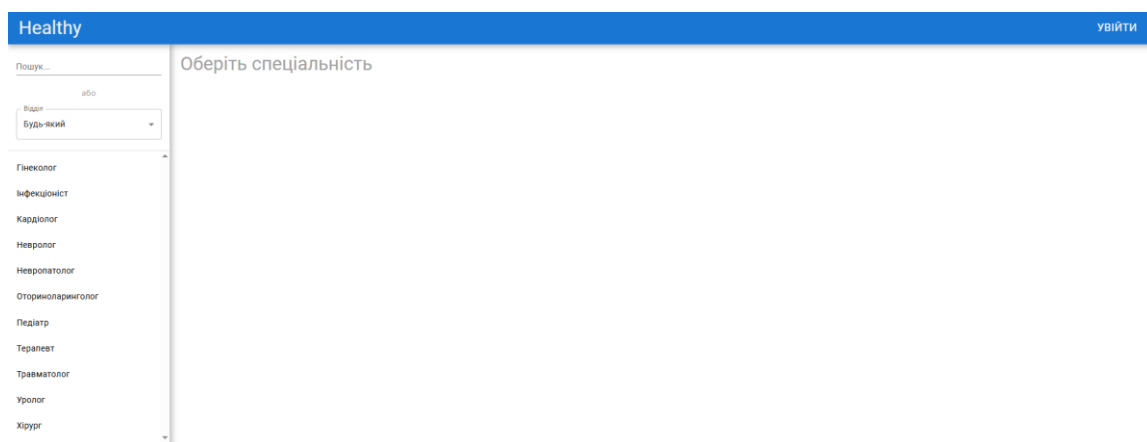


Рис. 3.1. Домашня сторінка незареєстрованого користувача

Після заходу на сторінку входу (Рис. 3.2) користувач, може ввести email та пароль, і далі в залежності від своєї призначеної ролі зайти в обліковий запис клієнта, лікаря або адміністратора. Також можна перейти на сторінку реєстрації (Рис. 3.3).

Рис. 3.2. Сторінка входу

На сторінці реєстрації можна побачити поля для: електронної пошти, пароля, телефону, ім'я, прізвища, статі та місця народження. По замовчуванню після реєстрації створюється користувач з роллю клієнт.

Рис. 3.3. Сторінка реєстрації

Після входу до облікового запису клієнта, буде видно домашню сторінку схожу до домашньої сторінки незареєстрованого користувача (Рис. 3. 4), але із кнопкою кабінету та виходу справа.

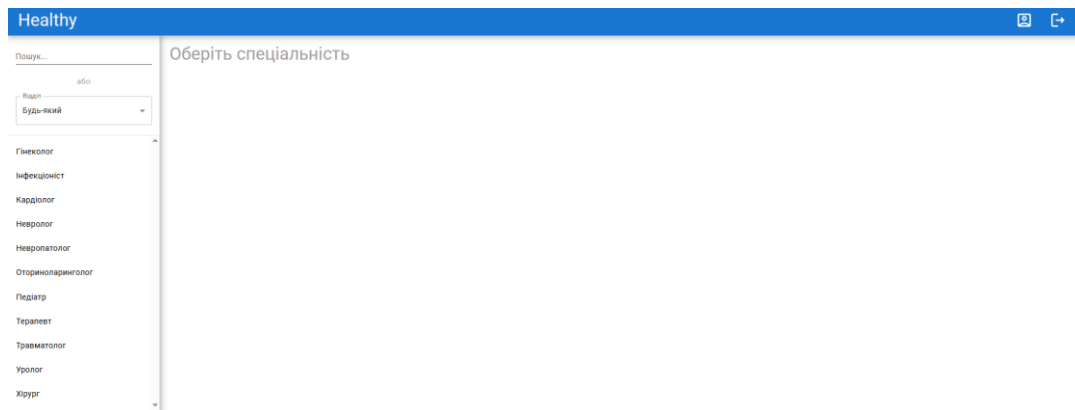


Рис. 3.4. Домашня сторінка клієнта

На сторінці кабінету клієнта можна спостерігати данні профіля та майбутні (Рис. 3.5) або записи, що вже минули (Рис. 3.7). При бажанні, запис можна скасувати (Рис. 3.6). А данні профілю, в разі чого, можна змінювати (Рис. 3.8).

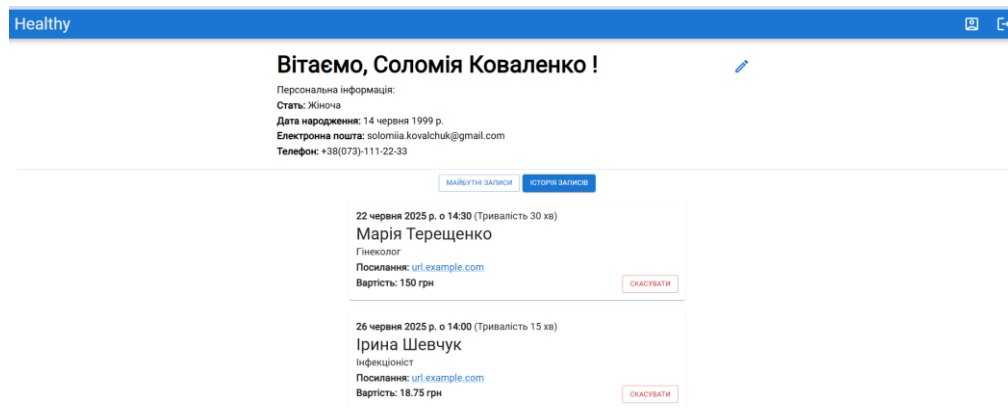


Рис. 3.5. Кабінет клієнта з майбутніми записами

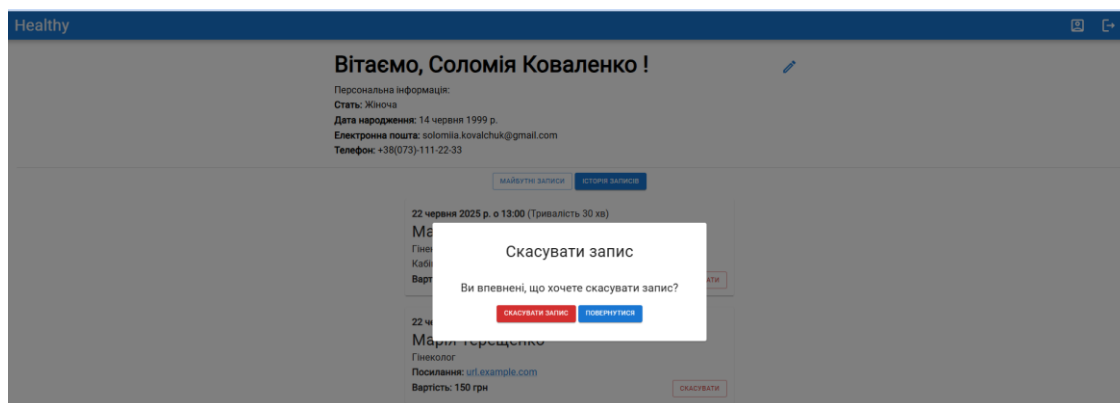


Рис. 3.6. Запит на скасування запису

Healthy

Вітаємо, Соломія Коваленко !

