МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ПРОГРАМУВАННЯ ТА МАТЕМАТИКИ

# Пояснювальна записка

**до дипломної роботи**

**_____ бакалавр _____**

(освітньо-кваліфікаційний рівень)

**на тему «Інтерактивни веб-сервіс для медичного закладу»**

Виконав: ст.групи ІТ-151
напряму підготовки 6.040302 «Інформатика»
_____Омельченко А.В.
(підпис)

Керівник,
к.ф.-м.н._____ Ковальов Ю.Г.
(підпис)

Рецензент,
_____ Батурін О.І.
(підпис)

СЄВЄРОДОНЕЦЬК
2019 року

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет інформаційних технологій та електроніки
Кафедра програмування та математики
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність **6.040302 «Інформатика»**

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМ,
д.т.н., доцент
_____Лифар В.О.
«___» _____ 2019 р.

# З А В Д А Н Н Я
## НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ
## ПАПІРНИЙ ДМИТРО МИКОЛАЙОВИЧ

1. Тема роботи Інтерактивни веб-сервіс для медичного закладу
керівник роботи доцент Ковальов Юрій Григорійович
2. Строк подання студентом роботи 06 червня 2019 р.
3. Вихідні дані до роботи

Об'єктом дослідження даної роботи є програмне забезпечення для реєстрації пацієнтів на прийом до лікаря та ведення історії результатів аналізів та діагнозів.

3.1 Літературні джерела:

Дейт К . Введение в системы баз даных /К.Дейт// – К. ; М. ; СПб.: Изд.дом “Вильямс”. – 2000. –560 с.

Квентин Зервас. Web 2.0: создание приложений на PHP = Practical Web 2.0 Applications with PHP. —М.:«Вильямс», 2009. — С. 544.

Костарев А. Ф. PHP 5. — Спб.: «БХВ-Петербург», 2008. — С. 1104.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1 Вступ
4.2 Аналіз предметної області
Коротка характеристика об’єкту управління.
Опис предметної області
Специфікація вимог до програмного продукту
4.3 Основна частина, в якої висвітлити:
Розробка архітектури програмної системи
Проектування структури бази даних.
4.4 Програмна реалізація
4.5 Тестування та дослідна експлуатація
4.6 Висновки
4.7 Перелік використаних джерел

5. Перелік графічного матеріалу немає
6. Дата видачі завдання 02 лютого 2019 року.

# КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Одержання завдання на виконання роботи | 01.02.19 | |
| 2 | Укладання і погодження з керівником плану і етапів виконання роботи | 20.02.19 | |
| 3 | Узагальнення даних літературних джерел, укладання першого розділу | 1.03.19 | |
| 4 | Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху | 10.03.19 | |
| 5 | Проектування інфологічної моделі задачі що реалізується. | 01.04.19 | |
| 6 | Укладання та тестування програмного продукту | 20.04.19 | |
| 7 | Укладання, оформлення та погодження пояснювальної записки з керівником | 15.05.19 | |
| 8 | Здача готової пояснювальної записки на кафедру | 06.06.19 | |
| 9 | Укладання доповіді і презентації | 10.06.19 | |

Студент _____ Омельченко А.В.
(підпис)

Керівник роботи _____ Ковальов Ю.Г.
(підпис)

ЛИСТ ПОГОДЖЕННЯ І ОЦІНЮВАННЯ

дипломної роботи студента гр. ІТ-151 Омельченко А.В..

Науковий керівник

Доцент, к.ф.-м..н. _____ Ковальов Ю.Г.

