

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ ПІДТРИМКИ
ЕЛЕКТРОННОЇ КАРТКИ ПАЦІЄНТА НА ОСНОВІ БЛОКЧЕЙНУ
WEB APPLICATION DEVELOPMENT FOR SUPPORTING AN
ELECTRONIC HEALT RECORD BASED ON BLOCKCHAIN

Виконав: студент 4 курсу, групи КНТ-118сп
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Шелудько В.А.

(прізвище та ініціали)

Керівник Шитікова О.В.

(прізвище та ініціали)

Рецензент Морщавка С.В.

(прізвище та ініціали)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Шитікова О.В., доцент		
Нормоконтроль	Камінська Ж.К.		

7. Дата видачі завдання “__” _____ року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	1 тиждень	
2	Розробка та затвердження технічного завдання	2 тиждень	Технічне завдання
3	Проектування структури системи	3-4 тижні	Структурна схема
4	Розробка схеми функціонування системи та модулів програми	5-6 тижні	Функціональна схема. Модулі
5	Створення програмного коду системи	7-8 тижні	Текст програми
6	Тестування та відлагодження програми	9 тиждень	Працездатна система
7	Розробка розділів пояснювальної записки	10 тиждень	ПЗ
8	Розробка слайдів презентації	11-12 тижні	Слайди презентації
9	Захист роботи	13 тиждень	

Студент

(підпис) Шелудько В.А.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Шитікова О.В.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 93 с., 25 рис., 15 табл., 4 дод., 16 джерел.

ЕЛЕКТРОННА КАРТКА ПАЦІЄНТА, МЕДИЧНІ ІНФОРМАЦІЙНІ СИСТЕМИ, БЛОКЧЕЙН, ВЕБЗАСТОСУНОК, JAVASCRIPT, REACT-REDUX.

Об'єкт дослідження – вебзастосунок для підтримання роботи електронної картки пацієнта.

Предмет дослідження – методи підтримки електронної картки пацієнта на основі блокчейну.

Мета роботи – програмна реалізація методів для підтримання електронної картки пацієнта на основі блокчейну.

Актуальність та проблематика роботи полягає в тому, що останнім часом існує багато медичних інформаційних систем, що не завжди задовольняють вимоги користувачів систем та їх потреби.

В роботі наведено аналіз та дослідження існуючих медичних інформаційних систем. Наведені приклади існуючих медичних інформаційних систем та їх функціонал в залежності від модулів, що до них підключені і виконують конкретні функції. Розроблено метод для збереження даних картки пацієнта на основі блокчейну.

Особливість данного методу полягає в неможливості підробити записи в електронній картці пацієнта через наявність у кожному записі ідентифікаторів про час та користувача, що зробив зміни.

Розроблено вебзастосунок, який призначений для вирішення завдання про підтримання електронної картки пацієнта, а саме захист даних від підробки у зв'язку з унікальними ключами в кожному записі.

ABSTRACT

Explanatory note to the final qualifying work of the bachelor: 93 pages, 25 figures, 15 tables, 4 appendixes, 16 sources.

ELECTRONIC PATIENT CARD, MEDICAL INFORMATION SYSTEMS, BLOCKCHAIN, WEB APPLICATION, JAVASCRIPT, REACT-REDUX.

The object of study is web application to support the work of the patient's electronic card.

The subject of study is methods of maintaining an electronic patient card based on a blockchain.

The purpose of the work is software implementation of methods for maintaining an electronic patient card based on blockchain.

The relevance and problems of the work is that recently there are many medical information systems that do not always meet the requirements of system users and their needs.

The work presents an analysis and research of existing medical information systems. Examples of existing medical information systems and their functionality depending on the modules connected to them and perform specific functions are given. A method for saving patient card data based on a blockchain has been developed.

The peculiarity of this method is the impossibility of forging records in the electronic card of the patient due to the presence in each record of identifiers about the time and the user who made the changes.

A web application has been developed to solve the problem of maintaining an electronic patient card, namely to protect data from forgery due to the unique keys in each record.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	9
Вступ.....	10
1 Аналіз предметної області.....	12
1.1 Основні поняття	12
1.2 Огляд існуючих методів	14
1.3 Порівняльний аналіз існуючих систем електронної медицини	16
1.4 Постанова завдання.....	17
1.5 Висновок	18
2 Вибір та обґрунтування структури системи, яка проєктується.....	19
2.1 Вибір інструментарію розробки програмного продукту	19
2.2 Структурна схема програмного продукту	26
2.3 Висновок	33
3 Основні рішення щодо реалізації компонентів системи.....	34
3.1 Метод реалізації алгоритму блокчейну для захисту даних електронної картки пацієнта.....	34
3.2 Проєктування інтерфейсу.....	36
3.3 Висновок	43
4 Керівництво програміста.....	44
4.1 Призначення й умови використання	44
4.2 Характеристики програмного продукту	44
4.3 Звертання до програмного продукту (файли проєкту)	45
4.4 Вхідні і вихідні дані	45
4.5 Повідомлення	45
4.6 Висновок	46
5 Керівництво оператора	47
5.1 Призначення й умови виконання програмного продукту.....	47
5.2 Виконання програмного продукту	47
5.3 Повідомлення оператору	50

5.4 Висновок	50
Висновки	51
Перелік джерел посилення	53
Додаток А Технічне завдання	55
Додаток Б Опис програмного продукту	61
Додаток В Текст програми	70
Додаток Д Слайди презентації.....	88

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
IDE	– Integrated Development Environment;
БД	– база даних;
ЕМЗ	– електронний медичний запис;
МІС	– медичні інформаційні системи;
НМП	– надавання медичних послуг;
ПМД	– первинна медична допомога;
ПП	– програмний продукт;
СМД	– спеціалізована медична допомога;
ТЗ	– технічне завдання;
ЦБД	– центральна база даних;

ВСТУП

За минулі роки в Україні було здійснено реформу медицини, яка полягала у тому, що населення повинне було перейти від звичного режиму запису до терапевта для подальшого лікування до прикріплення осіб до сімейних лікарів для ведення їх лікування на основі декларації, що підписується обома сторонами. Разом із цим, лікарі почали переходити на електронну систему запису пацієнтів на прийом та призначення лікування, а також оперативне отримання аналізів пацієнтів через мережу Інтернет та вебзастосунки.

Система спочатку тестувалася в Дніпропетровській області на протязі декількох років і показала непогані результати порівняно зі старою системою лікування хворих. Після тестування влада прийняла рішення впровадити систему в інші області України, щоб через декілька років взагалі позбутися старої системи.

Однак на жаль не всі лікарі та пацієнти задоволені нині існуючими вебзастосунками, для деяких пацієнтів занадто складний спосіб реєстрації на прийом лікаря або немає населеного пункту, в якому вони проживають, для лікарів є деякі нюанси медичних закладів, які відрізняються в залежності від області, де вони розташовані.

Наразі функції, які потрібні для основної роботи вебзастосунку є:

- можливість для пацієнтів легко знайти свого або обрати лікаря, що має вільні місця для підписання декларації;
- можливість записатися на прийом;
- можливість переглянути електронну картку;
- можливість отримати від лікаря рекомендації;
- особистий кабінет лікаря для ведення пацієнтів;
- можливість отримання аналізів пацієнтів;
- зручний доступ до електронних карток пацієнтів, що підписали декларацію.

Система може використовувати електронну пошту Gmail для доставлення повідомлень пацієнтам та лікарям.

Система розроблена за допомогою середовища програмування Visual Studio Code мовою програмування JavaScript. Для зберігання даних було обрано NoSQL базу даних MongoDB.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття

Лікар, лікарка – фахівець із повною вищою медичною освітою, який в установленому законом порядку постійно займається підтримкою або відновленням людського здоров'я, через запобігання (профілактика), розпізнавання (діагностика) та лікування захворювань і травм [1].

Сімейний лікар – лікар, який отримав спеціалізовану багатoproфільну підготовку з надання первинної медико-санітарної допомоги членам родини будь-якого віку та статі [2].

Пацієнт – фізична особа, яка отримує медичну допомогу (профілактичну, діагностичну, лікувальну), або піддається медико-біологічним дослідженням (клінічним випробуванням), або є донором крові чи певного органа при трансплантації тощо [3].

Хвороба або захворювання – патологічний процес, який проявляється порушеннями морфології (анатомічної, гістологічної будови), обміну речовин та функціонування організму (його частин) у людини або тварини [4].

Терапія або лікування – процес, бажаною (але не завжди досяжною) метою якого є полегшення, зняття чи усунення симптомів (компенсація, купірування) і проявів того чи іншого захворювання, патологічного стану чи іншого порушення життєдіяльності, нормалізація порушених процесів життєдіяльності і одужання, відновлення здоров'я [5].

Лікарня або шпиталь – медичний заклад та лікувально-профілактична установа, яка надає стаціонарну кваліфіковану і спеціалізовану медичну допомогу. Існує також термін госпіталь, що наразі в Україні вживається переважно щодо військових лікарень, шпиталів [6].

Амбулаторія – лікувально-профілактичний заклад, який подає основні види медичної допомоги як в самій амбулаторії, так і вдома [7].

Декларація – документ, офіційна заява, де проголошуються основні принципи зовнішньої та внутрішньої політики держави чи програмні положення партій та організацій [8].

Електронна медицина – діяльність з використанням електронних інформаційних ресурсів у сфері охорони здоров'я та забезпечення оперативного доступу медичних працівників та пацієнтів до них[9].

Листок непрацездатності або лікарняний листок – документ, що підтверджує тимчасову непрацездатність громадянина і є підставою для призначення та виплати йому допомоги по тимчасовій непрацездатності [10].

Електронний медичний запис, електронна медична картка – медична картка пацієнта медичного закладу в електронній (без паперовій) формі. Складається і зберігається в автоматизованій інформаційній базі даних медичної установи [11].

Електронна система охорони здоров'я eHealth – українська інформаційно-телекомунікаційна система, що забезпечує автоматизацію ведення обліку медичних послуг та управління медичною інформацією в електронному вигляді. До її складу входять центральна база даних та медичні інформаційні системи, між якими забезпечено автоматичний обмін даними через відкритий програмний інтерфейс[12].

Автоматизація – є одним з напрямів науково-технічного прогресу, який спрямовано на застосування саморегульованих технічних засобів, економіко-математичних методів і систем керування, що звільняють людину від участі у процесах отримання, перетворення, передавання і використання енергії, матеріалів чи інформації, істотно зменшують міру цієї участі чи трудомісткість виконуваних операцій. Разом з терміном автоматичний, використовується поняття автоматизований, що підкреслює відносно великий ступінь участі людини у процесі [13].

Блокчейн – розподілена база даних, що зберігає впорядкований ланцюжок записів (так званих блоків), що постійно довшас. Кожен блок містить часову позначку, хеш попереднього блоку та дані транзакцій, подані

як хеш-дерево. Інформація про транзакції зазвичай надається відкритою, не шифрованою. Захистом від підробки та спотворення слугує включення хешу всього блоку у наступний блок. Тому внесення змін в один з блоків вимагає відповідних змін в усіх блоках після нього, що зазвичай виявляється або дуже складно, або дуже коштовно[14].

1.2 Огляд існуючих систем

Електронні системи почали з'являтися декілька років тому, наразі існує вже більше 40 електронних систем різного призначення, одні використовуються лише як довідники, інші можуть виконувати специфічні функції, деякі з систем навіть зберігають конфіденційність даних користувачів або надають можливість анонімно здавати тести на деякі хвороби.

Існуючі електронні системи дозволяють здійснювати отримання інформації щодо отримання медичної допомоги, наявності лікарів, що мають вільні місця для підписання декларацій, відгуки про лікарів, надають змогу лікарям в таких системах можуть здійснювати доступ до свого електронного кабінету, за допомогою якого можуть призначати лікування пацієнтам, отримувати результати аналізів із лабораторій, до яких було надіслано пацієнтів, вести медичні картки пацієнтів, виписувати електронні лікарняні листи та листи непрацездатності, в деяких системах є можливість отримувати від лікарів рецепти для отримання ліків у спеціалізованих аптеках, а також отримувати знижки на деякі імпортовані препарати.

Усі системи підключені до однієї великої системи eZdorovya, що впроваджена на нормативно-правовому рівні Міністерством охорони здоров'я. Національна служба здоров'я України укладає договори з постачальниками медичних послуг; аналізує та використовує дані з метою прогнозування потреб населення у медичних послугах, розробки програми

медичних гарантій, встановлення тарифів, здійснення моніторингу договорів; здійснює оплати за медичні послуги на умовах договору.

Основні цілі системи eZdorovya:

- здійснення адміністрування та технічну підтримку ЦБД
- забезпечення безперебійної роботи ЦБД
- забезпечення розробки, оновлення та підтримки програмного забезпечення ЦБД
- прийняття рішень про підключення, відключення та зупинення доступу МІС до ЦБД
- вживання заходів до захисту інформації, що міститься у ЦБД

Загальна архітектура системи показана на рис.1.1.

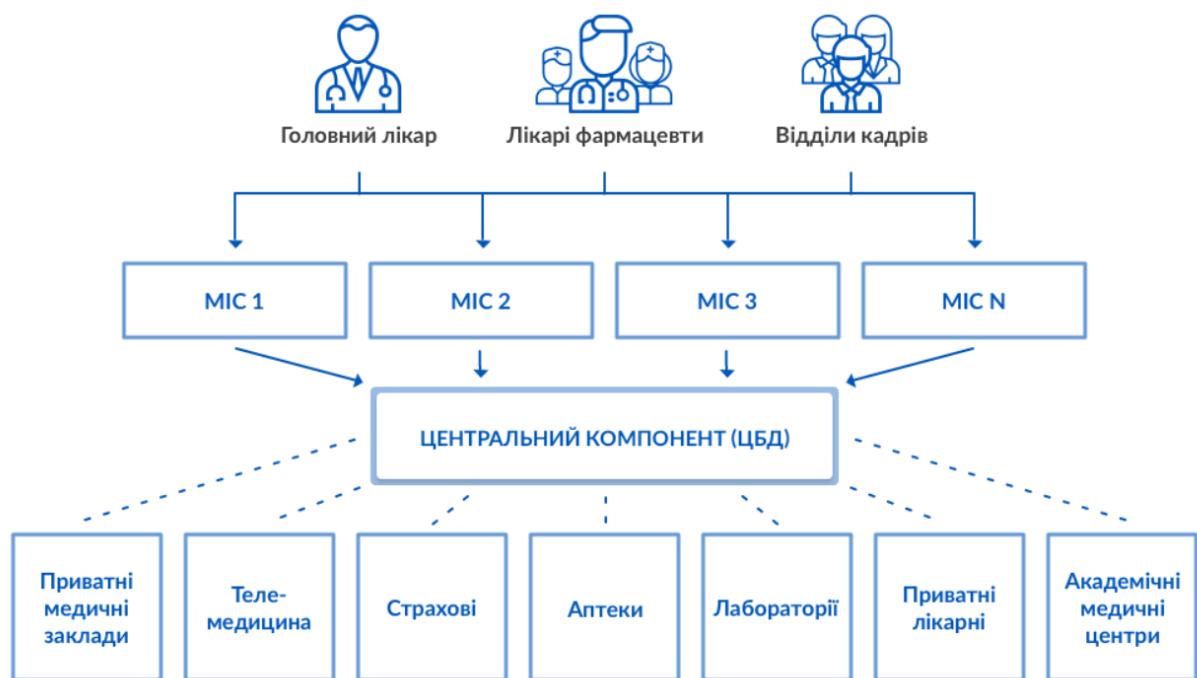


Рисунок 1.1 – Загальна архітектура системи

Отже розглянемо декілька прикладів електронних систем.

1.3 Порівняльний аналіз існуючих систем електронної медицини

Наразі електронна система охорони здоров'я працює за модульною системою. Модулем у даному випадку виступає набір функціоналу необхідний для різних типів користувачів системи. Це означає, що один медичний заклад може використовувати адміністративний модуль однієї МІС, а лікарі – користуватися модулем “Робоче місце лікаря” іншої МІС. Порівняння декількох електронних систем за наявністю в них модулів представлена на табл. 1.1, за приклад було взято системи на основі опитування деяких лікарів та пацієнтів.