Персональна інформація:

Стать: Жіноча

Дата народження: 14 червня 1999 р.

Електронна пошта: solomiia.kovalchuk@gmail.com

Телефон: +38(073)-111-22-33

МАЙБУТНІ ЗАПИСИ

ІСТОРІЯ ЗАПИСІВ

13 червня 2025 р. о 12:30 (Тривалість 30 хв) Пропущено

Степан Міст

Оториноларинголог

Вартість: 200 грн

12 червня 2025 р. о 16:30 (Тривалість 30 хв) Пропущено

Яна Коваленко

Терапевт

Кабинет 104 (Кабинет терапевта №3)

Вартість: 250 грн

12 червня 2025 р. о 13:00 (Тривалість 30 хв) Пропущено

Яна Коваленко

Терапевт

Вартість: 250 грн

Рис. 3.7. Кабінет клієнта з історією записів

Healthy

Налаштування профілю

Персональна інформація:

Ім'я: Соломія

Прізвище: Коваленко

Стать: ☐ Чоловіча ☒ Жіноча

Дата народження: 06/14/1999

Телефон: +38(073)-111-22-33

ЗМІНИТИ ПАРОЛЬ

МАЙБУТНІ ЗАПИСИ

ІСТОРІЯ ЗАПИСІВ

22 червня 2025 р. о 13:00 (Тривалість 30 хв)

Марія Терещенко

Гінеколог

Кабинет 701 (Кабинет гінеколога)

Вартість: 150 грн

СКАСУВАТИ

Рис. 3.8. Редагування профілю клієнта

На домашній сторінці можна знайти доступних лікарів по певній спеціальності та записатися в їхні доступні дні, побачивши ціни за прийом та консультацію (Рис. 3.9). Вибравши доступну дату можна вибрати час запису (Рис. 3.10). Після чого з'являється запит на підтвердження запису на прийом або консультацію (Рис. 3.11).

Healthy

Пошук...

або

Вид: Будь-який

Гінеколог

Інфекціоніст

Кардіолог

Невролог

Невропатолог

Оториноларинголог

Педіатр

Терапевт

Травматолог

Уролог

Хірург

Гінеколог

Марія Терещенко

Прийом/Консультація

701

Кабинет

Марія Терещенко

Прийом / Консультація

Телефон: +38(038)-050-02-02

Кабинет: 701 (Кабинет гінеколога)

Ціна за консультацію: 175 грн

Ціна за прийом: 300 грн

Обрати дату

MM/DD/YYYY

June 2025

S

M

T

W

T

F

S

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

Рис. 3.9. Дані про для запису до певного лікаря та його доступні дні

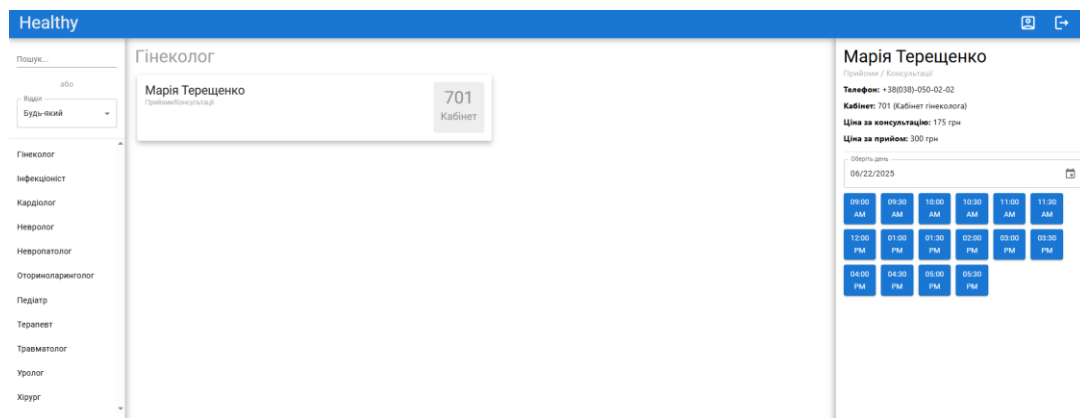


Рис. 3.10. Доступні проміжки часу для запису

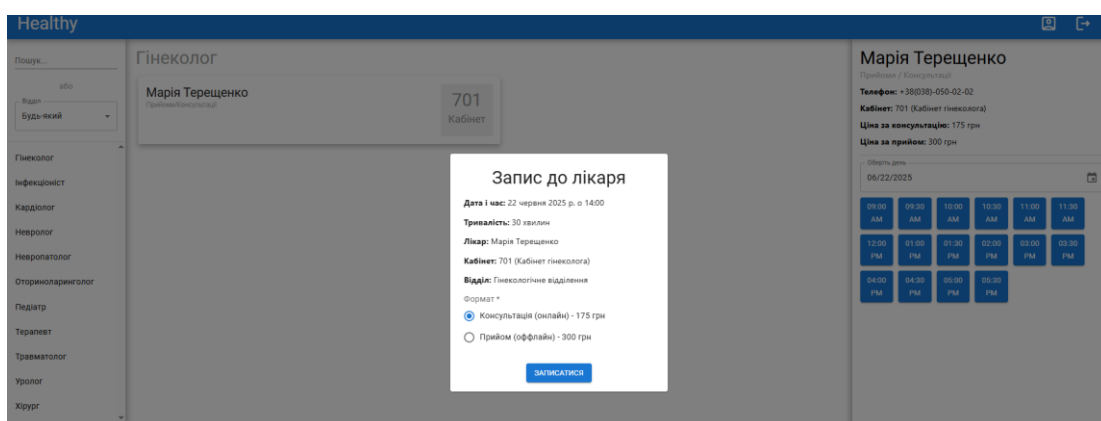


Рис. 3.11. Запит на підтвердження запису

Домашня сторінка лікаря ідентична до клієнтової, але без можливості запису до лікаря (Рис. 3.12). В його кабінеті знаходиться форма для встановлення його графіку (Рис. 3.13). В ній вибираються доступні дні для записів, а також доступний час для них. Справа від неї лікар може знайти ціни на записи до себе, та їх тривалість, та при бажанні, змінити їх (Рис. 3.14).