Оцінка наукового керівника: _____

Рецензент ст..викл. каф. ПМ СНУ ім.В.Даля Батурін О.І.
місто роботи, посада, ПІБ

Оцінка рецензента: _____

Кінцева оцінка за результатами захисту:
_____

Голова ЕК

_____ Лифар В.О.
підпис

# РЕФЕРАТ

Текст – 65., рис. – 6, табл. – 6, додатків – 1, літературних джерел – 22

Метою роботи є створення веб-додатка для медичного закладу (або закладів) для реєстрації пацієнтів на прийом до лікаря, тобто електронна реєстратура, та ведення історії сдаваних пацієнтом аналізів та поставлених йому діагнозів. Такий програмний продукт набагато пришвидшить та спростить процес запису на прийом до лікаря. Метою роботи є дослідження та детальний аналіз процесу запису пацієнтів на прийом до лікаря в медичних закладах, а також аналіз існуючих програмних аналогів, які надають можливість користувачу записатися на прийом до лікаря, використовуючи графічний інтерфейст веб-сайту. Розробка власного програмного продукту, який би надавав можливість реєструватися на прийом в медичних закладах, з врахуванням переваг та недоліків проаналізованих продуктів-аналогів.

ВЕБ-ДОДАТОК, САЙТ, ВЕБ-СЕРВІС, РЕЄСТРАЦІЯ

# ЗМІСТ

# ВСТУП

Актуальність. В людей часто виникають проблеми зі здоров'ям і в зв'язку з цим виникає потреба в отриманні допомоги від працівників сфери медицини. Тому організація процесу запису пацієнта на прийом до лікаря завжди буде актуальною задачею. На даний момент велика кількість медичних закладів мають власні веб-сайти, але там не має реалізованого функціоналу для запису на прийом. Великим недоліком є те, що пацієнт повинен записуватися на прийом через реєстратуру, а це завжди займає багато часу. Тому розробка веб-сервісу, який надаватиме можливість користувачу записуватися на прийом до потрібного йому лікаря в обраному медичному закладі є досить актуальним завданням. Такий програмний продукт набагато пришвидшить та спростить процес запису на прийом до лікаря. Метою роботи є дослідження та детальний аналіз процесу запису пацієнтів на прийом до лікаря в медичних закладах, а також аналіз існуючих програмних аналогів, які надають можливість користувачу записатися на прийом до лікаря, використовуючи графічний інтерфейст веб-сайту. Розробка власного програмного продукту, який би надавав можливість реєструватися на прийом в медичних закладах, з врахуванням переваг та недоліків проаналізованих продуктів-аналогів.

Об'єкт дослідження – програмне забезпечення медичних систем.

Предмет дослідження – веб-додаток для реєстрації пацієнтів на прийом до лікаря в обраному медичному закладі та для підтримки комплексного моніторингу стану здоров'я населення

Мета дослідження: розробка веб-додатку для онлайн реєстрації пацієнтів до лікаря та для моніторингу результатів аналізів.

Задачі дослідження:

1. дослідити та проаналізувати веб-сайти, які надають користувачам можливість запису на прийом до лікаря, та які надають можливість співробітникам клініки зберігати та аналізувати результати аналізів, діагнози.

2. на основі проведеного аналізу розробити специфікацію функціональних та нефункціональних вимог до майбутнього програмного продукту;

3. розробити архітектуру програмної системи та спроектувати структуру бази даних;

4. зробити вибір мов програмування та технологій для програмної реалізації описаного продукту;

5. розробити веб-сервіс згідно проведених робіт, перерахованих в пунктах 1-4.

# РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1. Коротка характеристика об’єкту управління

У будь-якій поліклініці наявний відділ реєстратури. Там же починається «знайомство» з чергами: за талонами, за карточками (якщо їх не розносять централізовано), за консультаціями типу: "Що, де і коли?". Регулювання потоку пацієнтів у конкретній поліклініці можна здійснювати через використання талонної системи, попереднього запису по телефону, в реєстратурі чи у журналах само запису тощо. При застосуванні талонної системи черга за ними в деяких поліклініках утворюється за годину до їх відкриття. Пізніше людей може бути значно менше, але талонів може не бути. Тоді до лікаря можна й не потрапити, якщо немає гострого болю чи високої температури.

Працівник реєстратури має наступні обов’язки [2]: · пояснювати місцезнаходження конкретних лікарів; · повідомляти пацієнтам графік роботи того чи іншого спеціаліста; організовувати позачергове надання медичної допомоги пацієнтам з високою температурою і гострим болем; · забезпечувати виписку талонів на прийом до лікаря; · забезпечувати рух амбулаторних карт.