Таблиця 1.1 – Порівняльна характеристика електронних систем з підключеними до них модулями

Модулі	Medcore	Поліклінік а без черг	Helsi
Робоче місце лікаря ПМД. Робота з деклараціями	Ні	Так	Так
Робоче місце лікаря ПМД. Електронні медичні записи	Ні	Так	Так
Робоче місце лікаря ПМД. Імунізація	Ні	Так	Так
Робоче місце лікаря ПМД. Виписування електронного рецепту «Доступні ліки»	Ні	Так	Так
Робоче місце лікаря ПМД. Електронні направлення	Ні	Так	Так
Реєстрація користувачів	Ні	Так	Ні
Робоче місце лікаря СМД. Електронні медичні записи	Так	Ні	Так
Робоче місце лікаря СМД. Імунізація	Ні	Ні	Так
Робоче місце лікаря СМД. Діагностичні звіти	Так	Ні	Так
Робоче місце лікаря СМД. Виписування електронних направлень	Так	Ні	Так
Робоче місце лікаря СМД. Оброблення та погашення електронного направлення СМД	Так	Ні	Так
Робоче місце лікаря СМД. ЕМЗ стаціонар: надходження	Ні	Ні	Ні

Продовження таблиці 1.1

Модулі	Medcore	Поліклінік а без черг	Helsi
Робоче місце лікаря СМД. ЕМЗ стаціонар: виписка	Ні	Ні	Так
Робоче місце лікаря СМД. ЕМЗ для неідентифікованих пацієнтів	Ні	Ні	Так
Робота з записами про ідентифікованих пацієнтів	Так	Так	Так
Робота з записами про неідентифікованих пацієнтів	Так	Ні	Так
Приєднання записів неідентифікованого пацієнта до ідентифікованого	Ні	Ні	Ні
Доступ до даних	Ні	Ні	Ні
Медичні висновки про народження	Ні	Ні	Так

На основі табл.1.1 можна сказати, що не всі модулі є підключеними до кожної електронної системи, а також, що в деяких електронних системах можуть працювати виключно лікарі або уповноважені лікарями особи, тобто щоб записатися на прийом до лікаря у відповідній системі необхідно спочатку фізично з'явитися до сімейного лікаря та підписати декларацію. Пацієнти, що користуються деякими системами не можуть отримати доступ до деяких своїх даних без допомоги лікаря або уповноваженої особи, що може викликати дискомфорт.

1.4 Постанова завдання

В даній роботі представлена електронна система для пацієнтів та лікарів для отримання медичної допомоги. Основними завданнями є реєстрація та авторизація лікарів та пацієнтів у вебзастосунку для подальшої роботи. Для різних користувачів повинен бути розроблений свій інтерфейс, що надає функціонал в залежності від ролі користувача. Для захисту даних

від фальсифікації можна використовувати технологію блок-чейн(block-chain), що дозволяє створювати унікальні виписні листи і блокує доступ до ключа унікальності виписки для запобігання фальсифікації даних. Такі записи будуть містити часові маркери, що дозволять відстежити коли та хто робив зміни.

Метою роботи є створення вебзастосунку, що дозволяє користувачам отримувати або створювати виписні медичні документи.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- реалізація процесу реєстрації та авторизації користувачів в системі;
- надати можливість пацієнту обирати лікаря для подальшого отримання виписних документів;
- надати можливість пацієнту переглядати свої документи;
- налаштувати систему для перегляду даних пацієнтів та створення виписних документів;
- реалізувати алгоритм захисту даних про документи пацієнта на основі технології блок-чейн.

1.5 Висновок

В даному розділі було розглянуто предметну область, а саме визначено основні поняття і терміни, що пов'язані з предметною областю і об'єктом розробки та характеристики комплексу завдань для створення медичної інформаційної системи.

Проведено порівняльний аналіз існуючих медичних інформаційних систем, виділено їх переваги і недоліки. Було сформульовано і поставлено завдання до роботи.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Вибір інструментарію розробки програмного продукту

Одним із важливих етапів життєвого циклу інформаційної системи, а саме проєктування ПП, є вибір інструментарію та підключення необхідних технологій до проєкту.

Основна задача ПП – програмна реалізація вебзастосуку для підтримки електронної картки пацієнта на основі блокчейну.

Розроблюваний ПП повинен обов’язково містити наступні модулі:

- зберігання даних;
- вводу-виводу даних;
- обробляючий.

Для проєктування ПП необхідно обрати такі засоби, як мова програмування, середовище розробки(Integrated Development Environment – IDE) та система управління базами даних(СУБД).

2.1.1 Вибір мови програмування

Для вирішення поставленої задачі, якою є розробка вебзастосунку для підтримання електронної картки пацієнта на основі блокчейну необхідно зробити порівняння сучасних мов програмування, щоб на основі них зробити вибір тієї, яка найкраще підходить для реалізації даного завдання. Нижче наведемо опис найбільш популярних мов для розробки вебзастосунків, розглянемо їх переваги та недоліки.

PHP – скриптова мова програмування, що була створена для генерації HTML-сторінок на стороні вебсервера. PHP одна із популярних мов програмування, що підтримується більшою частиною хостинг-провайдерів.

Основними особливостями мови PHP є:

- мова була складена на основі мов C, Perl, тому програмістам, що працювали з даними мовами легко адаптуватися до синтаксису;
- завдяки стандарту відкритого інтерфейсу зв'язку з БД можна підключатися до БД, для яких існує драйвер;
- мова належить до інтерпритованих, тому обробка сценаріїв виконується з великою швидкістю;
- користувачі PHP по усьому світі є свого роду колективною підтримкою, що дозволяє отримати відповіді на найскладніші питання.

Недоліками мови є:

- складність створення користувацьких змінних та занадто довгі імена вбудованих функцій;
- відсутність підтримки Unicode до версії 6.0;
- використання великої кількості оперативної пам'яті для здобування швидкості обробки сценаріїв.

Python – інтерпритована об'єктно-орієнтована мова програмування із строгою динамічною типізацією. Наявність підтримки модулів дозволяє неодноразово використовувати код. Підтримуються об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована парадигми програмування. Основне призначення мови це написання серверних програм та ПП для керування вебзастосунками.

Основними перевагами Python можна зазначити:

- інтерпретатор Python можна використовувати на більшості відомих операційних системах та платформах;
- можливість виконувати найскладніші завдання за допомогою великої кількості модулів для різних задач;
- синтаксис програми легкий для розуміння, тому що було запозичено багато рис з таких мов як C та C++.

Недоліками мови Python є:

- як і інші інтерпритовані мови програмування, має не високу швидкість роботи;
- відсутність статичної типізації даних, що призводить до великої помилок на етапі виконання програми і які не можливо побачити на етапі компіляції;
- неможливість модифікації існуючих класів на відміну від деяких інших мов.

JavaScript – об'єктно-орієнтована динамічна мова програмування. Найчастіша область застосування – створення сценаріїв вебсторінок. Дозволяє взаємодіяти з користувачем, керувати браузером, змінювати структуру та зовнішній вигляд вебсторінки.

Мову JavaScript визначають як прототипну, скриптову мову з динамічною типізацією. Крім цього може використовувати і інші парадигми програмування і деякі архітектурні властивості.

Перевагами JavaScript є:

- є однією з найбільш популярних мов програмування, про що свідчить велика кількість статей на форумах програмування, зокрема: StackOverFlow містить велику кількість запитів від користувачів, що мають велику кількість готових рішень для багатьох завдань;
- дана мова використовується для написання як серверних програм, так і клієнтських вебзастосунків;
- велика кількість готових фреймворків та бібліотек суттєво спрощують процес реалізації багатьох задач та завдань.

Недоліки JavaScript:

- потреба використання готових фреймворків та бібліотек, через обмеженість самої мови;
- неможливість виявити помилки до етапу компіляції та виконання.

З інформації, що зазначено вище, можна зробити висновок, що кожна з мов має свої переваги та недоліки, які будуть впливати на процес розробки. Також можна зазначити, що усі 3 мови є найбільш розповсюдженими та

використовуваними у світі та постійно поповнюють кількість бібліотек та фреймворків для реалізації різноманітних завдань.

Після збору необхідної інформації було створено порівняльну характеристику мов програмування, що представлена на табл. 2.1.

Таблиця 2.1. – Порівняльна характеристика мов програмування вебзастосунків

Критерії порівняння	PHP	Python	JavaScript
Мова створювалася спеціально для веброзробки	+	—	+
Кросплатформеність	+	+	+
Швидка інтерпретація коду	+	—	++
Наявність зрозумілої та повної документації	+	+	+
Простота вивчення та розробки	+	++	—
Виробничість(продуктивність роботи)	++	—	+
Наявність бібліотек та фреймворків з повною документацією	—	+	+
Простота вивчення та розробки	+	++	—
Наявність великої кількості методів обробки даних	+	++	+
Паралельне програмування	+	+	+
Можливість інтеграції та взаємодії з іншими мовами	+	+	++

Як видно з табл. 2.1 найбільші переваги має мова програмування JavaScript, бо незважаючи на складність вивчення вона володіє тими перевагами, що забезпечують зручне написання коду, швидкість його виконання та продуктивність роботи вебзастосунків на її основі. Мова JavaScript має велику кількість фреймворків та бібліотек з готовими рішеннями більшої частини завдань та документацію для цих бібліотек. Дана мова широко поширена, має велику спільноту розробників та документацію, що спрощує розуміння використання мови. Мова JavaScript підтримується на більшості сучасних браузерів. Отже, на основі всього, що було сказано вище, для розробки ПП було обрано мову програмування JavaScript.

2.1.2 Вибір середовища розробки

Для використання обраної мови програмування потрібно обрати відповідне середовище розробки.

Для мови JavaScript існує велика кількість середовищ розробки, такі як: Visual Studio Code, WebStorm, Notepad++, Vim, Sublime Text, SciTe.

Усі з вказаних середовищ розробки дозволяють писати ПП мовою JavaScript. Найбільш популярні IDE Visual Studio Code та WebStorm – середовища розробки з великою кількістю інструментів для написання, компіляції, відстеження виконання та тестування ПП.

WebStorm – інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains. Воно вміщує в себе усі необхідні інструменти для написання ПП мовами JavaScript, HTML та CSS. Має в собі інструменти для автоматизації створення структури проєктів для бібліотек та фреймворків мови JavaScript, зокрема: Node.js, Angular, Vue.js, ReactJS та інших.

Інструментарій WebStorm доволі великий і окрім підтримки роботи з мовою програмування JavaScript та її фреймворками дозволяє виконувати взаємодію з системою контролю версій, базами даних та модулі тестування.

WebStorm доступний для безкоштовного використання як пробної версії лише перші 30 днів з моменту встановлення на операційну систему, після чого необхідно обов'язково придбати ліцензію на використання IDE.

Перевагами WebStorm є:

- вбудований компілятор;
- вбудована система контролю версій з можливістю, що може синхронізуватися з сервісом GitHub;
- велика кількість інструментів для швидкого створення структури проєктів, зокрема: Webpack та Apache.

Недоліки WebStorm:

- доступ до IDE надається на основі купівлі ліцензії;

– необхідність вивчення великої кількості документації для використання більшої частини вбудованого інструментарію, що не завжди зручно для користувачів початківців.

Visual Studio Code – середовище розробки для створення, редагування та зневадження (налагодження програми) сучасних вебзастосунків і сервісів зберігання даних у хмарних сховищах. Visual Studio Code є безкоштовним та розповсюджено на платформах Windows, MacOS та Linux.

Visual Studio Code знаходиться у вільному доступі без обмежень, що робить його однією з найпопулярнішою IDE серед розробників ПП

Редактор Visual Studio Code містить в собі вбудований відлагоджувач, засоби рефакторингу, інструменти роботи з Git, автодоповнення типових конструкцій та контекстні підказки. У порівнянні з WebStorm не має великого інструментарію, але є легшим для розробників початківців та нових користувачів IDE. Крім того при встановленні додаткових модулів та плагінів, дозволяє використовувати нові можливості і функціонал, а також спростити процес написання та редагування коду.

Visual Studio Code доступний для роботи на наступних платформах: Windows, MacOS, Linux.

Переваги Visual Studio Code:

- безкоштовний доступ до можливостей IDE з можливістю налагодження середовища розробки під особисті вподобання та потреби;
- інтегровані системи компіляції коду, контролю версій та робота з сервісом GitHub;
- легкість в розумінні інтерфейсу, через відсутність нагромадженості інструментів розробки

Недоліками Visual Studio Code є:

- невеликий набір інструментарію на початку використання IDE;
- необхідність уміти обирати, підключати та налагоджувати інструменти для конкретних задач в процесі розробки ПП.

В результаті дослідження вищеописаних було складено порівняльну характеристику середовищ розробки, яка наведена у табл. 2.2.

Таблиця 2.2 – Порівняльна характеристика середовищ розробки

Критерії порівняння	VSC	WebStorm
Кросплатформеність	+	+
Доступність	±	++
Наявність автодоповнення	+	+
Простота встановлення	+	+
Підтримка рефакторингу коду	+	+
Можливість налаштування плагінів	—	+
Зручний користувацький інтерфейс	±	++
Підтримка аналізу коду	+	+
Наявність автоматичного закриття дужок, брекєтів	+	+
Наявність інтерактивної консолі	+	+
Швидкість модернізації	+	++
Інтеграція сервісу GitHub	+	+

Роблячи висновок на основі табл. 2.2 можна сказати, що Visual Studio Code має переваги, бо в цьому середовищі розробки можна налаштувати середовище розробки, має вбудований помічник для написання та редагування коду та область навігації для зручного переміщення між файлами проєкту, підтримку IDE здійснюють не тільки розробники, й користувачами, має великий рівень доступності. На основі цих переваг було обрано середовище розробки Visual Studio Code.

2.2 Структурна схема програмного продукту

Розроблюваний вебзастосунок повинен обов'язково містити такі модулі:

- зберігання даних;
- оброблюючий;
- вводу-виводу даних;
- модуль зв'язку всіх елементів ПП.

В якості вхідних даних використовують інформацію представлену користувачами в момент реєстрації та інформацію про виписні документи, що представлені користувачами для створення виписних листів. Для цих даних необхідно обрати відповідне сховище.

Виділяють наступні види сховищ:

- бінарний файл даних;
- текстовий файл;
- реляційна БД;
- не реляційна БД.

MongoDB – документоорієнтована система управління базами даних з відкритим програмним кодом, що не вимагає опису схеми таблиць. MongoDB зберігає інформацію у вигляді JSON-документів згрупованих в «колекції». В колекціях MongoDB можна зберігати відомості будь-якого вигляду, що дозволяє використовувати різні типи даних для зберігання даних. Разом із цим MongoDB підтримує горизонтальне масштабування і дозволяє задавати правила щодо якогось поля чи групи полів.

Переваги MongoDB:

- відсутність схеми, що дозволяє додавати та вилучати поля при розробці ПП;
- легкість горизонтального масштабування;
- багатий функціонал агрегації, можливість «на льоту» переформатовувати документи під потрібний вигляд.

Розглянемо опис сутностей з їх атрибутами у вигляді табл. 2.3 за такими критеріями як тип використовуваного атрибута, його розмір, чи повино бути обов’язково заповнене поле та загальний опис сутності.

Таблиця 2.3 –Сутності та їх атрибути

Сутність	Атрибут	Тип	Розмір	Обов’язкове	Опис сутності
User	Id_user	Лічильник	Довге ціле	Так	Містить інформацію про користувача системи
	Role	Текстовий	Текст	Так	
	Full_name	Текстовий	Текст	Так	
	Date_of_birt h	Текстовий	Текст	Так	
	Login	Текстовий	Текст	Так	
	Password	Текстовий	Текст	Так	
Patient	Id_patient	Числовий	Довге ціле	Так	Містить інформацію про користувача з типом пацієнт
	Address	Текстовий	Текст	Так	
	Number	Текстовий	Текст	Так	
	Blood_type	Текстовий	Текст	Ні	
Doctor	Id_doctor	Числовий	Довге ціле	Так	Містить інформацію про користувача з типом лікар
	Exp_of_wor k	Текстовий	Текст	Ні	
	Address	Текстовий	Текст	Так	
Doctor_Patient	Id_doctor	Числовий	Довге ціле	Так	Містить інформацію про зв’язок лікарів та пацієнтів
	Id_patient	Числовий	Довге ціле	Так	
Block	Id_block	Числовий	Довге ціле	Так	Містить інформацію про блоки блокчейну
	Id_patient	Числовий	Довге ціле	Так	
	Hash	Текстовий	Текст	Так	
	Last_hash	Текстовий	Текст	Так	
	Time_stamp	Текстовий	Текст	Так	
	List	Об’єкт	Текст	Так	

Як видно з табл. 2.3 під час концептуального проєктування БД визначено такі сутності, як User, що містить основну інформацію про користувачів системи, Patient – містить спеціальну інформацію про користувачів з роллю «Patient», Doctor – містить спеціальну інформацію про користувачів з роллю «Doctor», Doctor_Patient – містить інформацію про зв'язок лікарів та пацієнтів, Block – містить інформацію про виписні листи на основі виконання блоків блокчейну.

На рис. 2.1 представлена ER-діаграма БД.