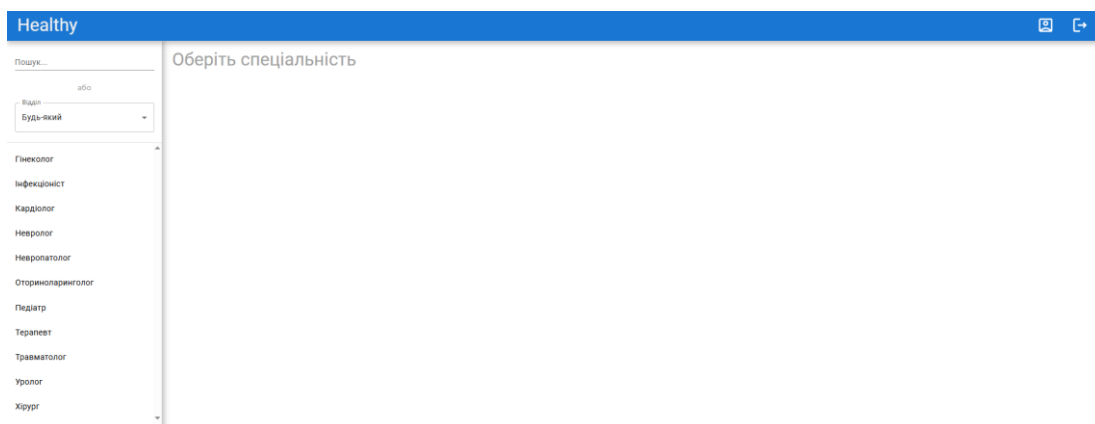


Рис. 3.12. Домашня сторінка лікаря



Рис. 3.13. Сторінка кабінету лікаря

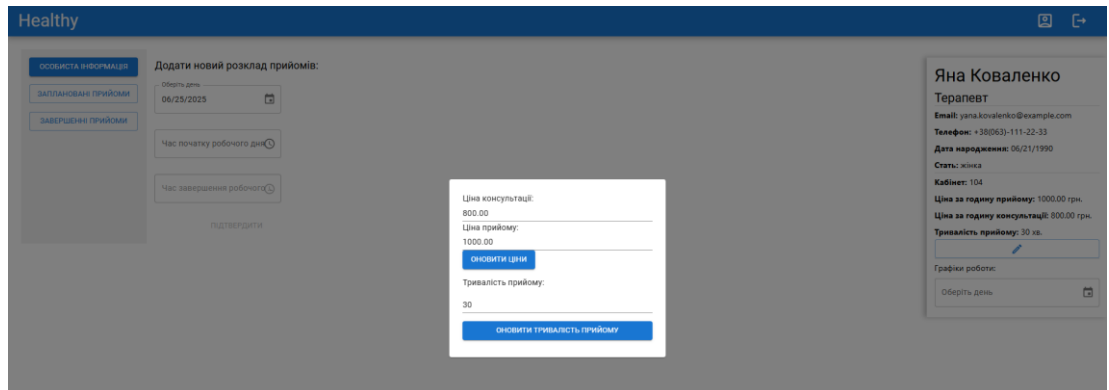


Рис. 3.14. Викликана форма змін цін на записи або їх тривалість

Далі, якщо зайти у обліковий запис адміністратора, та пройти через ідентичну до лікарської або клієнтської домашньої сторінки в його кабінеті, то можна знайти меню управління лікарями. Першими в меню йдуть відділи лікарні, які можна додати через форму, або ж відредагувати (Рис. 3.15).

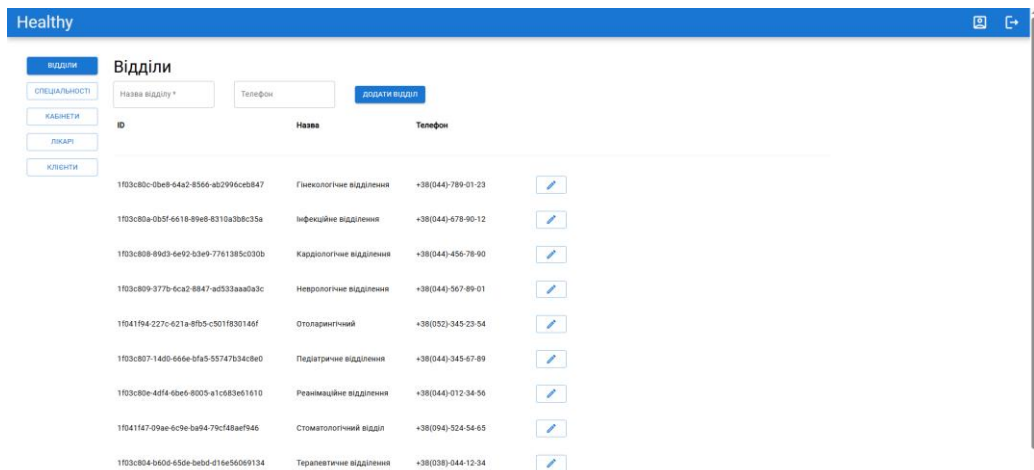


Рис. 3.15. Сторінка управління відділами

Далі йде сторінка управління спеціальностями (Рис. 3.16), які може додавати або корегувати адміністратор. Також спеціальності прив'язуються до певних відділів. Переходячи нижче, відкривається сторінка керування кабінети,

що прив'язуються до певних відділів (Рис. 3.17). Адміністратор також зданих їх коригувати.