Проте основним завданням працівника реєстратури поліклініки є запис пацієнта на прийом до лікаря та видача йому талона з даними про прийом. Зараз у більшості поліклінік запис пацієнта на прийом до лікаря відбувається безпосередньо у відділі реєстратури, де працівник реєстратури записує в талон прийому особисті дані пацієнта, такі як: прізвище, ім’я, по-батькові, адреса проживання та дані про прийом, а саме: дату, час, місце прийому, спеціальність та прізвище, ім’я, по-батькові лікаря. Після цього пацієнту видається заповнений паперовий талон на прийом [13].

Проведемо аналіз переваг та недоліків використання талонної системи у поліклініках (див. таблицю 1.1).

Таблиця 1.1

Аналіз переваг та недоліків талонної системи у поліклініках

| Функція | Недолік | Перевага |
|---|---|---|
| Запис на прийом по телефону | Працівник реєстратури може неправильно записати дані пацієнта | Пацієнт може записатися на прийом не виходячи з дому |
| Запис на прийом в реєстратурі поліклініки | Потрібно стояти у черзі для отримання талону | Працівник реєстратури може надати різного роду інформацію, яка цікавить пацієнта |
| Запис на прийом у журналі самозапису | Оскільки ведення журналу самозапису відбувається в паперовому вигляді, то можливі дублювання та втрата даних | Пацієнт може записатися на прийом самостійно, без допомоги працівника реєстратури |

На сьогоднішній день практично всі медичні заклади мають власні веб-сайти, де пацієнти можуть знайти потрібну їм інформацію. Крім того, деякі з них реалізують функцію реєстрації на прийом до лікаря. Метою цієї дипломної роботи є розробка веб-сайту, на якому будуть зареєстровані різнотипні медичні заклади. Такий веб-сайт забезпечить для широкого кола користувачів можливість переглядати інформацію про будь-який медичний заклад, доданий в систему (загальна інформація про медичний заклад, список лікарів закладу, графіки роботи лікарів), з подальшим записом на прийом до обраного спеціаліста.

Розробка цієї системи та її використання надасть такі переваги користувачам:

- велика кількість медичних закладів зібрана в одному місці, користувач матиме можливість вибрати потрібний йому;
- відсутність черг в реєстратурі для отримання талона на прийом до лікаря;
- відсутність черг біля кабінетів лікарів, оскільки в талоні будуть вказані дата та час прийому;
- збереження електронної історії направлень пацієнта;
- електронний графік прийому лікарів;
- електронний запис на прийом відбувається набагато швидше ніж видача талона працівниками реєстратури;
- якість паперового талону може зіпсуватися або його можна втратити, на відміну від електронного.

Користувач розроблюваної системи зможе зареєструватися на прийом до потрібного йому спеціаліста не виходячи з дому, що є дуже зручно і не вимагає значних затрат часу.

### 1.2. Опис предметної області

Для представлення роботи інтерактивного веб-сервісу виділено наступні бізнес-процеси (рисунок 1.1): реєстрація користувача в системі; управління даними медичних закладів; реєстрація на прийом до лікаря. За управління даними медичних закладів в системі відповідатиме адміністратор, а реєструватися в системі і на прийом до лікаря буде сам користувач.

Рис. 1.1. Діаграма бізнес-процесів розроблюваного програмного продукту

Розглянемо детальніше кожен бізнес-процес з представленого списку (рисунок 1.1). На рисунку 1.2 зображено діаграму функцій процесу реєстрації користувача в системі.


Рис. 1.2. Діаграма функцій процесу реєстрації користувача

Для того, щоб користувач мав можливість зареєструватися на прийом до лікаря, він повинен себе ідентифікувати в системі, пройшовши процес реєстрації та авторизації. Для цього потрібно перейти на сторінку реєстрації, заповнити поля реєстраційної форми (ім’я користувача, електронна пошта, пароль) та авторизуватися, використовуючи ім’я користувача та пароль, вказані при реєстрації. Характеристику бізнес-процесу реєстрації користувача в системі наведено в таблиці 1.2.