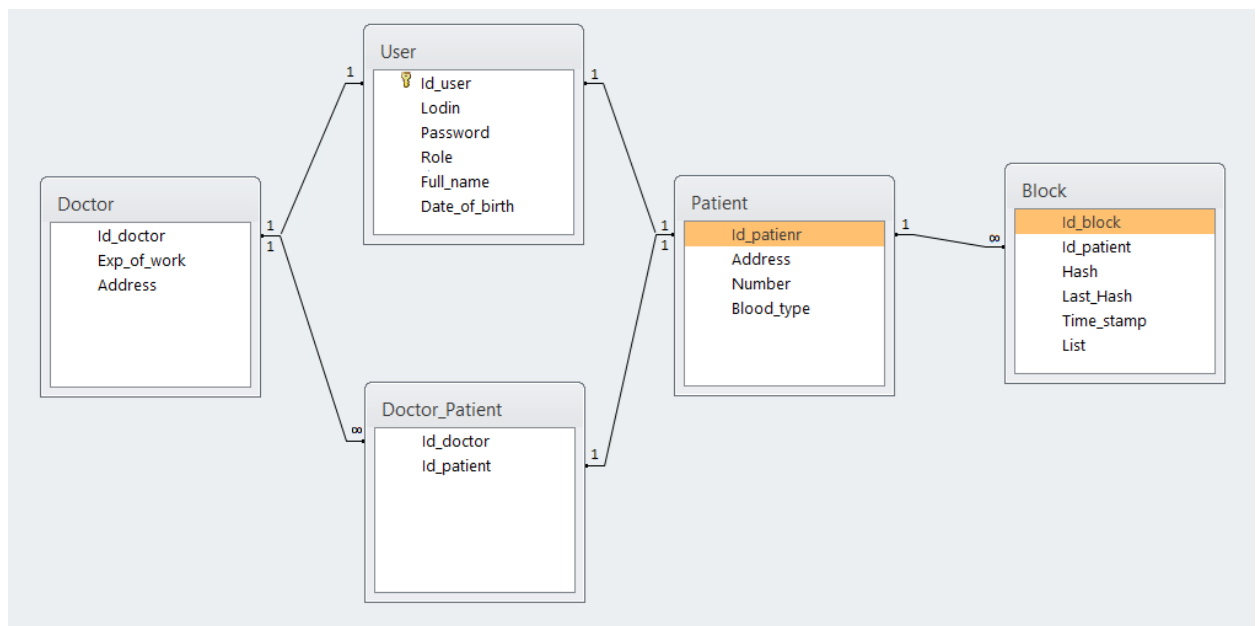


Рисунок 2.1 – ER-діаграма БД

Як видно з рис 2.1 між парами сутностей User та Doctor, User та Patient, Patient та Doctor_Patient зв'язок один до одного, який вказує на те, що екземпляру однієї сутності відповідає один екземпляр іншої сутності у нашому випадку це відповідність ідентифікаторів у різних сутностях один одному. Між парами сутностей Doctor та Doctor_Patient, Patient та Block зв'язок один до багатьох, який означає, що одному екземпляру з перших сутностей може відповідати декілька екземплярів других сутностей.

Модуль вводу-виводу даних – модуль, що працює на стороні клієнта та призначений для обробки дій користувача і передачі сигналів на сторону серверної частини.

Для реалізації модуля вводу-виводу даних була застосована технологія React-Redux. React-Redux є додатковою службою бібліотеки ReactJS та реалізує зв'язок вебзастосунку з модулем зберігання даних через взаємодію клієнта з інтерфейсом та проходження компонентів через життєві цикли існування компоненту. Технологія React-Redux може використовувати як прості(string, number, boolean, symbol), так і складні(object, array) типи даних, що реалізовані мовою JavaScript.

Переваги використання React-Redux:

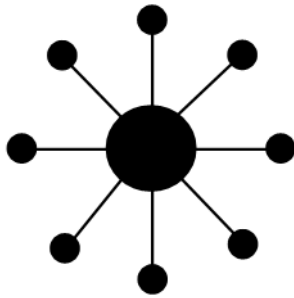
- легкість використання подій та сценаріїв;
- отримання доступу до збережених даних у будь-якому компоненті системи;
- можливість написання сценаріїв до заздалегідь прописаних подій;
- можливість організовувати власну архітектуру організації пам'яті.

Саме цьому в якості модуля вводу-виводу даних було обрано технологію React-Redux.

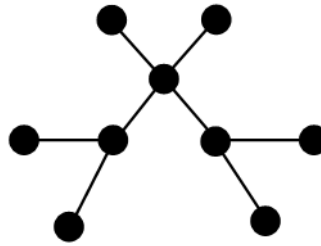
Оброблюючий модуль – модуль, що призначений для обробки деяких запитів від клієнтської частини для формування документів, що призначені для виписних листів на основі блокчейну.

Блокчейн – децентралізований цифровий реєстр транзакцій, що зберігається на великому проміжку часу та захищений засобами шифрування даних. Дані реєстру блокчейну розподіляються в мережу комп'ютерів. Його користувачі можуть напряду взаємодіяти зі збереженими даними в режимі реального часу, не потребуючи посередника для підтвердження достовірності транзакцій. Ця технологія забезпечує незалежну, захищену від зовнішнього втручання та прозору платформу для учасників блокчейну, що дозволяє безпечно зберігати, передавати та обробляти конфіденційну інформацію. Загальні схеми мережей представлено на рис. 2.2.

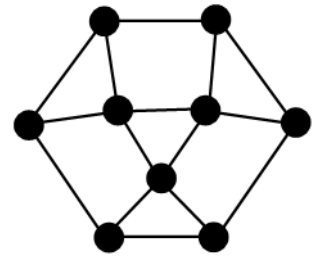
Моделі мережі



Централізована



Децентралізована



Розподілена

Рисунок 2.2 – Загальна схема моделей мережі

Шифрування – технологія захисту інформації та зв'язку, що використовується в блокчейні для передавання та збереження даних для того, щоб доступ до них могли мати лише ті отримувачі, для яких вони призначені.

Хеш-код – різновид шифрування, в блокчейні це зашифрований рядок букв і цифр, що постійний та унікальний для кожного блоку ланцюжка. Іноді хеш-код називають «цифровим відбитком».

Блок – «запечатаний» відсік даних, що містить: особистий хеш-код, хеш-код попереднього блоку та сукупність транзакцій з часовими мітками.

Транзакція в блокчейні – перевірений учасниками запис даних, що служить майже незмінним підтвердженням достовірності комерційної угоди.

БД на основі блокчейну для збереження транзакцій мрже використовуватися для рішення найрізноманітніших задач. Перепони, що заважають наразі широкому розповсюдженню блокчейну, з часом скоріше за все будуть усунені. Хоча широкого рекламування початку використання блокчейну сягає корінням у сверу валют та фінансів, ця технологія вже зачіпає багато галузей людського побуту, а саме:

- управління ланцюжками поставок – в поєднанні з належним чином перевіреними методами ведення бізнесу блокчейн може слугувати методом для перевірки документування ланцюжків поставок;

- смарт-контракти – забезпечують швидкість, ефективність та захищеність за рахунок включення умов згоди в транзакцію блокчейну, в смарт-контрактах на основі блокчейну всі умови угоди перелічуються, змінюються та узгоджуються без необхідності створення фізичного документа;

- медичні записи – потенційно блокчейн дозволяє стандартизувати захищений обмін медичними записами між постачальниками послуг в електронній формі, що було б менш обтяжливо у порівнянні з раніше використовуваними підходами;

- державні послуги – національні системи управління встановлення особистості, моніторинг збору податків, голосування та управління земельними ресурсами – це лише деякі приклади використання екосистеми блокчейн владою.

Модуль зв'язку – основний модуль, що призначений для обробки даних між модулями вводу-виводу даних, оброблюючим модулем та модулем зберігання даних. Він відповідає за обробку інформації, її валідацію та передачу до інших модулів.

Для реалізації модуля обробки та модуля зв'язку було затосована платформа Node.js, а зокрема вебфреймворк Express.

Node.js – платформа з відкритим кодом для виконання високопродуктивних мережових застосунків, написаних мовою JavaScript. Node.js надає можливість виконувати JavaScript-сценарії на сервері та відправляти результати їх виконання користувачеві. Для керування модулями використовується пакетний менеджер «node package manager»(npm) [15].

Express – фреймворк розробки серверної частини вебзастосунків для Node.js, реалізований як вільне і відкрите програмне забезпечення. Він спроектований для створення серверної частини вебзастосунків [16].

На рис. 2.3 представлена структурна схема розроблюваного ПП.

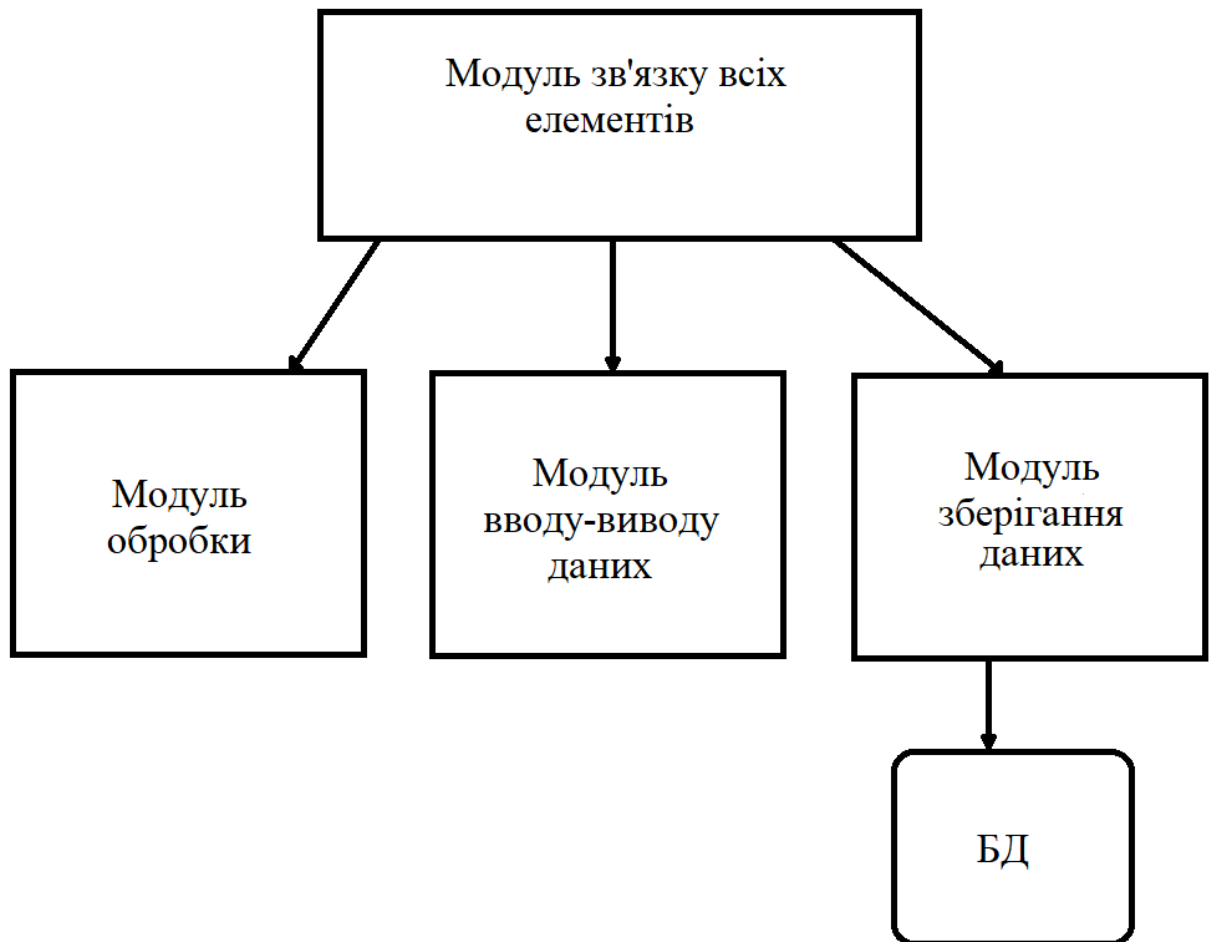


Рисунок 2.3 – Структурна схема програмного продукту

Отже, як видно з рис. 2.3 розроблюваний вебзастосунок має модуль зберігання даних, який виконує запити та приймає відповіді від СУБД. Для зберігання даних було обрано СУБД MongoDB. Модуль вводу-виводу даних відповідає за спілкування з користувачем для отримання від нього інформації та дій, а також за виведення відповідей інших модулів. Для його розробки було обрано технологію React-Redux. Для обробки деяких подій від користувача системи використовується модуль обробки. Модуль зв'язку отримує та оброблює події як серверної, так і клієнтської частини та виконує сценарії, що відповідають цим подіям. Для реалізації модулів обробки та зв'язку було використано платформу Node.js та фреймворк Express.

2.3 Висновок

В даному розділі було проведено аналіз та вибір інструментарію розробки вебзастосунку та було спроектовано архітектуру майбутнього вебзастосунку. Під час аналізу було виконано порівняльну характеристику мов програмування вебзастосунків та встановлено, що найбільші переваги для розробки приведених модулів має мова програмування JavaScript. Також було обрано середовищем розробки IDE Visual Studio Code, на основі порівняння його характеристик з іншим середовищем розробки та встановлено, що Visual Studio Code, найбільш задовольняє потреби для розробки даної системи. В якості системи для зберігання даних було обрано MongoDB.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

В процесі виконання роботи розглядається проблема захисту даних в електронній картці пацієнта. Дану проблему можна роз'язати примінивши технологію блокчейн.

Завданням роботи є програмна реалізація методів захисту даних в електронній картці пацієнта за допомогою технології блокчейн.

Для виконання поставленої мети необхідно створити метод, що реалізує алгоритм блокчейн, та примінити його до електронної картки пацієнта.

3.1 Метод реалізації алгоритму блокчейну для захисту даних електронної картки пацієнта

Для реалізації алгоритму блокчейну необхідно зрозуміти принцип дії та основний процес транзакцій в блокчейні.

Додавання нової запису(блоку) в послідовність блокчейну потребує встановлення її достовірності великою кількістю учасників, що підключені до мережу блокчейн. Ці блоки даних зв'язуються один з одним, що формує ланцюжок.

Технологія блокчейн використовує алгоритм присвоєння кожному блоку зашифрованого хеш-коду. Окрім хеш-коду, кожен блок містить сукупність попередніх транзакцій з мітками часу, а також хеш-код попереднього блоку, який служить незмінною сполучної ланкою між блоками ланцюжка, що ідуть один за одним.

Шифрування та часові мітки в сукупності дозволяють технології блокчейн автоматично перевіряти незмінність цієї послідовності хеш-кодів, що постійно збільшується.

На першому етапі необхідно визначити структуру блоку, тобто виділити елементи, що повинні міститися в блоці. Для цього виділимо, що для роботи нам обов'язково потрібні наступні елементи:

- индекс(Id_block) – для ідентифікування блоку в ланцюжку;
- индекс пацієнта – для подальшого пошуку електронних записів в електронній картці пацієнта;
- хеш-код – набір букв і цифр для однозначного визначення блоку в ланцюжку;
- хеш-код попереднього блоку – для структурної цілісності ланцюжка;
- часова мітка – мітка часу коли було створено блок та проведено транзакцію;
- дані – дані про виписний лист пацієнта.

На другому етапі ми повинні створити перший блок ланцюжка за заданою структурою та провести його підтвердження.

Хешування блоку потрібно для збереження цілісності даних та ланцюжка. В нашому випадку було використано алгоритм SHA-256.

SHA-256 – алгоритм хешування з сімейства SHA-2, що містить в собі набір криптографічних хеш-функцій. Хеш-функції цього сімейства побудовані на основі структури Меркла-Дамгарда.

Оригінал тексту після доповнення розбивається на блоки, кожен блок – на 16 слів. Алгорит пропускає кожен блок тексту через цикл з 64 ітераціями. На кожній ітерації 2 слова перетворюються, функцію перетворення задають інші слова. Результати обробки кожного блоку складаються, сумою є значення хеш-функції.

Для генерації блоку необхідно знати хеш-код попереднього блоку та внести інші елементи, які ми зазначили в структурі блоку.

Для збереження на початку роботи потрібно створити перший блок, що називається «генезис-блоком». Він створений для того, щоб у наступних записів обов'язково був хеш-код попереднього блоку.

Після створення блоку ми підтверджуємо цілісність блоку та приймаємо його до ланцюжка.

На останньому етапі ми повинні сповістити інші вузли мережі про оновлення ланцюжка.

3.2 Проєктування інтерфейсу

Для розробки клієнтської частини вебзастосунку було використано інструмент React-Redux, який є додатковим інструментом бібліотеки ReactJS. Кожен елемент клієнтської частини являє собою елемент гіпертекстової розмітки мови HTML, що вбудована до інструменту React-Redux і дозволяє створювати інтерфейс для спілкування з користувачем вебзастосунку.

Опис вебсторінок для роботи з різними користувачами наведено у наступних підрозділах.

3.2.1 Вебсторінка авторизації та реєстрації

Вебсторінки «Аторизація» та «Реєстрація» представляє собою декілька елементів керування для авторизації користувача в системі або реєстрація нового користувача в системі.

Для авторизації необхідно заповнити поля «Login» та «Password» та натиснути кнопку «Авторизація». Після вище описаних дій користувач авторизується в системі і увійде до однієї з двох вебсторінок, що призначені для користувачів з відповідними ролями.