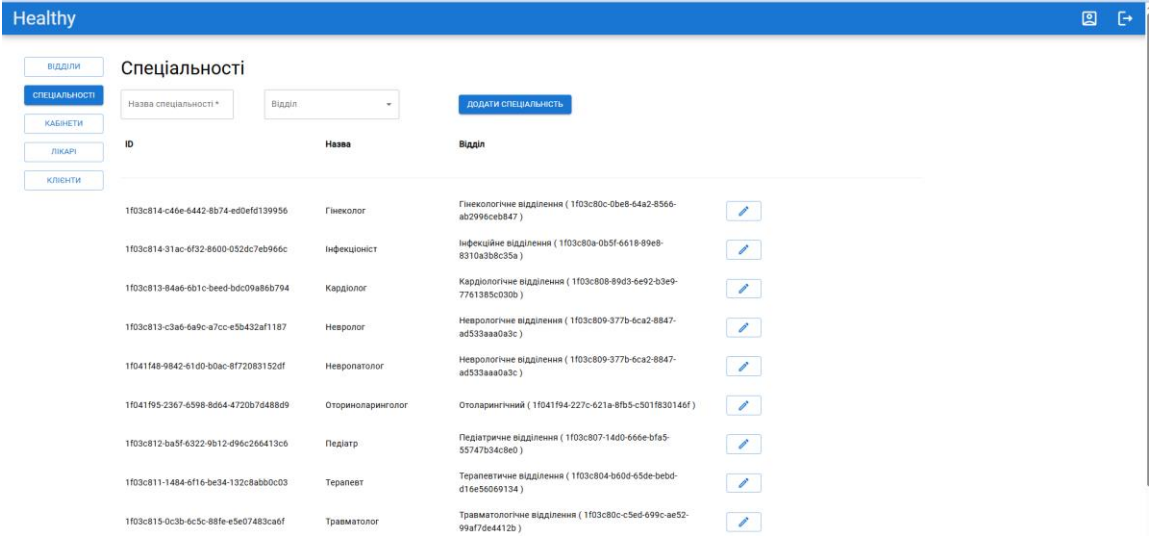


Рис. 3.16. Сторінка керування спеціальностями

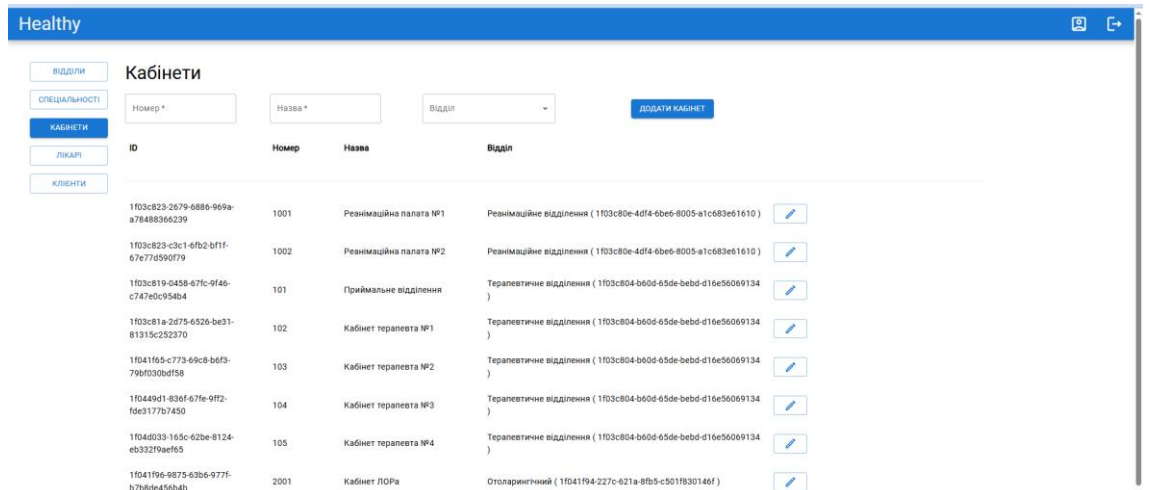


Рис. 3.17. Сторінка керування кабінетами

Спускаючись далі, знаходимо сторінку та панель керування лікарями (Рис. 3.18), де можна додати лікаря, відкоригувати його відомості та додати ціни та тривалість записів до нього. Перейшовши на сторінку реєстрації лікаря (Рис 3.20), адміністратор вбиває його електрону пошту, пароль, ім'я та прізвище, телефон, спеціальність та особистий кабінет. В останній панелі меню (Рис. 3.19) адміністратор має змогу спостерігати дані клієнтів: дати народження, електроні пошти і т. д.

Healthy

ВІДІЛТИ

СПЕЦІАЛЬНОСТІ

КАБІНЕТИ

ЛІКАРІ

КЛІЄНТИ

Лікарі

ДОДАТИ ЛІКАРЯ

ID Доктору	Прізвище	Ім'я	Телефон	Email
Ціна консультації	Ціна прийому	Тривалість	Формати	Кабинет
1f04156b-6802-620c-9eb3-595650643b8b	Бондаренко	Андрій	+38(093)-123-45-67	andriy.bondarenko@clinic.ua
1000.00	500.00	30	Прийоми / Консультації	403
1f041e4a-580b-648a-a315-e315f3baa945	Войтенко	Казимир	+38(098)-765-43-21	kazymyrvoytenko@gmail.com
1000.00	500.00	15	Прийоми / Консультації	601
1f03c9a5-609e-663c-a8be-19e6b78a2d29	Зінченко	Андрій	+38(050)-010-10-10	doc.kovalenko@med.ua
250.00	400.00	25	Прийоми	501

Рис. 3.18. Сторінка керування лікарями

Healthy

ВІДІЛТИ

СПЕЦІАЛЬНОСТІ

КАБІНЕТИ

ЛІКАРІ

КЛІЄНТИ

Клієнти

ID	Фамілія	Ім'я	Телефон	Email	Стать	Дата народження
Адрес проживання	Адрес реєстрації	Паспорт				
1f03c863-1a93-6770-8249-85e70a602ade	Іваненко	Олександр	+38(050)-123-45-67	o.ivanenko@gmail.com	m	3/12/1985
("address":"json")	("address":"json")	000000000				
1f0408a7-432f-6e8e-8cef-bd08572c0d44	Іваненко	Олексій	+38(038)-050-12-34	oleksii.ivanenko85@example.com	m	5/3/1985
("address":"json")	("address":"json")	000000000				
1f03c9b4-7611-6938-8fb7-e90495e3c0d6	Клименко	Іванна	+38(050)-123-45-67	ivanna.klymenko@gmail.com	f	7/18/1994
("address":"json")	("address":"json")	000000000				
1f04100a-2b43-624a-a08d-7d66cbdd22f2	Коваленко	Соломія	+38(073)-111-22-33	solomiia.kovalchuk@gmail.com	f	6/14/1999
("address":"json")	("address":"json")	000000000				

Рис. 3.19. Сторінка спостереження за клієнтами

Healthy

ВІДІЛТИ

СПЕЦІАЛЬНОСТІ

КАБІНЕТИ

ЛІКАРІ

КЛІЄНТИ

Реєстрація лікаря

E-mail *

Пароль *

Підтвердіть пароль *

Телефон

Ім'я *

Прізвище *

Стать *

Дата народження

Спеціальність *

Кабинет *

Пароль має складатися з хоча б одної великої та маленької ліпелених літер, цифри і має бути не коротше 8 символів!

Введені паролі мають співпадати!

Чоловіча Жіноча

ЗАРЕЄСТРУВАТИ ЛІКАРЯ

Рис. 3.20. Сторінка реєстрації лікаря

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3. Тестування роботи системи управління лікарнею

Тестування системи «Healthy» проведено шляхом виконання типової взаємодії користувача, від входу до системи до перегляду доступних лікарів і запису на прийом. Також перевірено функціональність адміністрування та роботи з розкладом. Метою тестування було переконатися в коректності виконання основних функцій, зручності інтерфейсу та відповідності системи функціональним вимогам.

Незареєстрований користувач заходить в додаток на домашню сторінку, де бачить заголовок, поле пошуку лікаря або спеціальності, а також блок фільтрації за відділенням. Зліва виводиться список усіх спеціальностей лікарів, натискання на які дозволяє переглянути доступних фахівців. У верхньому правому куті знаходиться кнопка входу для авторизації в системі. Вибір відділення відображає, наявні в ньому спеціальності (Рис. 3.21).

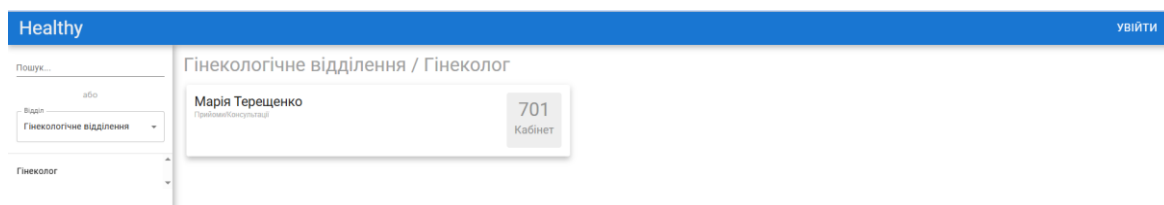


Рис. 3.21. Наявна спеціальність у відділу

Далі користувач, натискає на «Увійти» в панелі навігації та вибирає створення нового облікового запису, та вводить свої дані (Рис. 3.22). Після натискання реєстраційної кнопки, дані облікового запису зберігаються в MySQL, що можна перевірити в phpMyAdmin (Рис. 3.23). У разі однакової електронної пошти, з'явиться помилка. А також під час введення неправильних даних при вході в систему (Рис. 3.24).