Таблиця 1.2

Характеристика бізнес-процесу реєстрації користувача в системі

| Назва характеристики | Значення характеристики |
|---|---|
| Ім'я бізнес-процесу | Реєстрація користувача в системі |
| Основні учасники | Клієнт |
| Вхідна подія | Перехід на сторінку реєстрації |
| Вхідні документи | Дані з реєстраційної форми |

| | |
|---|---|
| Вихідна подія | Авторизація на сайті |
| Вихідні документи | Лист з повідомленням про реєстрацію |
| Клієнт бізнес-процесу | Користувач |

На рисунку 1.3 зображено діаграму функцій процесу реєстрації медичного закладу в системі.


Рис. 1.3. Діаграма функцій процесу управління даними медичних закладів

Функції, які відносяться до управління даними медичних закладів, доступні тільки адміністратору через адміністративну панель сайту: при додаванні нового медичного закладу адміністратор повинен вказати наступну інформацію: назву, адресу, контактні дані, також додати всіх лікарів даного медичного закладу з графіками робіт; при редагуванні, адміністратор повинен обрати зі списку доданих медичний заклад, внести потрібні зміни та зберегти їх: · для видалення адміністратор повинен обрати зі списку доданих медичний заклад та натиснути кнопку “видалити”. Характеристику бізнес-процесу реєстрації користувача в системі наведено в таблиці 1.3.

Таблиця 1.3

Характеристика бізнес-процесу управління даними медичних закладів

| Назва характеристики | Значення характеристики |
|---|---|
| Ім'я бізнес-процесу | Управління даними медичних закладів |
| Основні учасники | Адміністратор |
| Вхідна подія | Перехід на сторінку управління даними медичних закладів в адмінпанелі сайту |
| Вхідні документи | Дані з форми додавання закладу |
| Вихідна подія | Здійснені операції над даними медичних закладів |
| Вихідні документи | - |
| Клієнт бізнес-процесу | Адміністратор |

На рисунку 1.4 зображено діаграму функцій процесу реєстрації на прийом до лікаря.


Рис. 1.4. Діаграма функцій процесу реєстрації на прийом

Користувач переходить на сторінку реєстрації на прийом, обирає потрібний йому медичний заклад, після чого перед користувачем з’явиться список спеціальностей лікарів даного закладу. Після обрання спеціальності користувачу буде показано список лікарів обраної спеціальності з подальшим

вибором конкретної особи лікаря. Після цього користувач обирає зручну йому дату прийому. На наступному кроці користувач заповнює форму де потрібно заповнити такі поля: «Прізвище», «Ім’я», «По-батькові», «Рік, місяць та день народження», «Населений пункт», «Вулиця», «Будинок», «Квартира». Після заповнення останньої форми буде сформовано талон на прийом до лікаря.

Характеристику бізнес-процесу реєстрації користувача в системі наведено в таблиці 1.4.

Таблиця 1.4

Характеристика бізнес-процесу реєстрації на прийом до лікаря

| Назва характеристики | Значення характеристики |
|---|---|
| Ім'я бізнес-процесу | Реєстрація на прийом до лікаря |
| Основні учасники | Користувач |
| Вхідна подія | Перехід на сторінку реєстрації на прийом |
| Вхідні документи | Дані введені користувачем |
| Вихідна подія | Пацієнт зареєстрований на прийом до лікаря |
| Вихідні документи | Електронний талон на прийом |
| Клієнт бізнес-процесу | Користувач |

## 1.3. Огляд та аналіз існуючих аналогів

Для аналізу функціональності та інтерфейсу було обрано наступні програмні продукти:

“Інтернет-реєстратура – запис на прийом до лікаря”, режим доступу: (http://iregistratura.ru/);

“Інтернет-портал охорони здоров’я ”, режим доступу: (https://www.zdrav29.ru/);

“Веб-реєстратура”, режим доступу: (http://web-registratura.ru/).