Якщо у користувача не створено облікового запису він може використати кнопку «Реєстрація» на вебсторінці «Авторизація» та перейти на вебсторінку «Реєстрація» для нових користувачів системи. На вебсторінці «Реєстрація» користувачу необхідно заповнити обов'язкові поля значеннями та натиснути кнопку «Зареєструватися». Після чого користувача поверне на

вебсторінку «Авторизація», де йому потрібно заповнити поля та потрапити в систему.

Перелік елементів вебсторінки «Авторизація» приведено в табл. 3.1.

Таблиця 3.1 – Перелік елементів вебсторінки авторизації користувача

Назва елементу	Тип	Опис
Login	Input	Текстове поле для отримання від користувача логіну.
Password	Input	Текстове поле для отримання від користувача пароля.
Реєстрація	Button	Кнопка, яка дозволяє перейти до вебсторінки «Реєстрація».
Авторизація	Button	Кнопка, яка дозволяє увійти до системи на основі введених логіну та пароля

Функції елементів вебсторінки «Авторизація» представлено в табл.3.2.

Таблиця 3.2 – Функції-оброблювачі подій вебсторінки «Авторизація»

Назва функції	Синтаксис	Опис
Реєстрація	onClick={registerHandler}	Кнопка призначена для переходу на сторінку «Реєстрація»
Авторизація	onClick={authHandler}	Кнопка призначена для входу в систему на основі введених даних

Перелік елементів вебсторінки «Реєстрація» приведено в табл. 3.3.

Таблиця 3.3 – Перелік елементів вебсторінки «Реєстрація» користувача

Назва елементу	Тип	Опис
Login	Input	Текстове поле для введення користувачем логіну.
Password	Input	Текстове поле для введення користувачем пароля.
ПІБ	Input	Текстове поле для введення користувачем прізвища, імені та по батькові.

Продовження таблиці 3.3

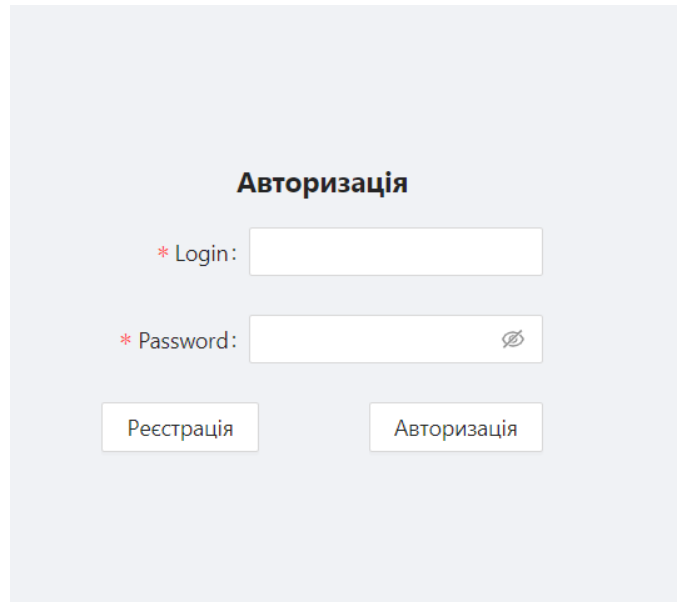
Назва елемента	Тип	Опис
Дата народження	Daterangepicker	Елемент у форматі дати для введення користувачем дати народження.
Роль	Select	Елемент вибору серед існуючих даних для вибору користувачем ролі.
Gmail	Input	Текстове поле для введення користувачем поштової скриньки Gmail.
Телефон	Input	Спеціалізоване текстове поле для введення користувачем номеру телефону, для користувачів з роллю «Пацієнт».
Група крові	Input	Спеціалізоване текстове поле для введення користувачем групи крові, для користувачів з роллю «Пацієнт»..
Стаж роботи	Input	Спеціалізоване текстове поле для введення користувачем стажу роботи, для користувачів з роллю «Лікар»..
Зареєструватися	Button	Кнопка для реєстрації користувача в системі.
Назад	Button	Кнопка для повернення на сторінку авторизації

Функції елементів вебсторінки «Реєстрація» представлено в табл.3.4.

Таблиця 3.4 – Функції-оброблювачі подій вебсторінки «Реєстрація»

Назва функції	Синтаксис	Опис
Зареєструватися	onClick={registrationHendler}	Кнопка призначена для реєстрації нового користувача системи на основі введених даних
Назад	onClick={backHendler}	Кнопка призначена для повернення на сторінку «Авторизація»

Зовнішній вигляд вебсторінки «Авторизація» представлено на рис. 3.1, зовнішній вигляд вебсторінки «Реєстрація» представлено на ри 3.2.

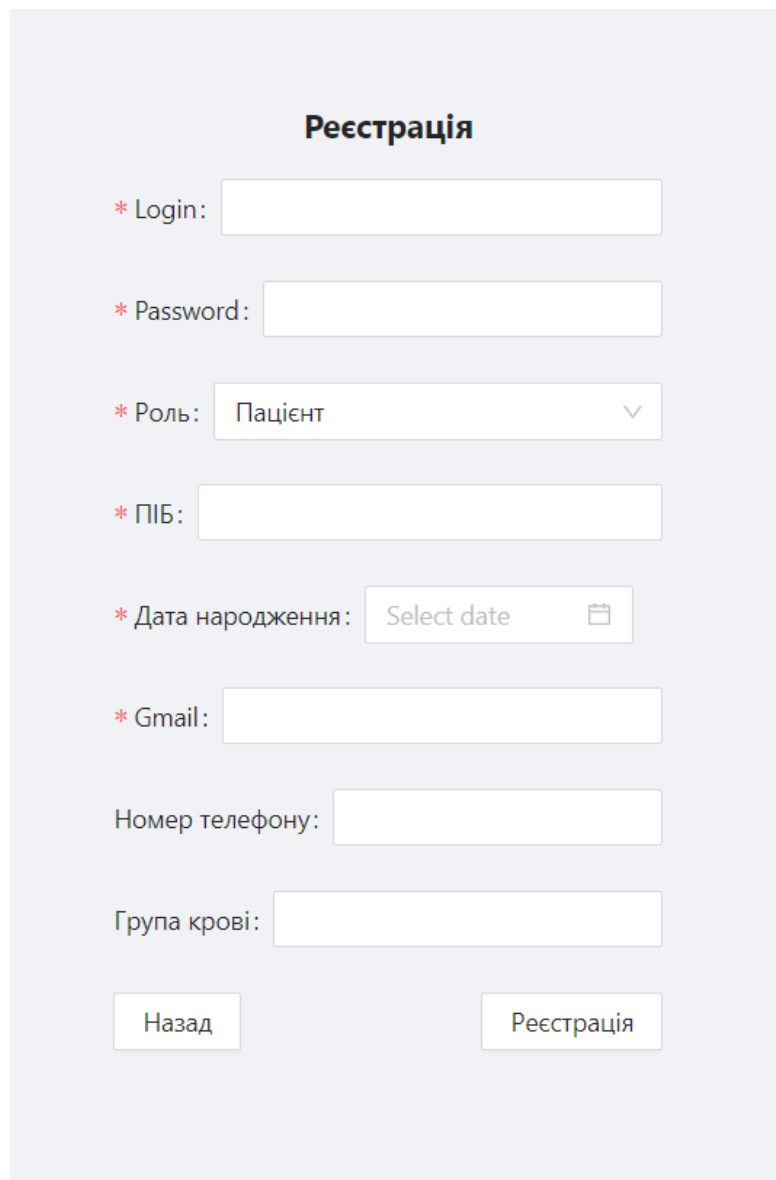


Авторизація

* Login:

* Password: 

Рисунок 3.1 – Зовнішній вигляд вебсторінки «Авторизація»



Реєстрація

* Login:

* Password:

* Роль: 

* ПІБ:

* Дата народження: 

* Gmail:

Номер телефону:

Група крові:

Рисунок 3.2 – Зовнішній вигляд вебсторінки «Реєстрація»

3.2.2 Вебсторінка пацієнта

Після авторизації користувача з роллю «Пацієнт» він потрапляє на вебсторінку «Пацієнт» з особистою інформацією про користувача. Пацієнт має можливість прикріпитися до сімейного лікаря для подальшого отримання медичної допомоги. Пацієнт має можливість отримати свої попередні виписні листи на поштову скриньку Gmail, що була введено в процесі реєстрації.

Перелік елементів вебсторінки «Пацієнт» приведено в табл. 3.5.

Таблиця 3.5 – Перелік елементів вебсторінки «Пацієнт» користувача

Назва елементу	Тип	Опис
ПІБ	Text	Текстове поле для виведення інформації про прізвища, імені та по батькові пацієнта.
Дата народження	Text	Текстове поле для виведення інформації про дату народження пацієнта.
Gmail	Text	Текстове поле для виведення інформації про поштову скриньку Gmail пацієнта.
Номер телефону	Text	Текстове поле для виведення інформації про номеру телефону пацієнта.
Група крові	Text	Текстове поле для виведення інформації про групи крові пацієнта.
Вибір лікаря	Select	Елемент вибору серед існуючих даних для вибору пацієнтом сімейного лікаря. Елемент відсутній у користувачів, що обрали лікаря
Обрати	Button	Кнопка для підтвердження вибору сімейного лікаря, що обрано у попередньому елементі вебсторінки. Кнопка відсутня у користувачів, що обрали лікаря
Список виписних листів	Table	Табличний елемент для перегляду та обрання виписних листів для відправлення на поштову скриньку пацієнта.

Функції елементів вебсторінки «Пацієнт» представлено в табл.3.6.

Таблиця 3.6 – Функції-оброблювачі подій вебсторінки «Пацієнт»

Назва функції	Синтаксис	Опис
Обрати	<code>onClick={()=>setDoctor(selectedDoctor)}</code>	Кнопка для підтвердження вибору лікаря та відправлення даних до системи
Відправити документ	<code>onClick={()=>setDoctor(selectedDoctor)}</code>	Кнопка, що дозволяє відправити документ у відповідному рядку до поштової скриньки користувача

Зовнішній вигляд вебсторінки «Пацієнт» представлено на рис. 3.3.

Добрий день, Sergienko O.V.			
Дата народження: 10.10.2014			
Gmail: wlad.scheludko@gmail.com			
Номер телефону: 0665409080			
Ваш сімейний лікар: Matish A. U.			
Номер документа	Діагноз	Дата виписки	Відправити
60a5141af82334439879155c	Kurva	11.05.2021	Відправити документ
60a514581f2d884fb0ef3fa6	Privetiky	12.05.2021	Відправити документ
60a78b905f351c481455ce5d	Privetiky	12.05.2021	Відправити документ

Рисунок 3.3 – Зовнішній вигляд вебсторінки «Пацієнт»

3.2.3 Вебсторінка лікаря

Після авторизації користувача з роллю «Лікар» він потрапляє на вебсторінку «Лікар» з особистим кабінетом для роботи. Лікар може переглядати виписні листи пацієнтів, що обрали його в своєму особистому кабінеті. Також лікар може створювати виписні листи для пацієнтів, що його обрали, для цього йому потрібно натиснути на кнопку «Створити» та у модальному вікні заповнити дані.

Перелік елементів вебсторінки «Лікар» приведено в табл. 3.7.

Таблиця 3.7 – Перелік елементів вебсторінки «Лікар» користувача

Назва елементу	Тип	Опис
ПІБ	Text	Текстове поле для виведення інформації про прізвища, імені та по батькові лікаря.
Дата народження	Text	Текстове поле для виведення інформації про дату народження лікаря.
Gmail	Text	Текстове поле для виведення інформації про поштову скриньку Gmail лікаря.
Страж роботи	Text	Текстове поле для виведення інформації про стаж роботи лікаря.
Список пацієнтів	Table	Табличний елемент для перегляду та обрання пацієнтів, що обрали лікаря в особистому кабінеті
Створити	Button	Кнопка для створення виписного листа для пацієнта, що обрано у попередньому елементі вебсторінки.
Список виписних листів	Table	Табличний елемент для перегляду та обрання виписних листів обраного пацієнта для відправлення на поштову скриньку лікаря.

Функції елементів вебсторінки «Пацієнт» представлено в табл.3.8.

Таблиця 3.8 – Функції-оброблювачі подій вебсторінки «Лікар»

Назва функції	Синтаксис	Опис
Обрати	<pre>onClick={() => { getListPatient(record.login); const selected = patientList.find(({login})=> login===record.login) setSelectedPatient(selected)}}}</pre>	Кнопка призначена для вибору пацієнта для перегляду або створення його виписних документів
Створити	<pre>onClick={() => { showModal()}}</pre>	Кнопка призначена для створення виписного документа для пацієнта лікарем, що перебуває у системі
Відправити документ	<pre>onChange={() => getEmail(record.numDoc)}</pre>	Кнопка, що дозволяє відправити документ у відповідному рядку до поштової скриньки користувача

Зовнішній вигляд вебсторінки «Лікар» представлено на рис. 3.4.

Добрий день, Matish A. U.

Дата народження: 01.10.1990
Gmail: wlad.scheludko@gmail.com
Стаж роботи: 11

Login пацієнта	ПІБ	Документи
Login1	Sergienko O.V.	Обрати
Login4	Konienko A.V.	Обрати

Створити

Номер документа	Діагноз	Дата виписки	Відправити
60a5141af82334439879155c	Kurva	11.05.2021	Відправити документ
60a514581f2d884fb0ef3fa6	Privetiky	12.05.2021	Відправити документ
60a78b905f351c481455ce5d	Privetiky	12.05.2021	Відправити документ

Рисунок 3.4 – Зовнішній вигляд вебсторінки «Лікар»

3.3 Висновок

В даному розділі було виконано опис алгоритму роботи блокчейну для збереження цілісності даних виписних листів пацієнта. Описано основний алгоритм роботи блокчейну та його використання для електронної картки пацієнта.

Також в даному розділі було описано вебсторінки вебзастосунку, які складаються з вебсторінок «Авторизація», «Реєстрація», «Пацієнт» та «Лікар».

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови використання

Розроблений вебзастосунок призначений для підтримки електронної картки пацієнта на основі блокчейну.

Вебзастосунок повинен відповідати наступним вимогам:

- всі сторінки вебзастосунку та інтегровані в них елементи повинні бути стислими за змістом;
- на всіх сторінках повинен застосовуватися один дизайнерський стиль;
- імена файлів повинні бути «смысловими»;
- не використовувати на одній вебсторінці більше 3 різних шрифтів;
- не використовувати на одній вебсторінці більше 3 різних кольорів;
- використовувати тільки коректні кольорові схеми.

4.2 Характеристики програмного продукту

Вебзастосунок містить в собі наступні модулі:

- зберігання даних. Для зберігання даних та зв'язку з СУБД MongoDB
 - оброблюючий. Цей модуль реалізує технологію блокчейн та використовує її для захисту даних електронної картки пацієнта;
 - вводу-виводу даних. Даний модуль дозволяє отримувати дані та події від користувача системи та передавати їх до інших модулів, а також використовується для виведення результатів виконання серверних операцій користувачу;
 - модуль зв'язку всіх елементів ПП. Цей модуль є основним модулем вебзастосунку та призначений для керування та взаємодії між іншими модулями та виконанням сценаріїв на відповідні події.

4.3 Звертання до програмного продукту (файли проєкту)

Для того, щоб використовувати вебзастосунок необхідно запустити серверну частину вебзастосунку, а саме відкрити серверну частину у Visual Studio Code та запустити 2 каталоги з назвами «back» та «front» , після чого у будь-якому браузері, що встановлений на комп'ютері, у адресну строку ввести `http://localhost:3000`:

4.4 Вхідні і вихідні дані

Вхідними даними вебзастосунку є дані, що було введені користувачами в процесі реєстрації, авторизації та заповнення виписних листів. Ці дані отримуються під час дій користувача, що створюють події, які викликають відповідні сценарії в серверній частині.

Вихідними даними є виписні листи, що відправляються на поштову скриньку Gmail користувачів системи.

4.5 Повідомлення

Під час роботи розроблений вебзастосунок генерує повідомлення, що описані в табл. 4.1.

Таблиця 4.1 – Повідомлення під час роботи

Повідомлення	Опис
Wronld login. Please, try again.	Повідомлення означає, що за введеними логіном та паролем не існує користувачів в БД.
Заповніть обов'язкові поля	Повідомлення означає, що необхідно заповнити усі поля, що відмічені на вебсторінці як обов'язкові.

Продовження таблиці 4.1

Повідомлення	Опис
List sozdan	Повідомлення означає, що виписний лист пацієнта було створено та додано до БД.

4.6 Висновок

В даному розділі було проведено детальний опис призначення та умови використання розробленого вебзастосунку. Визначені вимоги до вебзастосунку. Вказані дії, що потрібно виконати, для роботи вебзастосунку. Виконано опис вхідних та вихідних даних, а також перелік повідомлень для користувачів, які генерує серверна частина вебзастосунку в залежності від дій користувача при роботі з вебзастосунком.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення й умови виконання програмного продукту

Розроблений вебзастосунок призначений для підтримки електронної картки пацієнта на основі блокчейну.