Healthy

УВІЙТИ

Реєстрація

E-mail *

iryna.kozlovska@gmail.com

Пароль *

.....

Підтвердіть пароль *

.....

Телефон

+38(067)-444-55-66

Ім'я *

Ірина

Прізвище *

Козловська

Стать *

☐ Чоловіча
 ☒ Жіноча

Дата народження

09/27/1998

ЗАРЕЄСТРУВАТИСЯ

Рис. 3.22. Заповнена форма реєстрації клієнта

Edit Copy Delete 0x1f04d9eddfb269ba98f04f83970625e4	iryna.kozlovska@gmail.com	[ROLE_CLIENT]	S2y\$138kKB7cWoNFX1YnEpB1mxDF.VL4iokqv17mj/3YQ5.Csl...	+38(067)-444-55-66	Ірина	Козловська
--	---------------------------	---------------	--	--------------------	-------	------------

Рис. 3.23. Дані зареєстрованого користувача в базі даних

Невірні дані для входу!

Вхід

E-mail *

example3@gmail

Пароль *

.....

УВІЙТИ

або

ЗАРЕЄСТРУВАТИСЯ

Рис. 3.24. Помилка входу через невірні дані

Далі у разі спроби записатися лікарем або адміністратором, також виникне помилка (Рис. 3. 25). У кабінеті лікаря, можна додати вільний день, тільки від нинішнього дня (Рис. 3.26). Також буде перевірка на несумісний час для записів (Рис. 3.27).

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

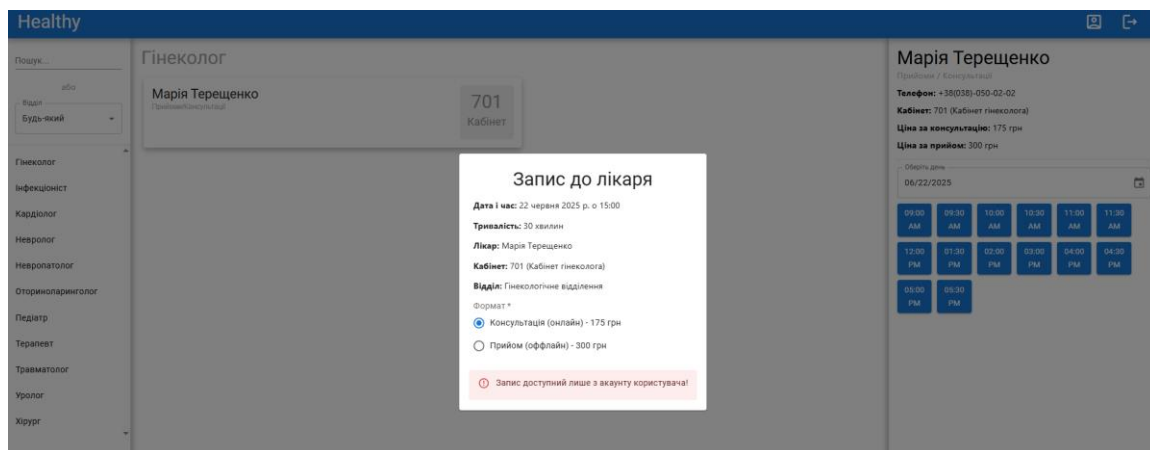


Рис. 3.25. Помилка при спробі запису лікарем або адміністратором

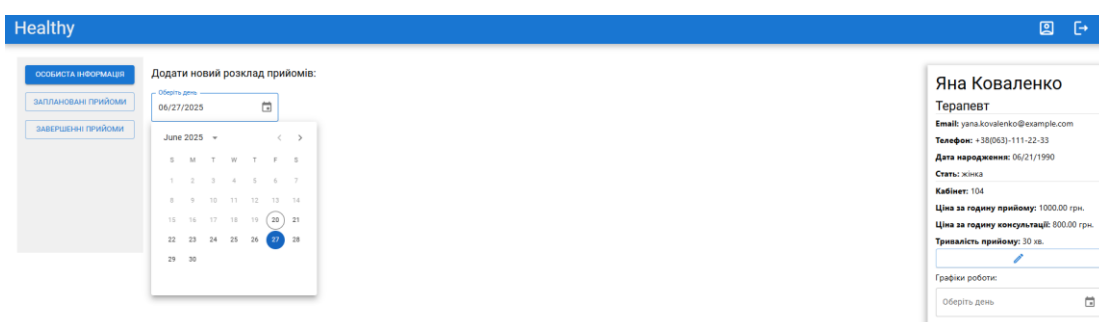


Рис. 3.26. Доступні дні для додання лікарем

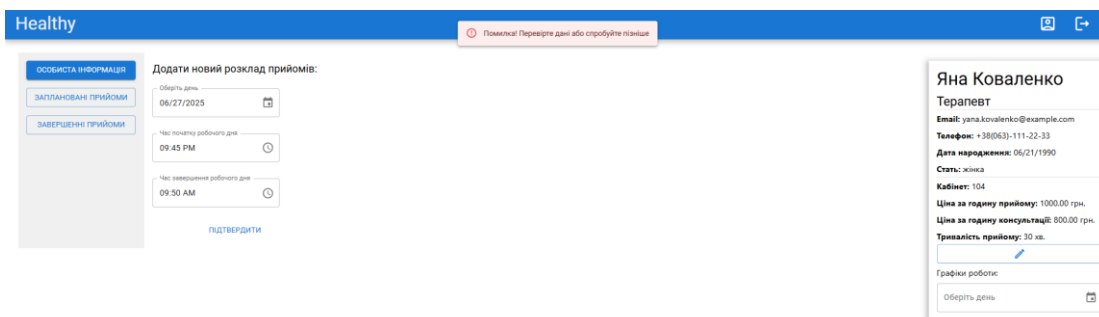


Рис. 3. 27. Помилка при несумістному часі

В кабінеті облікового запису адміністратора, окрім того що він може додавати, та й ще відредагувати дані відділів (Рис. 3.28) або видалити їх. Аналогічні дії він може провести зі спеціальностями, кабінетами, та лікарями.

Висновки до третього розділу

У третьому розділі було розглянуто інтерфейс користувача та порядок роботи із системою «Healthy». Детально описано логіку взаємодії між пацієнтом, лікарем та адміністратором, враховуючи особливості кожної ролі в системі. Наведено приклади екранів: домашньої сторінки, кабінетів користувачів, сторінки реєстрації та редагування профілю.

Були продемонстровані ключові функції: авторизація, реєстрація, перегляд лікарів, запис на прийом, керування розкладом, створення нового лікаря, адміністрування відділень, спеціальностей і кабінетів. Візуалізація інтерфейсу дала змогу оцінити зручність і доступність функцій системи для кінцевих користувачів.

Також описано процес тестування системи: перевірено коректність реєстрації, обробку помилок, валідацію введених даних та відповідність дій очікуваному результату. Особливу увагу приділено поведінці системи в межах типових сценаріїв: запису на прийом, фільтрації лікарів, редагуванню профілю, а також роботі адміністратора зі структурними елементами лікарні.

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		56

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було спроектовано та реалізовано вебзастосунок для керування лікарнею, що автоматизує роботу лікарів, адміністрації та пацієнтів. Було проведено аналіз предметної області, визначено основні вимоги до системи та розроблено її архітектуру з використанням сучасного стеку технологій.

У рамках проєктування реалізовано об'єктно-орієнтовану модель даних, спроектовано реляційну базу даних на основі MySQL, а також створено серверну частину з використанням фреймворку Symfony. Для зручного адміністрування бази даних використовувався інтерфейс phpMyAdmin, що дозволяє переглядати, редагувати та керувати даними безпосередньо в браузері. Інтерфейс користувача розроблено з використанням React, що забезпечує інтерактивність і зручність роботи зі сторінками без перезавантаження.

В процесі розробки реалізовано ключовий функціонал: реєстрацію пацієнтів і лікарів, запис на прийом, ведення розкладу лікаря, фільтрацію візитів. Особливу увагу приділено безпеці доступу та автентифікації на основі ролей користувачів.

Система має модульну структуру, що дозволяє масштабувати її під потреби різних медичних закладів. Використання Docker забезпечує просте розгортання у різних середовищах. Розроблений продукт покращує якість обслуговування пацієнтів, оптимізує робочі процеси персоналу та надає ефективні засоби управління медичною установою.

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Symfony Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://symfony.com/doc/current/index.html>
2. Doctrine ORM [Електронний ресурс] – Режим доступу до ресурсу: <https://www.doctrine-project.org/projects/orm.html>
3. MySQL Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/>
4. React Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://reactjs.org/docs/getting-started.html>
5. Docker Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.docker.com/>
6. PHP Manual [Електронний ресурс] – Режим доступу до ресурсу: <https://www.php.net/manual/en/index.php>
7. Composer [Електронний ресурс] – Режим доступу до ресурсу: <https://getcomposer.org/>
8. API Platform [Електронний ресурс] – Режим доступу до ресурсу: <https://api-platform.com/docs/>
9. JWT Authentication Bundle [Електронний ресурс] – Режим доступу до ресурсу: <https://github.com/lexik/LexikJWTAuthenticationBundle>
10. Vite [Електронний ресурс] – Режим доступу до ресурсу: <https://vitejs.dev/>
11. Node.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nodejs.org/en/docs/>
12. npm Docs [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.npmjs.com/>
13. Git Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://git-scm.com/doc>
14. Postman [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postman.com/>

15. Bootstrap [Електронний ресурс] – Режим доступу до ресурсу:
<https://getbootstrap.com/docs/>

16. JWT.io [Електронний ресурс] – Режим доступу до ресурсу:
<https://jwt.io/introduction>

17. Swagger [Електронний ресурс] – Режим доступу до ресурсу:
<https://swagger.io/docs/>

18. YAML Language [Електронний ресурс] – Режим доступу до ресурсу:
<https://yaml.org/>

19. HTML MDN [Електронний ресурс] – Режим доступу до ресурсу:
<https://developer.mozilla.org/en-US/docs/Web/HTML>

20. CSS MDN [Електронний ресурс] – Режим доступу до ресурсу:
<https://developer.mozilla.org/en-US/docs/Web/CSS>

21. JavaScript MDN [Електронний ресурс] – Режим доступу до ресурсу:
<https://developer.mozilla.org/en-US/docs/Web/JavaScript>

22. SymfonyCasts [Електронний ресурс] – Режим доступу до ресурсу:
<https://symfonycasts.com/>

23. API REST Guide [Електронний ресурс] – Режим доступу до ресурсу:
<https://restfulapi.net/>

24. PHP The Right Way [Електронний ресурс] – Режим доступу до ресурсу:
<https://phptherightway.com/>

25. UML Diagrams [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.uml-diagrams.org/>

26. Figma [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.figma.com/>

27. Linux Command [Електронний ресурс] – Режим доступу до ресурсу:
<https://linuxcommand.org/>

28. NGINX Docs [Електронний ресурс] – Режим доступу до ресурсу:
<https://nginx.org/en/docs/>

29. Axios [Електронний ресурс] – Режим доступу до ресурсу:
<https://axios-http.com/docs/intro>

					ІПЗ.КР.Б-121-25-ІІЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

30. Laravel vs Symfony [Електронний ресурс] – Режим доступу до ресурсу:
<https://kinsta.com/blog/laravel-vs-symfony/>

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

Створення візиту

```
<?php

namespace App\Action;

use App\Entity\DoctorInfo;
use App\Entity\User;
use App\Entity\Visit;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Component\HttpFoundation\Request;
use
Symfony\Component\HttpKernel\Exception\UnprocessableEntityHttpException;
use Symfony\Component\Security\Core\Security;

class CreateVisitAction

    /**
     * @var EntityManagerInterface
     */
    private EntityManagerInterface $entityManager;

    /**
     * @var Security
     */
    private Security $security;

    /**
     * @param EntityManagerInterface $entityManager
     * @param Security $security
     */

    public function __construct(EntityManagerInterface
$entityManager, Security $security)
```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

{
    $this->entityManager = $entityManager;
    $this->security = $security;
}

/**
 * @param Request $request
 * @param Visit $data
 * @return Visit
 */
public function __invoke(Request $request, Visit $data): Visit
{
    $doctorId = $data->getDoctor();

    $clientEmail = $this->security->getUser()-
>getUserIdentifier();

    if (!$clientEmail) {
        throw new UnprocessableEntityHttpException();
    }

    /** @var User $client */
    $client = $this->entityManager->getRepository(User::class)
        ->findOneBy([
            'email' => $clientEmail
        ]);

    /** @var User $doctor */
    $doctor = $this->entityManager->getRepository(User::class)
        ->findOneBy([
            'id' => $doctorId
        ]);

    if (!$doctor) {
        throw new UnprocessableEntityHttpException();
    }

    /** @var DoctorInfo $doctorInfo */
    $doctorInfo = $this->entityManager-
>getRepository(DoctorInfo::class)

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        ->findOneBy([
            'id' => $doctor->getDoctorInfo()->getId()
        ]);
    if (!$doctorInfo) {
        throw new UnprocessableEntityHttpException();
    }

    $formats = $doctorInfo->getFormats();

    if ((!$formats["online"] && !$data->getRoom()) ||
        (!$formats["offline"] && $data->getRoom())) {
        throw new UnprocessableEntityHttpException();
    }

    $price = $data->getRoom() ? $doctorInfo->getVisitPrice() :
        $doctorInfo->getConsultationPrice();

    if (!$price || !$doctorInfo->getDuration()) {
        throw new UnprocessableEntityHttpException();
    }