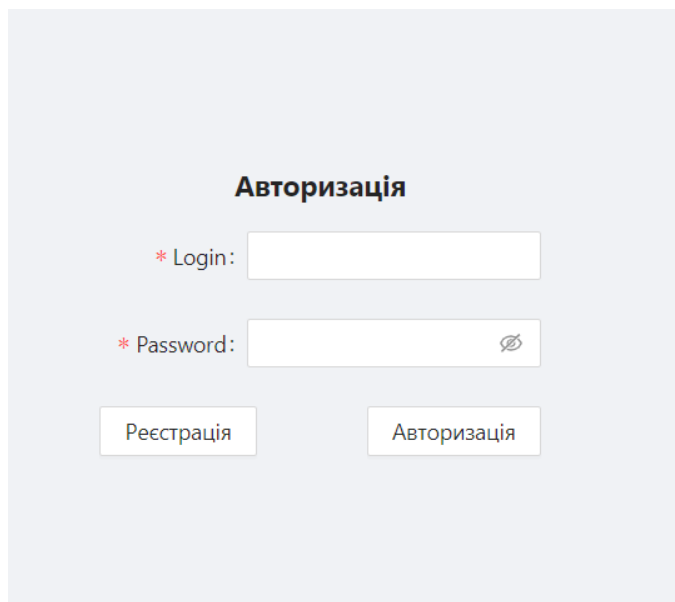
Вебзастосунок повинен відповідати наступним вимогам:

- всі сторінки вебзастосунку та інтегровані в них елементи повинні бути стислими за змістом;
- на всіх сторінках повинен застосовуватися один дизайнерський стиль;
- імена файлів повинні бути «смысловими»;
- не використовувати на одній вебсторінці більше 3 різних шрифтів;
- не використовувати на одній вебсторінці більше 3 різних кольорів;
- використовувати тільки коректні кольорові схеми.

5.2 Виконання програмного продукту

Після запуску серверної частини вебзастосунку та відкриття в адресному рядку браузеру `http://localhost:3000` відкривається вебсторінка «Авторизація».

Для авторизації необхідно заповнити поля «Login» та «Password» та натиснути кнопку «Авторизація», зовнішній вигляд вебсторінки «Авторизація» представлено на рис. 5.1.



Авторизація

* Login:

* Password: 

Рисунок 5.1 – Зовнішній вигляд вебсторінки «Авторизація»

Після реєстрації користувач потрапляє на одну з двох сторінок вебзастосунку, яка залежить від ролі користувача, що була обрана при реєстрації в систему.

Для ролі «Пацієнт» відкривається вебсторінка «Пацієнт», на якій пацієнт може побачити свої особисті дані, що були введені на момент реєстрації, а також має можливість обрати лікаря та отримати особисті виписні документи на поштову скриньку Gmail. Для отримання виписного документу необхідно обрати документ в таблиці та натиснути на кнопку «Відправити». Зовнішній вигляд вебсторінки «Пацієнт» представлено на рис. 5.2.

Добрий день, Sergienko O.V.			
Дата народження: 10.10.2014			
Gmail: wlad.scheludko@gmail.com			
Номер телефону: 0665409080			
Ваш сімейний лікар: Matish A. U.			
Номер документа	Діагноз	Дата виписки	Відправити
60a5141af82334439879155c	Kurva	11.05.2021	Відправити документ
60a514581f2d884fb0ef3fa6	Privetiky	12.05.2021	Відправити документ
60a78b905f351c481455ce5d	Privetiky	12.05.2021	Відправити документ

Рисунок 5.2 – Зовнішній вигляд вебсторінки «Пацієнт»

Для ролі «Лікар» відкривається вебсторінка «Лікар», на якій лікар може побачити свої особисті дані, що були введені на момент реєстрації, а також має можливість переглянути пацієнтів, що обрали його у своїх особистих кабінетах, переглянути їх виписні документи, отримати особисті виписні документи пацієнтів на поштову скриньку Gmail, а також створити виписні документи для обраних пацієнтів. Для отримання виписних документів пацієнтів необхідно обрати пацієнта в таблиці «Пацієнти», після чого обрати виписний документ в таблиці «Документи» та натиснути кнопку «Відправити». Для створення виписних документів необхідно обрати пацієнта в таблиці «Пацієнти», після чого натиснути на кнопку «Створити», після чого відкривається модальне вікно для введення інформації, яку потрібно заповнити, та при натисненні кнопки «Створити» відбудеться відправлення введеної інформації на зберігання до сховища вебзастосунку. Зовнішній вигляд вебсторінки «Лікар» представлено на рис. 5.3.

Добрий день, Matish A. U.

Дата народження: 01.10.1990

Gmail: wlad.scheludko@gmail.com

Стаж роботи: 11

Login пацієнта	ПІБ	Документи
Login1	Sergienko O.V.	<button>Обрати</button>
Login4	Konienko A.V.	<button>Обрати</button>

Створити

Номер документа	Діагноз	Дата виписки	Відправити
60a5141af82334439879155c	Kurva	11.05.2021	<button>Відправити документ</button>
60a514581f2d884fb0ef3fa6	Privetiky	12.05.2021	<button>Відправити документ</button>
60a78b905f351c481455ce5d	Privetiky	12.05.2021	<button>Відправити документ</button>

Рисунок 5.3 – Зовнішній вигляд вебсторінки «Лікар»

5.3 Повідомлення оператору

Під час роботи розробленого вебзастосунку генеруються повідомлення, які виникають у відповідь на дії користувача. Перелік можливих повідомлень та їх опис зазначено в розділі «Керівництво програміста» в таблиці 4.1. дії зазначений в розділі «Керівництво програміста» в таблиці 4.1.

5.4 Висновок

В даному розділі було проведено опис призначення та умови використання розробленого вебзастосунку. Визначені вимоги до вебзастосунку. Вказані дії, що потрібно виконати, для роботи вебзастосунку. Була описана покрокова інструкція користувача вебзастосунку, в якій вказано, які дії користувач має здійснити для авторизації, реєстрації, отримання виписних документів на поштову скриньку та створення виписних документів для інших користувачів.

ВИСНОВКИ

Під час виконання роботи було виконано дослідження предметної області, розглянуто та визначено основні поняття, які пов'язані з обраною предметною областю. В даній роботі було проаналізовано алгоритм блокчейн для його використання при вирішенні поставленого завдання із захисту електронної картки пацієнта. Було визначено та сформульовано завдання до роботи.

Метою роботи є створення вебзастосунку, що дозволяє користувачам отримувати або створювати виписні медичні документи.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- реалізовано процес реєстрації та авторизації користувачів в системі;
- надано можливість пацієнту обирати лікаря для подальшого отримання виписних документів;
- надано можливість пацієнту переглядати свої документи;
- налаштувано систему для перегляду даних пацієнтів та створення виписних документів;
- реалізувано алгоритм захисту даних про документи пацієнта на основі технології блокчейн.

Також в роботі були розглянуті різні мови програмування вебзастосунків та середовища розробки, які дозволяють розробляти програмні продукти. Було проведено їх аналіз, виділено основні переваги і недоліки та складено порівняльну характеристику. В ході цієї роботи було встановлено, що для розробки вебзастосунку найбільш ефективною мовою програмування є JavaScript, бо вона володіє тими перевагами, що забезпечують зручне написання коду, швидкість його виконання та продуктивність роботи вебзастосунків на її основі. Також найбільш ефективним середовищем розробки було обрано середовище Visual Studio

Code, бо в цьому середовищі розробки дозволяє швидко налаштувати середовище розробки та відлагоджувати код проекту.

Для роботи було розроблено структурну схему програмного продукту. Встановлено, що для зберігання вхідних даних найкращий тип сховища є база даних MongoDB.

Графічний інтерфейс вебзастосунку було реалізовано за допомогою використання бібліотеки ReactJS у середовищі розробки Visual Studio Code.

В результаті було створено вебзастосунок для підтримання електронної картки пацієнта на основі блокчейну. Створений вебзастосунок дозволяє зберігати дані в захищеній алгоритмом блокчейн БД.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Словник української мови : у 20 т. / НАН України, Укр. мов.-інформ. фонд. – К. : Наукова думка, 2010–2020.
2. Закон України «Про затвердження Загальнодержавної програми розвитку первинної медико-санітарної допомоги на засадах сімейної медицини до 2011 року» від 22 січня 2010 р. № 1841-VI.
3. Болотіна Н.Б. Пацієнт. В кн.: Юридична енциклопедія. – К., 2002. – Т. 4.
4. Хвороба / Хороба // Ганна Дидик-Меуш. Українська медицина. Історія назв. – Львів: Інститут українознавства ім. І. Крип'якевича НАН України, 2008. – С. 119–128.
5. Внутрішня медицина: терапія: підручник / Н. М. Середюк, О. С. Стасишин, І. П. Вакалюк [та ін.]. – 4-те вид., виправлене. – К. : Медицина, 2013. – 686 с.
6. Словник української мови : в 11 т. / АН УРСР. Інститут мовознавства; за ред. І. К. Білодіда. – К. : Наукова думка, 1970–1980.
7. Українська радянська енциклопедія : у 12 т. / гол. ред. М. П. Бажан ; редкол.: О. К. Антонов та ін. – 2-ге вид. – К. : Головна редакція УРЕ, 1974–1985.
8. Великий тлумачний словник сучасної української мови (з дод. і допов.) / уклад. і гол. ред. В. Т. Бусел. – 5-те вид. – К. ; Ірпінь : Перун, 2005.
9. Електронна медицина [Electronic Resource] – Access mode: <https://uk.wikipedia.org/wiki/E-медицина>
10. Юридична енциклопедія : [у 6 т.] / ред. кол. Ю. С. Шемшученко (відп. ред.) [та ін.]. – К. : Українська енциклопедія ім. М. П. Бажана, 2001. – Т. 3 : К – М. – 792 с.
11. О.А. ХОРОЗОВ. Формування бази даних електронних медичних записів // Комп'ютерна математика. 2014, № 1.– С.60–69.
12. Електронна система охорони здоров'я [Electronic Resource] –

Access mode: <https://web.archive.org/web/20180408002147/https://portal.ehealth-ukraine.org/about.html>.

13. Пістун Є.П. Основи автоматики та автоматизації: навч. посіб. / Є. П. Пістун, І. Д. Стасюк; Нац. ун-т «Львів. політехніка». – Львів, 2014. – 333 с. – Бібліогр.: с. 303–304.

14. Дон Тепскотт, Алекс Тепскотт. «Блокчейн-революція». Переклад з англійської Юлії Григоренко, Ганни Лелів. – Львів: Видавництво «Літопис», 2019. 492 с.

15. Nodejs [Electronic Resource] – Access mode: <https://nodejs.org/docs/latest/api>.

16. Expressjs [Electronic Resource] – Access mode: <https://expressjs.com/en/5x/api.html>.

ДОДАТОК А
Технічне завдання

Вступ

Розроблюваний вебзастосунок використовується для ведення електронної картки пацієнта та захист даних пацієнта в мережі за допомогою використання технології блокчейн.

Вебзастосунок «Hospital» має функції аторизації та реєстрації користувачів у системі, дозволяє переглядати основні відомості про виписні документи та пацієнтів, що обрали лікаря в системі, створювати виписні документи та зберігати їх.

A.1 Мета розробки і її призначення

Об'єкт дослідження – вебзастосунок для підтримання роботи електронної картки пацієнта.

Предмет дослідження – методи підтримки електронної картки пацієнта на основі блокчейну.

Мета роботи – програмна реалізація методів для підтримання електронної картки пацієнта на основі блокчейну.

Актуальність та проблематика роботи полягає в тому, що останнім часом існує багато медичних інформаційних систем, що не завжди задовольняють вимоги користувачів систем та їх потреби. Розроблено програмний продукт, який призначений для вирішення завдання про підтримання електронної картки пацієнта, а саме захист даних від підробки у зв'язку з унікальними ключами в кожному записі.

A.2 Стислий зміст основних теоретичних посилок до розробки на основі проведених пошуків

За минулі роки в Україні було здійснено реформу медицини, яка полягала у тому, що населення повинне було перейти від звичного режиму

запису до терапевта для подальшого лікування до прикріплення осіб до сімейних лікарів для ведення їх лікування на основі декларації, що підписується обома сторонами. Разом із цим, лікарі почали переходити на електронну систему запису пацієнтів на прийом та призначення лікування, а також оперативне отримання аналізів пацієнтів через мережу Інтернет та вебзастосунки.

Однак на жаль не всі лікарі та пацієнти задоволені нині існуючими вебзастосунками, для деяких пацієнтів занадто складний спосіб реєстрації на прийом лікаря або немає населеного пункту, в якому вони проживають, для лікарів є деякі нюанси медичних закладів, які відрізняються в залежності від області, де вони розташовані.

A.3 Підстави для розробки

Підставою для виконання розробки є завдання на дипломний проєкт на тему «Програмна реалізація вирішення задачі комівояжера для автомату видачі ліків», затверджене наказом по НУ «Запорізька політехніка» № 103 від 30 березня 2021 року.

A.4 Основні вимоги до програми, що розробляється, початковим даним і результату

A.4.1 Вимоги до функціональних характеристик

Вимоги до функціональних характеристик:

- повинен бути реалізований графічний інтерфейс користувача;
- повинна бути реалізована можливість завантаження даних з БД;
- повинна бути реалізована можливість додавати користувачів до системи;
- повинна бути реалізована можливість авторизації користувачів у системі;

- повинна бути реалізована можливість отримання виписних документів на поштову скриньку Gmail пацієнтів або лікарів;
- повинна бути реалізована можливість отримання створення та додавання виписних документів для користувачів з типом «Лікар»;
- повинна бути реалізований метод для захисту інформації пацієнта на основі технології блокчейн

A.4.2 Вимоги до надійності

ПП повинна виконувати наступні вимоги до надійності:

- захищати вебзастосунок від помилкових дій користувача (наприклад користувач ввів невалідні дані);
- обробляти виняткові ситуації (наприклад, відсутність підключення до бази даних);

Вебзастосунок повинен нормально функціонувати при безперервній роботі браузера. У разі виникнення збоїв в роботі апаратури, відновлення нормальної роботи вебзастосунку повинно проводитися після перезавантаження системи та виконання дій для роботи вебзастосунку.

A.4.3 Умови експлуатації

Умови експлуатації:

- програмне забезпечення не потребує обслуговуючого персоналу;
- для користування програмним забезпеченням не потрібна додаткова кваліфікація.

A.4.4 Вимоги до складу та параметрів технічних засобів

Вимоги до складу та параметрів технічних засобів:

- платформа x64-based;
- операційна система Microsoft Windows 10/8/7;
- процесор з тактовою частотою не нижче 1,8 ГГц. Рекомендується використовувати як мінімум двоядерний процесор;
- 2 ГБ оперативної пам'яті;
- вільне місце на диску 2 ГБ мінімум.

A.4.5 Вимоги до маркування та упакування

Програмне забезпечення не потребує додаткових дій при маркуванні та упакуванні.

A.4.6 Вимоги до транспортування та збереження

Вимоги до транспортування й зберігання аналогічні вимогам, які пропонуються до носія, на якому записана програма.

A.5 Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання;
- керівництво програміста;
- керівництво оператора;
- опис програми.

А.6 Основні обмеження на установку і використання програмного продукту

Розроблюваний вебзастосунок потребує інсталяції. Вебзастосунок повинен використовуватись на комп'ютерах які задовільняють вимогам та параметрам технічних засобів. Вебзастосунок може використовуватись користувачем у будь-який момент часу, не конфліктуючи з іншими програмами.

ДОДАТОК Б**Опис програмного продукту**

Б.1 Загальні відомості

Назва програмного продукту – «Hospital».

Розроблюваний вебзастосунок використовується для ведення електронної картки пацієнта та захист даних пацієнта в мережі за допомогою використання технології блокчейн.

Вебзастосунок «Hospital» має функції аторизації та реєстрації користувачів у системі, дозволяє переглядати основні відомості про виписні документи та пацієнтів, що обрали лікаря в системі, створювати виписні документи та зберігати їх.

Вебзастосунок призначений для запуску в середовищі Windows в браузері на ЕОМ з наступними параметрами:

- платформа x64-based;
- операційна система Microsoft Windows 10/8/7;
- процесор з тактовою частотою не нижче 1,8 ГГц. Рекомендується використовувати як мінімум двоядерний процесор;
- 2 ГБ оперативної пам'яті;
- вільне місце на диску 2 ГБ мінімум.

Б.2 Функціональне призначення

Вимоги до функціональних характеристик:

- повинен бути реалізований графічний інтерфейс користувача;
- повинна бути реалізована можливість завантаження даних з БД;
- повинна бути реалізована можливість додавати користувачів до системи;
- повинна бути реалізована можливість авторизації користувачів у системі;
- повинна бути реалізована можливість отримання виписних документів на поштову скриньку Gmail пацієнтів або лікарів;

- повинна бути реалізована можливість отримання створення та додавання виписних документів для користувачів з типом «Лікар»;
- повинна бути реалізований метод для захисту інформації пацієнта на основі технології блокчейн

Б.3 Опис логічної структури

Розроблюваний вебзастосунок повинен обов'язково містити такі модулі:

- зберігання даних;
- оброблюючий;
- вводу-виводу даних;
- модуль зв'язку всіх елементів вебзастосунку.