    $conflictVisits = $this->entityManager-
        >createQueryBuilder()
        ->select('visit')
        ->from(Visit::class, 'visit')
        ->andWhere($this->entityManager-
            >getExpressionBuilder()->gt(
                'visit.date + (visit.duration * 60)',
                ':min'
            ))
        ->andWhere($this->entityManager-
            >getExpressionBuilder()->lt(
                'visit.date',
                ':max'
            ))
        ->andWhere('visit.doctor = :doctor')
        ->setParameter('min', $data->getDate())
        ->setParameter('max', $data->getDate() + ($doctorInfo-
            >getDuration() * 60))

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        ->setParameter('doctor', $data->getDoctor()->getId()-
>toBinary())
        ->getQuery()
        ->getArrayResult();

    if (!empty($conflictVisits)) {
        throw new UnprocessableEntityHttpException();
    }
    $visit = new Visit();
    $visit
        ->setDate($data->getDate())
        ->setDuration($doctorInfo->getDuration())
        ->setPrice($doctorInfo->getVisitPrice() / (60 /
$doctorInfo->getDuration()))
        ->setClient($client)
        ->setDoctor($doctor)
        ->setRoom($data->getRoom());
    return $visit;
}
}

```

Додаток Б

Оновлення часу прийому

```

<?php
namespace App\Action;
use App\Entity\ReceptionTime;
use App\Entity\User;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
use
Symfony\Component\HttpKernel\Exception\UnprocessableEntityHttpExce
ption;
use
Symfony\Component\Security\Core\Exception\AccessDeniedException;
use Symfony\Component\Security\Core\Security;

```

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

class UpdateDoctorReceptionTimesAction
{
    /**
     * @var EntityManagerInterface
     */
    private EntityManagerInterface $entityManager;
    /**
     * @var Security
     */
    private Security $security;
    /**
     * @param EntityManagerInterface $entityManager
     * @param Security $security
     */
    public function __construct(EntityManagerInterface
    $entityManager, Security $security)
    {
        $this->entityManager = $entityManager;
        $this->security = $security;
    }
    /**
     * @param string $id
     * @param ReceptionTime $data
     * @return ReceptionTime
     */
    public function __invoke(string $id, ReceptionTime $data):
    ReceptionTime
    {
        /** @var User $doctor */
        $doctor = $this->security->getUser();
        if (!in_array("ROLE_DOCTOR", $doctor->getRoles())
            || $doctor->getId() !== $data->getDoctor()->getId()) {
            throw new AccessDeniedException();
        }
    }
}

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    /** @var ReceptionTime $receptionTime */
    $receptionTime = $this->entityManager-
>getRepository(ReceptionTime::class)->findOneBy(
        [
            "id" => $id
        ]
    );

    if (!$receptionTime) {
        throw new NotFoundException('Can not find
reception time with id ' . $data->getId());
    }

    if (!$data->getStart() && $data->getEnd()) {
        throw new UnprocessableEntityHttpException();
    }

    $receptionTime
        ->setStart($data->getStart())
        ->setEnd($data->getEnd());

    return $receptionTime;
}
}

```

Додаток В

Отримання розкладу візитів

```

<?php
namespace App\Action;
use App\Entity\DoctorInfo;
use App\Entity\ReceptionTime;
use App\Entity\User;
use App\Entity\Visit;
use App\Services\FreeVisitsService;
use Doctrine\Common\Collections\ArrayCollection;
use Doctrine\Common\Collections\Collection;

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

use Doctrine\ORM\EntityManagerInterface;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpKernel\Exception\HttpException;
use
Symfony\Component\HttpKernel\Exception\UnprocessableEntityHttpException;

class GetDoctorScheduleAction
{
    /**
     * @var EntityManagerInterface
     */
    private EntityManagerInterface $entityManager;
    /**
     * @param EntityManagerInterface $entityManager
     */
    public function __construct(EntityManagerInterface
$entityManager)
    {
        $this->entityManager = $entityManager;
    }

    /**
     * @param Request $request
     * @param string $id
     * @param string $day
     * @param FreeVisitsService $freeVisitsService
     * @return array
     */
    public function __invoke(Request $request, string $id, string
$day, FreeVisitsService $freeVisitsService): array
    {
        /** @var User $doctor */

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        $doctor = $this->entityManager->getRepository(User::class)
            ->findOneBy(
                ['id' => $id]
            );

        if (!$doctor) {
            throw new UnprocessableEntityHttpException("Can`t find
doctor with id " . $id);
        }

        if (!in_array(User::ROLE_DOCTOR, $doctor->getRoles())) {
            throw new UnprocessableEntityHttpException("Given user
is not a doctor");
        }

        /** @var DoctorInfo $doctorInfo */
        $doctorInfo = $this->entityManager->getRepository(DoctorInfo::class)
            ->findOneBy([
                'user' => $doctor->getId()
            ]);

        if (!$doctorInfo) {
            throw new UnprocessableEntityHttpException("No
doctorInfo found for given user");
        }

        $timezoneOffset = $request->headers->get('Timezone-
Offset') * 60;

        $day = floor(($day - $timezoneOffset) / 86400) * 86400;

        $receptionTimes = $doctor
            ->getReceptionTimes()

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        ->filter(fn($item) => $item->getStart() < $day + 86400
        && $item->getEnd() > $day);

        $freeTimes = [];

        foreach ($receptionTimes as $item) {

            $freeTimes = array_unique(array_merge($freeTimes,
            $freeVisitsService->getFreeVisits($doctorInfo, $item,
            $timezoneOffset));

        }

        sort($freeTimes);

        return array_filter($freeTimes, fn($item) => $item >
        time());

    }

}

```

Додаток Г

Створення користувача

```

<?php

namespace App\Action;

use App\Entity\ClientInfo;
use App\Entity\User;
use Doctrine\ORM\EntityManager;
use Doctrine\ORM\EntityManagerInterface;

use
Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterfac
e;

class CreateUserAction

{

    /**
     * @var UserPasswordHasherInterface
     */

    private UserPasswordHasherInterface $passwordHasher;

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

/**
 * @var EntityManagerInterface
 */
private EntityManagerInterface $entityManager;

/**
 * @param UserPasswordHasherInterface $passwordHasher
 * @param EntityManagerInterface $entityManager
 */
public function __construct(UserPasswordHasherInterface
$passwordHasher, EntityManagerInterface $entityManager)
{
    $this->passwordHasher = $passwordHasher;
    $this->entityManager = $entityManager;
}

/**
 * @param User $data
 * @return User
 */
public function __invoke(User $data): User
{
    if (!$data->getPlainPassword()) {
        return $data;
    }

    $data = $this->hashUserPassword($data);

    if (empty($data->getRoles())) {
        $data->setRoles([User::ROLE_CLIENT]);
    }

    if (empty($data->getClientInfo())) {
        $clientInfo = new ClientInfo();

```

```

        $clientInfo->setUser($data)->setFamily(null)-
>setIsParent(false);

        $this->entityManager->persist($clientInfo);

    }

    return $data;

}

/**
 * @param User $data
 * @return User
 */
public function hashUserPassword(User $data): User
{
    $hashedPassword = $this->passwordHasher->hashPassword(
        $data,
        $data->getPlainPassword()
    );
    $data->setPassword($hashedPassword)
        ->eraseCredentials();
    return $data;
}
}

```

Додаток Г

Реалізація особистого кабінета пацієнта

```

import { Helmet } from "react-helmet-async";
import ClientInfo from "../ClientInfo";
import React, { useEffect, useState } from "react";
import jwt_decode from "jwt-decode";
import axios from "axios";
import userAuthenticationConfig from
"../../utils/userAuthenticationConfig";
import { responseStatus } from "../../utils/consts";

```

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import Notification from
"../../elements/notification/Notification";

import { Button, CircularProgress, Divider } from "@mui/material";
import ClientVisitsList from "../ClientVisitsList";

const CabinetContainer = () => {
  const { id } = jwt_decode(localStorage.getItem("token"));

  const [loading, setLoading] = useState(false);
  const [userInfo, setUserInfo] = useState(null);
  const [newInfo, setNewInfo] = useState(null);
  const [visits, setVisits] = useState(null);
  const [filterData, setFilterData] = useState({ type: "visits",
page: 1 });
  const [visitsLoading, setVisitsLoading] = useState(false);
  const [payId, setPayId] = useState(null);
  const [cancelId, setCancelId] = useState(null);
  const [notification, setNotification] = useState({
    visible: false,
    type: "",
    message: ""
  });

  const [paginationInfo, setPaginationInfo] = useState({
    totalItems: 0,
    totalPageCount: 0,
    itemsPerPage: 10,
    page: 1
  });

  const fetchUserInfo = () => {
    setLoading(true);

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        axios.get("/api/users/" + id,
userAuthenticationConfig()).then(response => {

        if (response.status === responseStatus.HTTP_OK &&
response.data) {

            setUserInfo(response.data);

            fetchUserVisits();

        }

    }).catch(error => {

        console.log("error fetching user data");

        setNotification({

            ...notification,

            visible: true,

            type: "error",

            message: "Помилка при завантаженні даних!"

        });

    }).finally(() => {setLoading(false);});

};

const fetchUserVisits = () => {

    setVisitsLoading(true);

    axios.get("/api/users/" + id + "/" + filterData.type +
`?itemsPerPage=${paginationInfo.itemsPerPage}&page=${filterData.pa
ge}`, userAuthenticationConfig()).then(response => {

        if (response.status === responseStatus.HTTP_OK &&
response.data["hydra:member"]) {

            setVisits(response.data["hydra:member"][1]);

            setPaginationInfo({

                ...paginationInfo,

                totalItems: response.data["hydra:member"][0],

                totalPageCount:
Math.ceil(response.data["hydra:member"][0] /
paginationInfo.itemsPerPage)

            });

        }

    })
}

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }).catch(error => {
        console.log("error fetching user visit");
        setNotification({
            ...notification,
            visible: true,
            type: "error",
            message: "Помилка при завантаженні візитів!"
        });
    }).finally(() => {
        setVisitsLoading(false);
    });
};

const patchUserInfo = () => {
    if (!newInfo) {
        return;
    }
    setLoading(true);

    axios.patch("/api/users/" + id, newInfo,
        userAuthenticationConfig()).then(response => {
        if (response.status === responseStatus.HTTP_OK &&
            response.data) {
            setUserInfo(response.data);
            setNotification({
                ...notification,
                visible: true,
                type: "success",
                message: "Дані успішно оновлено!"
            });
        }
    }).catch(error => {
        console.log("error patching user data");
        setNotification({
            ...notification,

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        visible: true,
        type: "error",
        message: "Помилка при зміні даних!"
    });
}).finally(() => {setLoading(false);});
};

const payVisit = () => {
    if (!payId) {
        return;
    }

    axios.patch("/api/visits/" + payId, { isPaid: true },
    userAuthenticationConfig()).then(response => {
        fetchUserVisits();
        setNotification({
            ...notification,
            visible: true,
            type: "success",
            message: "Консультацію успішно оплачено!"
        });
    }).catch(error => {
        console.log("error paying visit");
        setNotification({
            ...notification,
            visible: true,
            type: "error",
            message: "Помилка при оплаті консультації!"
        });
    }).finally(() => {setPayId(null);});
};

```

					ІПЗ.КР.Б-121-25-ІПЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const cancelVisit = () => {
  if (!cancelId) {
    return;
  }

  axios.delete("/api/visits/" + cancelId,
    userAuthenticationConfig()).then(response => {
    fetchUserVisits();
    setNotification({
      ...notification,
      visible: true,
      type: "success",
      message: "Запис скасовано!"
    });
  }).catch(error => {
    console.log(error);
    console.log("error paying visit");
    setNotification({
      ...notification,
      visible: true,
      type: "error",
      message: "Помилка при скасування запису!"
    });
  }).finally(() => {setCancelId(null);});
};

useEffect(() => {
  fetchUserInfo();
}, []);

useEffect(() => {
  fetchUserVisits();

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }, [filterData]);

    useEffect(() => {
        patchUserInfo();
    }, [newInfo]);

    useEffect(() => {
        payVisit();
    }, [payId]);

    useEffect(() => {
        cancelVisit();
    }, [cancelId]);

    return (
        <>
            <Helmet>
                <title>
                    Кабінет пацієнта
                </title>
            </Helmet>
            {notification.visible &&
                <Notification
                    notification={notification}
                    setNotification={setNotification}
                />
            }
            {loading &&
                <div style={{ marginTop: "15px", width: "100%", display:
                    "flex", justifyContent: "center" }}>
                    <CircularProgress />
                </div>
            }
        </>
    );
}

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    {!loading && <>
      <ClientInfo
        userInfo={userInfo}
        setEditedInfo={setNewInfo}
        setNotificaiton={setNotification}
        notification={notification}
      />

      <Divider style={{ margin: "20px auto 10px auto" }} />

      <div style={{ width: "max-content", display: "flex", gap:
"5px", margin: "10px auto" }}>
        <Button
          disabled={visitsLoading}
          variant={filterData.type === "visits" ? "outlined" :
"contained"}
          onClick={() => {setFilterData({ type: "visits", page:
1 });}}
        >
          Майбутні записи
        </Button>

        <Button
          disabled={visitsLoading}
          variant={filterData.type === "history" ? "outlined" :
"contained"}
          onClick={() => {setFilterData({ type: "history", page:
1 });}}
        >
          Історія записів
        </Button>
      </div>

      {visitsLoading &&
        <div style={{ marginTop: "15px", width: "100%", display:
"flex", justifyContent: "center" }}>
          <CircularProgress />

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        </div>
    }
    {!visitsLoading && <ClientVisitsList
        visits={visits}
        setPayId={setPayId}
        paginationInfo={paginationInfo}
        filterData={filterData}
        setFilterData={setFilterData}
        setCancelId={setCancelId}
    />}
</>
}
</>
);
};

export default CabinetContainer;

```

					ІІЗ.КР.Б-121-25-ІІЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		