В якості вхідних даних використовують інформацію представлену користувачами в момент реєстрації та інформацію про виписні документи, що представлені користувачами для створення виписних листів. Для збереження даних використовується БД.

Розглянемо опис сутностей з їх атрибутами у вигляді табл. Б.1 за такими критеріями як тип використовуваного атрибута, його розмір, чи повинно бути обов'язково заповнене поле та загальний опис сутності.

Таблиця Б.1 – Сутності та їх атрибути

Сутність	Атрибут	Тип	Розмір	Обов'язкове	Опис сутності
User	Id_user	Лічильник	Довге ціле	Так	Містить інформацію про користувача системи
	Role	Текстовий	Текст	Так	
	Full_name	Текстовий	Текст	Так	
	Date_of_birt h	Текстовий	Текст	Так	
	Login	Текстовий	Текст	Так	
	Password	Текстовий	Текст	Так	

Продовження таблиці Б.1

Сутність	Атрибут	Тип	Розмір	Обов'язкове	Опис сутності
Patient	Id_patient	Числовий	Довге ціле	Так	Містить інформацію про користувача з типом пацієнт
	Address	Текстовий	Текст	Так	
	Number	Текстовий	Текст	Так	
	Blood_type	Текстовий	Текст	Ні	
Doctor	Id_doctor	Числовий	Довге ціле	Так	Містить інформацію про користувача з типом лікар
	Exp_of_work	Текстовий	Текст	Ні	
	Address	Текстовий	Текст	Так	
Doctor_Patient	Id_doctor	Числовий	Довге ціле	Так	Містить інформацію про зв'язок лікарів та пацієнтів
	Id_patient	Числовий	Довге ціле	Так	
Block	Id_block	Числовий	Довге ціле	Так	Містить інформацію про блоки блокчейну
	Id_patient	Числовий	Довге ціле	Так	
	Hash	Текстовий	Текст	Так	
	Last_hash	Текстовий	Текст	Так	
	Time_stamp	Текстовий	Текст	Так	
	List	Об'єкт	Текст	Так	

Як видно з табл. Б.1 під час концептуального проєктування БД визначено такі сутності, як User, що містить основну інформацію про користувачів системи, Patient – містить спеціальну інформацію про користувачів з роллю «Patient», Doctor – містить спеціальну інформацію про користувачів з роллю «Doctor», Doctor_Patient – містить інформацію про зв'язок лікарів та пацієнтів, Block – містить інформацію про виписні листи на основі виконання блоків блокчейну.

На рис. Б.1 представлена ER-діаграма БД.

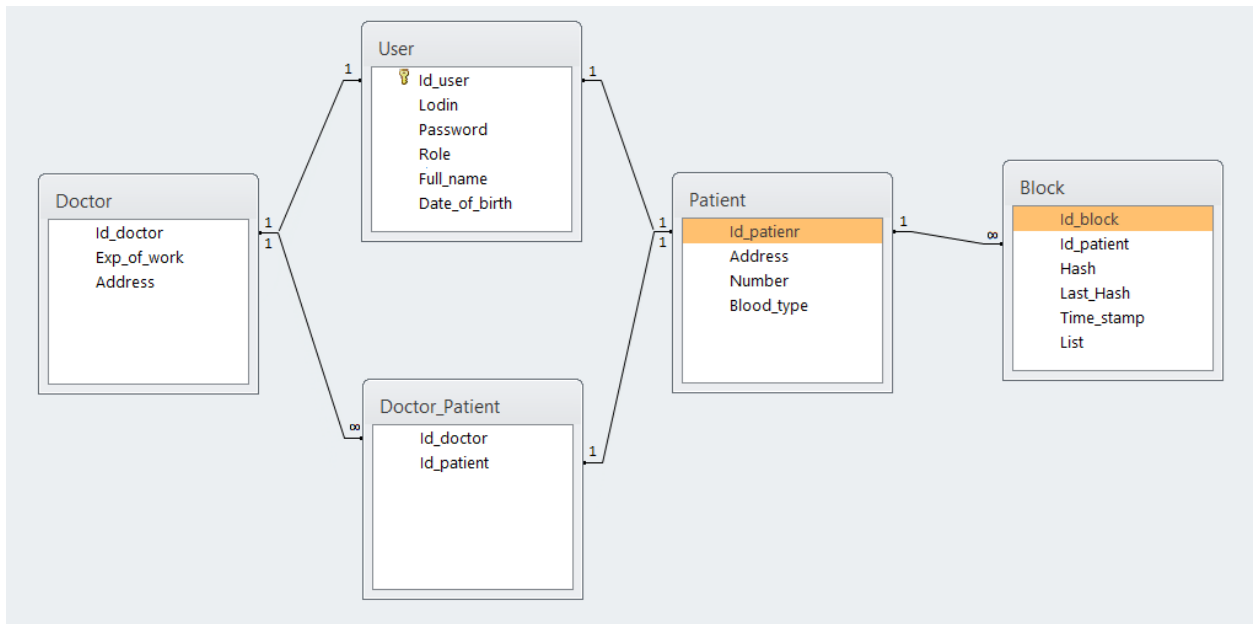


Рисунок Б.1 – ER-діаграма БД

Як видно з рис 2.1 між парами сутностей User та Doctor, User та Patient, Patient та Doctor_Patient зв'язок один до одного, який вказує на те, що екземпляру однієї сутності відповідає один екземпляр іншої сутності у нашому випадку це відповідність ідентифікаторів у різних сутностях один одному. Між парами сутностей Doctor та Doctor_Patient, Patient та Block зв'язок один до багатьох, який означає, що одному екземпляру з перших сутностей може відповідати декілька екземплярів других сутностей.

Модуль вводу-виводу даних – модуль, що працює на стороні клієнта та призначений для обробки дій користувача і передачі сигналів на сторону серверної частини.

Оброблюючий модуль – модуль, що призначений для обробки деяких запитів від клієнтської частини для формування документів, що призначені для виписних листів на основі блокчейну.

Модуль зв'язку – основний модуль, що призначений для обробки даних між модулями вводу-виводу даних, оброблюючим модулем та модулем зберігання даних. Він відповідає за обробку інформації, її валідацію та передачу до інших модулів.

На рис. Б.2 представлена структурна схема розробленого вебзастосунку.

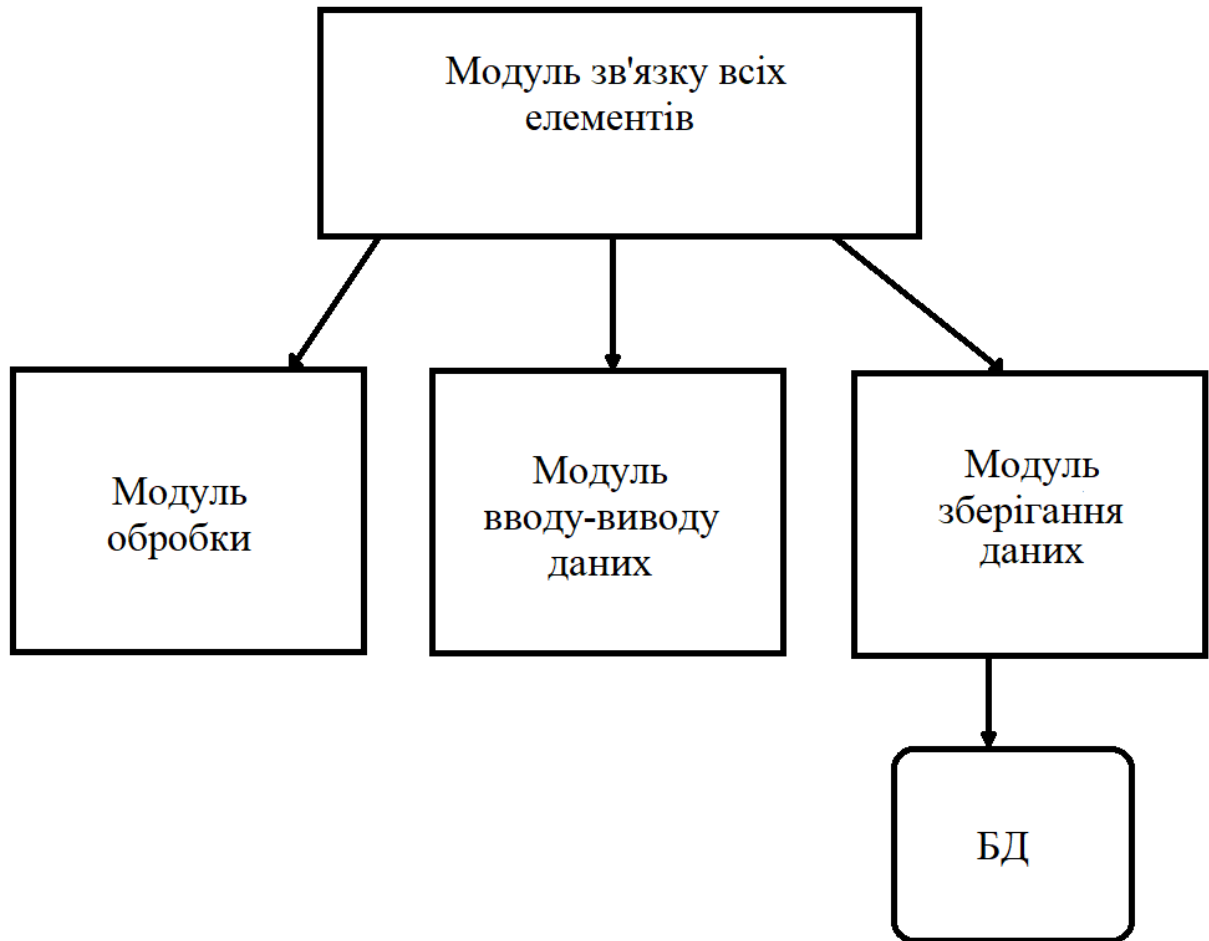


Рисунок Б.2 – Структурна схема програми

Отже, як видно з рис. 2.2 розроблюваний ПП має модуль зберігання даних, який виконує запити та приймає відповіді від СУБД. Для зберігання даних було обрано СУБД MongoDB. Модуль вводу-виводу даних відповідає за спілкування з користувачем для отримання від нього інформації та дій, а також за виведення відповідей інших модулів. Для його розробки було обрано технологію React-Redux. Для обробки деяких подій від користувача системи використовується модуль обробки. Модуль зв'язку отримує та оброблює події як серверної, так і клієнтської частини та виконує сценарії, що відповідають цим подіям.

Б.4 Вибір інструментарію розробки програмного продукту

Для розробки вебзастосунку було обрано мову програмування JavaScript, бо незважаючи на складність вивчення вона володіє тими перевагами, що забезпечують зручне написання коду, швидкість його виконання та продуктивність роботи вебзастосунків на її основі. Мова JavaScript має велику кількість фреймворків та бібліотек з готовими рішеннями більшої частини завдань та документацію для цих бібліотек. Дана мова широко поширена, має велику спільноту розробників та документацію, що спрощує розуміння використання мови. Мова JavaScript підтримується на більшості сучасних браузерів. Отже, на основі всього, що було сказано вище, для розробки ПП було обрано мову програмування JavaScript.

Для розробки вебзастосунку було обрано середовище Visual Studio Code, бо в цьому середовищі розробки можна налаштувати середовище розробки, має вбудований помічник для написання та редагування коду та область навігації для зручного переміщення між файлами проєкту, підтримку IDE здійснюють не тільки розробники, й користувачами, має великий рівень доступності. На основі цих переваг було обрано середовище розробки Visual Studio Code.

Б.5 Виклик і завантаження

Для запуску вебзастосунку необхідно запустити серверну частину за допомогою Visual Studio Code, після чого в браузері в адресному рядку ввести `http://localhost:3000`, після виконаних дій відкривається сторінка авторизації користувачів системи, що представлено на рис. Б.3.

Авторизація

* Login:

* Password: 

Рисунок Б.3 – Вебсторінка авторизації

Б.6 Приклади реалізації

Після авторизації користувач потрапляє на одну з двох вебсторінок «Доктор» або «Пацієнт» в залежності від його ролі в вебзастосунку. Якщо в користувача не має облікового запису, він може створити його натиснувши кнопку «Реєстрація» та зареєструвавшись у системі.

На вебсторінці «Пацієнт» користувач може переглядати короткі відомості про свої дані, що були введені при реєстрації, переглядати основні відомості про виписні документи та відправляти свої виписні документи на свою поштову скриньку. Зовнішній вигляд вебсторінки «Пацієнт» представлено на рис. Б.4.

Добрий день, Sergienko O.V.			
Дата народження: 10.10.2014 Gmail: wlad.scheludko@gmail.com Номер телефону: 0665409080 Ваш сімейний лікар: Matish A. U.			
Номер документа	Діагноз	Дата виписки	Відправити
60a5141af82334439879155c	Kurva	11.05.2021	Відправити документ
60a514581f2d884fb0ef3fa6	Privetiky	12.05.2021	Відправити документ
60a78b905f351c481455ce5d	Privetiky	12.05.2021	Відправити документ

Рисунок Б.4 – Зовнішній вигляд вебсторінки «Пацієнт»

Б.7 Вхідні та вихідні дані

Вхідними даними вебзастосунку є дані, що було введені користувачами в процесі реєстрації, авторизації та заповнення виписних листів. Ці дані отримуються під час дій користувача, що створюють події, які викликають відповідні сценарії в серверній частині.

Вихідними даними є виписні листи, що відправляються на поштову скриньку Gmail користувачів системи.

ДОДАТОК В
Текст програми

B.1 Файл errors.js

```
import axios from 'axios'

const TRIGGER_BUTTON = 'TRIGGER_BUTTON'
const AUTHORISATION_BUTTON = 'AUTHORISATION_BUTTON'
const REGISTRATION_BUTTON = 'REGISTRATION_BUTTON'
const BACK_BUTTON = 'BACK_BUTTON'
const REGISTER_BUTTON = 'REGISTER_BUTTON'
const GET_DOCTOR_TRIGGER = 'GET_DOCTOR_TRIGGER'
const SET_DOCTOR = 'SET_DOCTOR'
const CHECK_DOCTOR = 'CHECK_DOCTOR'
const CHECK_PATIENT = 'CHECK_PATIENT'
const GET_LIST = 'GET_LIST'
const GET_LIST_EMAIL = 'GET_LIST_EMAIL'
const GET_PATIENT_LOGIN = 'GET_PATIENT_LOGIN'
const SET_CREATE = 'SET_CREATE'

const initialState = {
  trigger: 0,
  user: {},
  newUser: {},
  doctorList: [],
  doctor: '',
  lists: [],
  patientList: [],
  patientLogin: ''
}

export const triggerButtonStore = (val) => (dispatch) => {
  try {
    dispatch(({ type: TRIGGER_BUTTON, payload: val }))
  } catch (err) {
    console.log({ err })
  }
}

export const authorisationTrigger = ({login, password, history}) => async (dispatch) => {
  try{
    await axios.post('http://localhost:4000/users/login', {
      login,
      password
    })
    .then(function ( response ) {
      if(response.status === 200){
```

```

    dispatch(({type: AUTHORISATION_BUTTON, payload: { ...response }}))
    switch(response.data.user.role){
      case 'Patient':
        history.push('/patient')
        break
      case 'Doctor':
        history.push('/doctor')
        break
    }
  }
})
.catch(function (error) {
  alert('Wrond login. Please, try again.')
  console.log(error);
});
} catch(err){
  console.log(err)
}
}
export const registerTrigger = ({history}) => (dispatch) => {
  try {
    dispatch(({ type: REGISTER_BUTTON}))
    history.push('/register')
  } catch(err) {
    console.log({ err })
  }
}
export const registrationTrigger = ({login,password,role,fullName,dateOfBirth,gmail,numberTel,bloodeType,expOfWork,history}) => async
(dispatch) => {
  try {
    switch(role){
      case 'Patient':{
        await axios.post('http://localhost:4000/users/register', {
          login,
          password,
          role,
          fullName,
          dateOfBirth,
          gmail,
          numberTel,
          bloodeType

```

```

    })
    .then(( response )=> {
      if(response.status === 200){
        dispatch(({type: REGISTRATION_BUTTON, payload: { ...response }}))
        history.push('/')
      }
    })
    .catch((error) =>{
      alert('Login already exists')
      console.log(error);
    });
    break
  }
  case 'Doctor':{
    await axios.post('http://localhost:4000/users/register', {
      login,
      password,
      role,
      fullName,
      dateOfBirth,
      gmail,
      expOfWork
    })
    .then(( response )=> {
      if(response.status === 200){
        dispatch(({type: REGISTRATION_BUTTON, payload: { ...response }}))
        history.push('/')
      }
    })
    .catch( (error) =>{
      alert('Login already exists')
      console.log(error);
    });
    break
  }
}
} catch(err) {
  console.log({ err })
}
}

export const backTrigger = ({history}) => (dispatch) => {
  try {

```

```

    dispatch(({ type: BACK_BUTTON}))
    history.push('/')
  } catch(err) {
    console.log({ err })
  }
}

export const getDoctorTrigger = ({login}) => async (dispatch)=>{
  try {
    await axios.post(`http://localhost:4000/users/get-all-doctors`, {
      login
    })
    .then(function ({data}) {
      dispatch(({ type: GET_DOCTOR_TRIGGER, payload:data.doctor}))

    })
    .catch(function (error) {
      console.log(error);
    });
  } catch(err) {
    console.log({ err })
  }
}

export const setDoctor = (val) => async ( dispatch,getState)=>{
  try {
    const {errors} = getState()
    console.log(errors.user.data.user.login)
    await axios.post(`http://localhost:4000/users/binding`, {
      loginPatient:errors.user.data.user.login,
      loginDoctor: val
    })
    .then(function (data) {
      if(data.status === 200){
        dispatch(({type: SET_DOCTOR}))
      }
    })
    .catch(function (error) {
      console.log(error);
    });
  } catch(err) {
    console.log({ err })
  }
}

```

```

export const checkDoctor = () => async (dispatch,getState)=>{
  try {
    const {errors} = getState()
    await axios.post(`http://localhost:4000/users/get-assigned-doctor`, {
      login:errors.user.data.user.login
    })
    .then(function ({data}) {
      dispatch(({ type: CHECK_DOCTOR, payload:data.doctor}))
    })
    .catch(function (error) {
      console.log(error);
    });
  } catch(err) {
    console.log({ err })
  }
}

export const checkPatient = () => async (dispatch,getState)=>{
  try {
    const {errors} = getState()
    await axios.post(`http://localhost:4000/users/get-assigned-patient`, {
      login:errors.user.data.user.login
    })
    .then(function ({data}) {
      dispatch(({ type: CHECK_PATIENT, payload:data.patients}))
    })
    .catch(function (error) {
      console.log(error);
    });
  } catch(err) {
    console.log({ err })
  }
}

export const getListPatient = (patientLogin) => async (dispatch)=>{
  try {
    dispatch(({ type: GET_PATIENT_LOGIN, payload:patientLogin}))
    await axios.post(`http://localhost:4000/users/get-list`, {
      login:patientLogin
    })
    .then(function ({data}) {
      // console.log('DATA', data);
      const result = [];
      data.lists.map(arr => {

```

```

        arr.map(_ =>{
            result.push(_)
        })
    });
    // console.log('RESULT', result);
    dispatch(({ type: GET_LIST, payload:result}))
})
.catch(function (error) {
    console.log(error);
});
} catch(err) {
    console.log({ err })
}
}
export const setListDoctor = () => async (dispatch)=>{
    try {
        dispatch(({ type: CHECK_DOCTOR, payload:"}))
    } catch(err) {
        console.log({ err })
    }
}
export const getListEmail = (listId) => async (dispatch,getState)=>{
    try {
        await axios.post(`http://localhost:4000/users/get-list-email`, {
            listId
        })
        .then(function ({data}) {
            if(data.status === 200){
                dispatch(({type: GET_LIST_EMAIL}))
                alert('Лист було відправлено')
            }
        })
        .catch(function (error) {
            console.log(error);
        });
    } catch(err) {
        console.log({ err })
    }
}
export const getList = () => async (dispatch,getState)=>{
    try {
        const {errors} = getState()

```

```

await axios.post(`http://localhost:4000/users/get-list`, {
  login:errors.user.data.user.login
})
.then(function ({data}) {
  // console.log('DATA', data);
  const result = [];

  data.lists.map(arr => {
    arr.map(_ =>{
      result.push(_)
    })
  });
  // console.log('RESULT', result);
  dispatch(({ type: GET_LIST, payload:result}))
})
.catch(function (error) {
  console.log(error);
});
} catch(err) {
  console.log({ err })
}
}

export const setCreate = ({loginPatientModal,
  fullNameModal,
  dateOfBirthModal,
  bloodeTypeModal,
  gmailModal,
  dateOfIssueModal,
  diagnozModal,
  nameDoctorModal}) => async (dispatch)=>{
  try {
    console.log('login ', loginPatientModal)
    console.log('full ', fullNameModal)
    await axios.post(`http://localhost:4000/users/create-list`, {
      loginPatient:loginPatientModal,
      listId:{
        fullName:fullNameModal,
        dateOfBirth:dateOfBirthModal,
        bloodeType:bloodeTypeModal,
        gmail:gmailModal,
        dateOfIssue:dateOfIssueModal,
        diagnoz:diagnozModal,

```

```

        nameDoctor:nameDoctorModal
    }
})
.then(function ({data}) {
    if(data){
        dispatch(({type: SET_CREATE}))
        console.log(data.newBlock)
        alert('List sozdan')
    }
})
.catch(function (error) {
    console.log(error);
});
} catch(err) {
    console.log({ err })
}
}

export default (state = initialState, action) => {
    switch (action.type) {
        case TRIGGER_BUTTON:
            return {
                ...state,
                trigger: action.payload
            }
        case AUTHORISATION_BUTTON:
            return{
                ...state,
                user: action.payload
            }
        case REGISTER_BUTTON:
            return{
                ...state
            }
        case BACK_BUTTON:
            return{
                ...state
            }
        case REGISTRATION_BUTTON:
            return{
                ...state,
                newUser: action.payload
            }
    }
}

```

```

case GET_DOCTOR_TRIGGER:
  return{
    ...state,
    doctorList: action.payload
  }
case SET_DOCTOR:
  return{
    ...state,
    doctor: action.payload
  }
case CHECK_DOCTOR:
  return{
    ...state,
    doctor: action.payload
  }
case GET_LIST:
  return{
    ...state,
    lists: action.payload
  }
case CHECK_PATIENT:
  return{
    ...state,
    patientList: action.payload
  }
case GET_PATIENT_LOGIN:
  return{
    ...state,
    patientLogin: action.payload
  }
default:
  return state
  }
}

export const getTriger = (state) => state.errors.trigger

```

В.2 Файл MainContentComponent/index.js

```

import React, { useCallback, useEffect } from 'react';
import { Layout, Button } from 'antd';
import { connect } from 'react-redux';

```

```

import { useHistory } from "react-router-dom";
import { getTriger, triggerButtonStore, autorisationTrigger, registerTrigger, setListDoctor } from
'../../store/reducers/errors';

const {
  Content
} = Layout
const MainAut = ({ //props parameters
  login,
  password,
  trigger,
  handleTrigger,
  autorisationTrigger,
  registerTrigger,
  setListDoctor,
}) => {
  const history = useHistory()
  useEffect(() => {
    setListDoctor();
  }, [])
  )
  const authHendler = useCallback(
    () => {
      autorisationTrigger(login, password, history)
    }, [login, password, history]
  )
  const registerHendler = useCallback(
    () => {
      registerTrigger(history)
    }, [history]
  )
  return (
    <Content style={{ display:'flex', justifyContent:'space-between', alignItems:'center'}}>
      <Button onClick={registerHendler}>
        Реєстрація
      </Button>
      <Button onClick={authHendler}>
        Авторизація
      </Button>
    </Content>
  )
}
export default connect(

```

```

(state) => ({
  trigger: getTriger(state)
}),
(dispatch) => ({
  handleTrigger: (val) => dispatch(triggerButtonStore(val)),
  autorisationTrigger:(login, password, history) => dispatch(autorisationTrigger({login, password,
history}))),
  registerTrigger:(history) => dispatch(registerTrigger({history})),
  setListDoctor:()=>dispatch(setListDoctor())
})
)(MainAut);

```

B.3 Файл Register/index.js

```

import React, { useCallback } from 'react';
import { Layout, Button } from 'antd';
import { connect } from 'react-redux';
import { useHistory } from "react-router-dom";
import { getTriger, backTrigger, registrationTrigger } from '../store/reducers/errors';
const {
  Content
} = Layout
const MainRegister = ({ //props parameters
  login,
  password,
  role,
  fullName,
  dateOfBirth,
  gmail,
  numberTel,
  bloodeType,
  expOfWork,
  backTrigger,
  registrationTrigger,
}) => {
  const history = useHistory()
  const backHendler = useCallback(
    () => {
      backTrigger(history)
    }, [history]
  )
}

```

```

const registrationHendler = useCallback(
  () => {
    registrationTrigger(login,password,role,
fullName,dateOfBirth,gmail,numberTel,bloodeType,expOfWork,history)
  }, [login,password,role, fullName,dateOfBirth,gmail,numberTel,bloodeType,expOfWork,history]
)
return (
  <Content style={{display:'flex', justifyContent:'space-between', alignItems:'center'}}>
    <Button onClick={backHendler}>
      Назад
    </Button>
    <Button onClick={registrationHendler}>
      Реєстрація
    </Button>
  </Content>
)
}
export default connect(
  (state) => ({
    trigger: getTriger(state)
  }),
  (dispatch) => ({
    backTrigger:(history) => dispatch(backTrigger({history})),
registrationTrigger:(login,password,role,fullName,dateOfBirth,gmail,numberTel,bloodeType,expOfWork,history)
=>
dispatch(registrationTrigger({login,password,role,
fullName,dateOfBirth,gmail,numberTel,bloodeType,expOfWork,history}))
  })
)(MainRegister);

```

В.4 Файл users.js

```

const express = require('express');
const moment = require('moment');
const sha256 = require('js-sha256');
const router = express.Router();
const User = require('../models/User');
const DoctorsPatients = require('../models/Doctors-Patients');
const List = require('../models/List');
const Blockchain = require('../models/Block');
const { registerValidation, patDocValidation, createListValidation } = require('../utils/validation');
const verifyUser = require('../utils/verifyUser');
const ROLES = require('../constants/roles');

```

```

const nodemailer = require('nodemailer');
let mailOptions = {
  host: 'smtp.gmail.com',
  port: 465,
  secure: true,
  auth: {
    user: 'wlad.scheludko@gmail.com',
    pass: 'Wlad1999033',
  },
};
router.post('/register', async (req, res) => {
  const { login } = req.body;
  const { value, error } = registerValidation(req.body);
  if (error) res.status(400).send(error.details[0].message);
  const userNameExist = await User.findOne({ login });
  if (userNameExist) res.status(400).send('Username already exists');
  try {
    const newUser = await User.create(value);
    res.status(200).send({ newUser });
  } catch (err) {
    res.send({ message: err })
  }
});
router.post('/login', async (req, res) => {
  try {
    const { login } = req.body;
    const user = await User.findOne({ login });
    if (!user) res.status(401).send('Wrond login. Please, try again. ');
    // const loginPatient = user.login;
    // const binding = await DoctorsPatients.find({loginPatient});
    const outputUserData = JSON.parse(JSON.stringify(user))
    delete outputUserData.password;
    res.status(200).send({ user: outputUserData })
  } catch (err) {
    res.status(401).send({ message: err });
  }
});
router.post('/get-all-doctors', async (req, res) => {
  const {login} = req.body;
  // console.log(login);
  // const binding = await DoctorsPatients.findOne({loginPatient:login});
  // console.log(binding);

```

```

const loginDoctor = await User.find({role:'Doctor'});
res.send({ doctor: loginDoctor });
});
router.post('/get-assigned-doctor', async (req, res) => {
  const {login} = req.body;
  // console.log(login);
  const binding = await DoctorsPatients.findOne({loginPatient:login});
  // console.log(binding);
  const loginDoctor = await User.findOne({login:binding.loginDoctor});
  // console.log(loginDoctor)
  res.send({ doctor: loginDoctor.fullName });
});
router.post('/get-assigned-patient', async (req, res) => {
  try {
    const {login} = req.body;
    const binding = await DoctorsPatients.find({loginDoctor:login});
    // console.log(binding, '*****')
    await Promise.all(
      binding.map(({ loginPatient }) => User.findOne({ login: loginPatient }))
    ).then(result => res.send({ patients: result }));
  } catch(err) {
    console.log(err)
    res.send(err)
  }
});
router.post('/binding', async (req, res) => {
  try {
    const { loginPatient, loginDoctor } = req.body;
    const {value, error} = patDocValidation(req.body);
    const user = await DoctorsPatients.findOne({ loginPatient });
    if (!user) {
      const newUser = await DoctorsPatients.create(value);
      res.status(200).send('Binding complete');
    }
  } catch (err) {
    res.send({ message: err });
  }
});
router.post('/create-list', async (req, res) => {
  try {
    let blockes = await Blockchain.find();
    if(!blockes.length){

```

```

    const lastHash = 0;
    const index = 0;
    const loginPatient = "GENESIS BLOCK";
    const timeStamp = moment(new Date()).format('YYYY-MM-DD HH:mm:ss');
    const listId = "SECRET_LIST_ID";
    const hash = sha256(index+lastHash+loginPatient+timeStamp+listId);
    console.log(hash)
    const newBlock = await Blockchain.create({index, loginPatient, hash, lastHash, timeStamp, listId});
    res.send({ newBlock });
  }
  const { loginPatient, listId } = req.body;
  const {value, error} = createListValidation(req.body);
  const newList = await List.create(listId);
  blockes = await Blockchain.find().sort({'timeStamp':-1}).limit(1);
  const oneBlock = blockes[0];
  // console.log(oneBlock);
  const lastHash = oneBlock.hash;
  const index = oneBlock.index+1;
  const timeStamp = moment(new Date()).format('YYYY-MM-DD HH:mm:ss');
  const hash = sha256(index+lastHash+loginPatient+timeStamp+newList._id);
  const newBlock = await Blockchain.create({index, loginPatient, hash, lastHash, timeStamp,
listId:newList._id});
  res.send({newBlock});
} catch (err) {
  console.log(err)
  res.send({ message: err });
}
});
router.post('/get-list', async (req, res) => {
  try {
    const {login} = req.body;
    const listBlockes = await Blockchain.find({loginPatient:login});
    console.log(listBlockes)
    await Promise.all(
      listBlockes.map(({ listId }) => List.find({ _id: listId }))
    ).then(result => res.send({ lists: result }));
  } catch(err) {
    console.log(err)
    res.send(err)
  }
});
router.post('/get-list-email', async (req, res) => {

```

```

try {
  const {listId} = req.body;
  const list = await List.findOne({_id:listId});
  const gmail = list.gmail;
  console.log(list);
  console.log(gmail);
  let transport = nodemailer.createTransport(mailOptions);
  let Options = {
    from: 'wlad.scheludko@gmail.com',
    to: gmail,
    subject: `Выписной документ`,
    html: `<!DOCTYPE html>
    <html lang="ru">
      <head>
        <meta charset="UTF-8"> <title>Title</title>
      </head>
      <body>
        <div style="border: 1px solid black; background-color: #bed4aa; padding: 15px; width: 400px;">
          <div style="text-align: center; margin-bottom: 15px; font-size: 18px; font-weight: bold;">
            Лист пацієнта ${list.fullName}
          </div>
          <div style="margin: 0 15px;">
            <div style="margin: 5px 0;">
              Дата народження: ${list.dateOfBirth}
            </div>
            <div style="margin: 5px 0;">
              Лікар: ${list.nameDoctor}
            </div>
            <div style="margin: 5px 0;">
              Діагноз: ${list.diagnos}
            </div>
            <div style="margin: 5px 0;">
              Дата виписки: ${list.dateOfIssue}
            </div>
          </div>
        </body>
      </html>`,
  };
  let result = transport.sendMail(Options, function(error, response) {
    if (error) {
      console.log(error);
    }
  });
}

```

```
    } else {  
      console.log('Message sent: OK');  
      res.status(200)  
    }  
  });  
} catch(err) {  
  console.log(err)  
  res.send(err)  
}  
});  
module.exports = router;
```

ДОДАТОК Д
Слайди презентації

1

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

**РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ ДЛЯ ПІДТРИМКИ
ЕЛЕКТРОННОЇ КАРТКИ ПАЦІЄНТА НА ОСНОВІ БЛОКЧЕЙНУ**

Виконав:
Студент групи КНТ-118сп
В.А. Шелудько

Керівник:
доцент
О.В. Шитікова

2021

Рисунок Д.1 – Слайд № 1

2

Мета, об'єкт, предмет дослідження

- Об'єкт дослідження – вебзастосунок для підтримання роботи електронної картки пацієнта.
- Предмет дослідження – методи підтримки електронної картки пацієнта на основі блокчейну.
- Мета роботи – програмна реалізація методів для підтримання електронної картки пацієнта на основі блокчейну.
- Актуальність та проблематика роботи полягає в тому, що останнім часом існує багато медичних інформаційних систем, що не завжди задовольняють вимоги користувачів систем та їх потреби.

Рисунок Д.2 – Слайд № 2

Постановка завдання до роботи

Для досягнення поставленої мети необхідно виконати наступні завдання:

- – реалізація процесу реєстрації та авторизації користувачів в системі;
- – надати можливість пацієнту обирати лікаря для подальшого отримання виписних документів;
- – надати можливість пацієнту переглядати свої документи;
- – налаштувати систему для перегляду даних пацієнтів та створення виписних документів;
- – реалізувати алгоритм захисту даних про документи пацієнта на основі технології блокчейн.

Рисунок Д.3 – Слайд № 3

Огляд існуючих мов програмування

Критерій порівняння	PHP	Python	JavaScript
Мова створювалася спеціально для веб-розробки	+	–	+
Кросплатформеність	+	+	+
Швидка інтерпретація коду	+	–	++
Наявність зрозумілої та повної документації	+	+	+
Простота вивчення та розробки	+	++	–
Продуктивність роботи	++	–	+
Наявність бібліотек та фреймворків з повною документацією	–	+	+
Простота вивчення та розробки	+	++	–
Наявність великої кількості методів обробки даних	+	++	+

Рисунок Д.4 – Слайд № 4

Огляд існуючих середовищ розробки

Критерії порівняння	Visual Studio Code	WebStorm
Кросплатформеність	+	+
Доступність	±	++
Наявність автодоповнення	+	+
Простота встановлення	+	+
Підтримка рефакторингу коду	+	+
Можливість налаштування плагінів	–	+
Зручний користувацький інтерфейс	±	++
Підтримка аналізу коду	+	+
Наявність автоматичного закриття дужок, брекєтів	+	+
Наявність інтерактивної консолі	+	+
Швидкість модернізації	+	++
Інтеграція сервісу GitHub	+	+

Рисунок Д.5 – Слайд №5

ER-діаграми БД

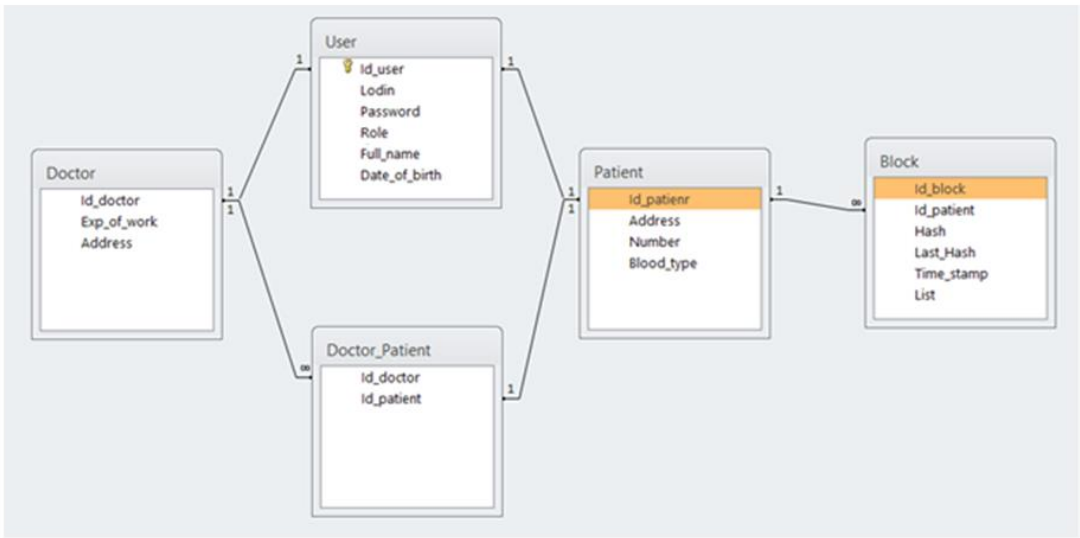


Рисунок Д.6 – Слайд № 6

Схема структури вебзастосунку

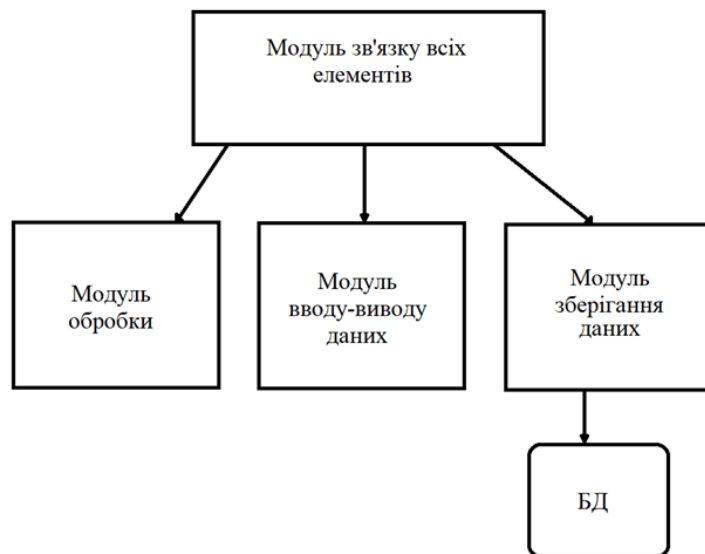


Рисунок Д.7 – Слайд № 7

Вебсторінки

Авторизація

* Login:

* Password: 

Рисунок Д.8 – Слайд № 8

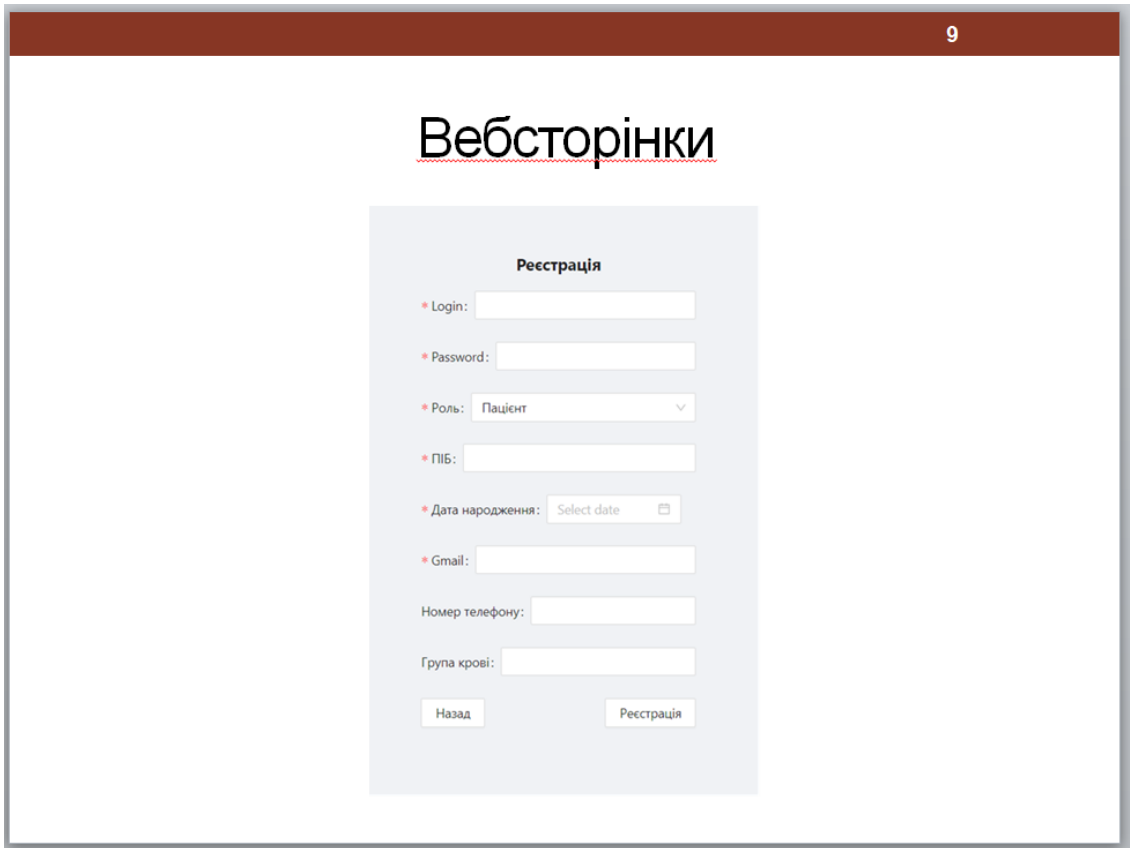


Рисунок Д.9 – Слайд № 9

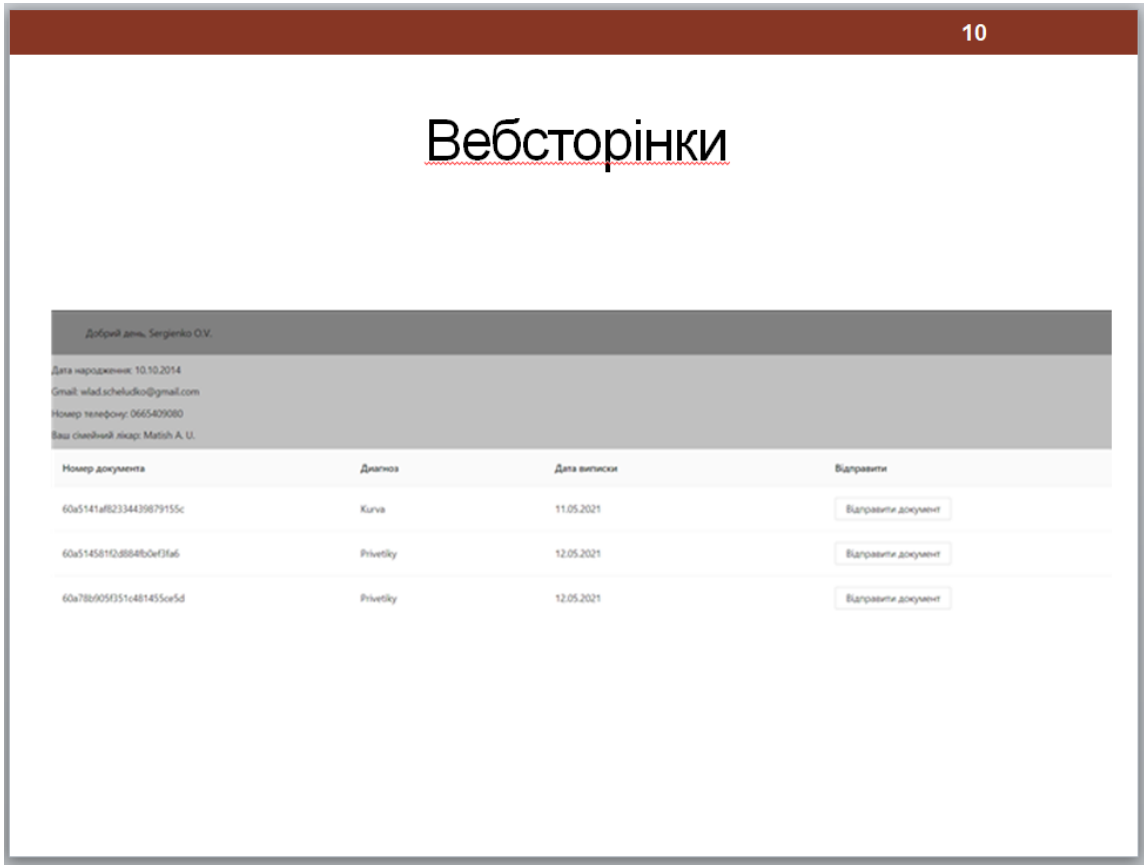


Рисунок Д.10 – Слайд № 10

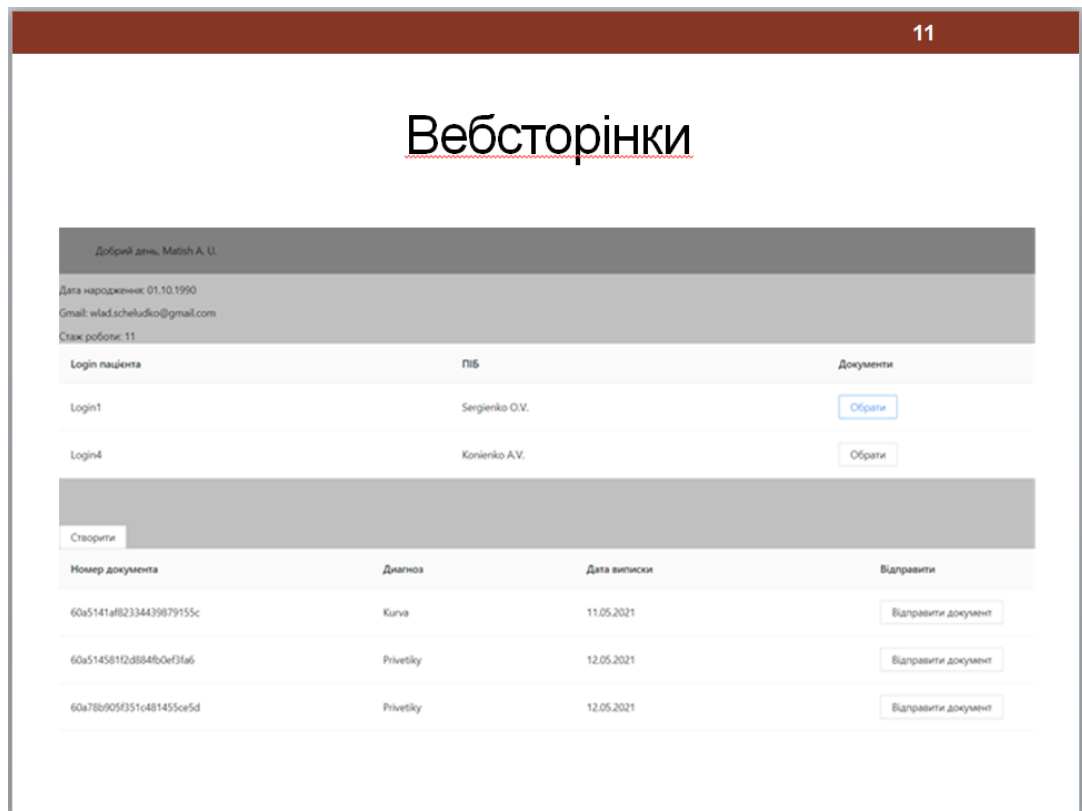


Рисунок Д.11 – Слайд № 11

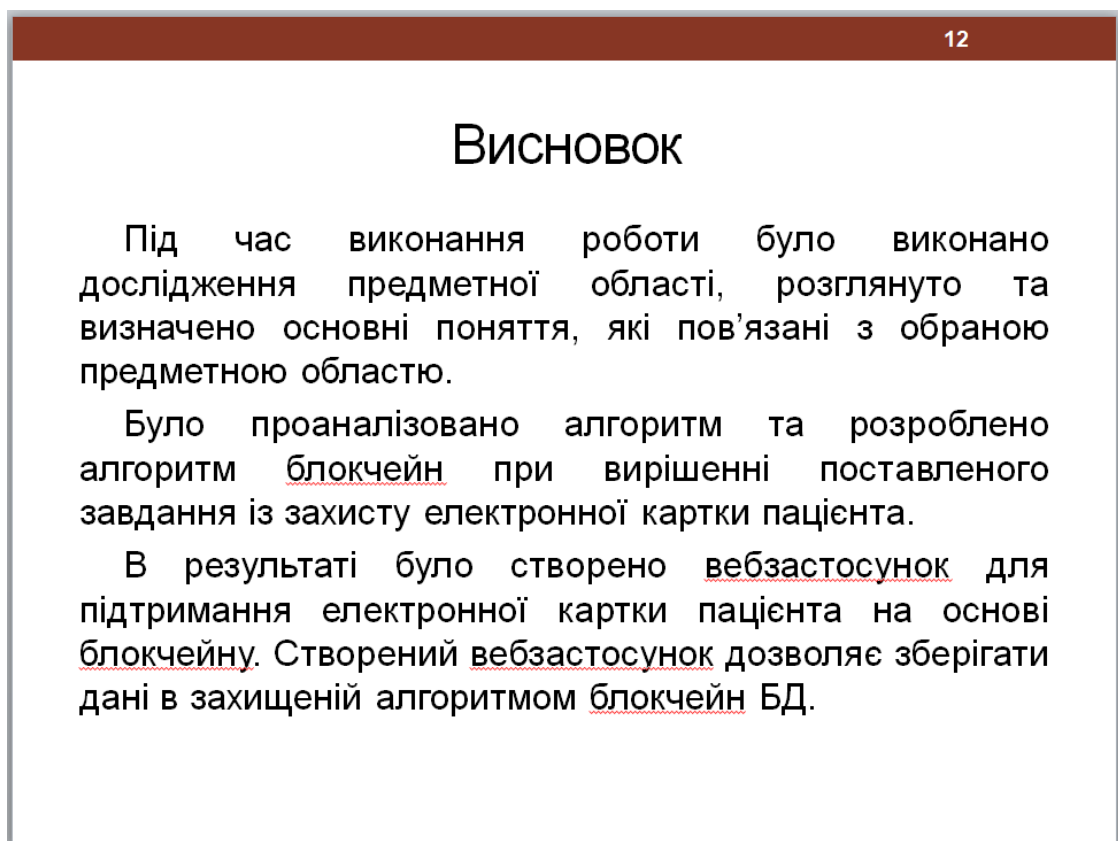


Рисунок Д.12 – Слайд № 12