«Інтернет-реєстратура – запис на прийом до лікаря» - єдина система інтернет-запису, що дозволяє кожному застрахованому жителю Росії записатися на прийом до лікаря в медичні установи свого міста, якщо вони додані в систему. Через сайт системи “Інтернет-реєстратура” користувач може: записатися на прийом до лікаря, викликати лікаря додому, переглянути графік роботи лікарів, знайти потрібний медичний заклад, переглянути залишки пільгових ліків.

На головній сторінці системи “Інтернет-реєстратура” відображається форма запису на прийом до лікаря Для того, щоб записатися на прийом до лікаря, на головній сторінці системи “Інтернет-реєстратура” розміщено випадаючий список з переліком регіонів, з якого користувач повинен обрати свій. Нижче знаходиться карта, на якій показані населені пункти, в медичних закладах яких можна записатися на прийом.

Після вибору конкретного регіону зі списку, на карті залишаться тільки населені пункти обраного регіону. Після цього користувач повинен обрати свій населений пункт, це можна зробити клацнувши мишкою на ньому або обрати з спадного списку під картою і підтвердити свій вибір натиснувши кнопку “Продовжити”. Після підтвердження вибору населеного пункту користувач буде направлений на сторінку вибору медичного закладу. Медичний заклад також обирається із випадаючого списку Після підтвердження вибору медичного закладу користувач заповнює форму, де необхідно вказати особисті дані. Форма для заповнення особистих даних Вказавши особисті дані, користувачу залишається обрати потрібного йому спеціаліста, дату та час прийому.

На головній сторінці розміщено навігаційне меню з посиланнями на такі розділи: “Реєстратура”, “Скасування запису”, “Інформація”, “Зворотній

зв’язок”, “Довідник”, “Інструкція”. Для того, щоб записатися на прийом до лікаря користувачу потрібно виконати декілька кроків: перейти в розділ “Реєстратура”; вибрати зі списку населений пункт; вибрати заклад охорони здоров'я і (якщо є) його підрозділ; вибрати фахівця; вибрати день; вибрати зручний час; вказати свої контактні дані; переконатися в коректності введених даних, при необхідності повернувшись на потрібний крок.

Після виконання всіх кроків відбудеться перехід на сторінку, вмістом якої буде сформований талон запису на прийом “Веб-реєстратура” – веб-сайт, використовуючи який, користувач може записатися на прийом до лікаря. Запис проводиться пацієнтом самостійно, без участі медичних працівників, через веб-сайт. На даному сайті організовано централізований ресурс, звідки можна зробити запис до лікаря в будь-який медичний заклад. У свою чергу, кожен медичний заклад має можливість розмістити форму запису на прийом на своєму сайті. Зайшовши на головну сторінку сайту серед всього вмісту сторінки виділені жирним, підкресленим текстом посилання на сторінку з графіком роботи лікарів та посилання на сторінку для запису на прийом.

Після проведення аналізу та порівняння систем схожих за функціоналом з розроблюваною, було виділено такі переваги: наповнення веб-сайту максимально корисною інформацією; зручний пошук потрібного медичного закладу; швидке проходження процесу реєстрації пацієнта на прийом; наявність інструкції з користування функціоналом сервісу.

Також в процесі проектування розроблюваної системи, слід уникати таких недоліків як надлишковість інформації та нагромадженість користувацького інтерфейсу зайвим функціоналом, який буде відволікати користувача від основної задачі сайту.

### 1.4. Специфікація вимог до програмного продукту

Глосарій проекту – це словник, в якому є головні терміни та поняття даної предметної області. Глосарій проекту знаходиться у додатку А. Для

візуального зображення можливостей ПЗ використовується діаграма варіантів використання системи (рисунок 1.13) [1].


Рис.1.13. Діаграма варіантів використання

Опис варіантів використання поданий у таблицях 1.6-1.13. Функція “Реєстрація в системі” вимагає від користувача системи пройти процедуру реєстрації на сайті, заповнивши поля реєстраційної форми. Процес реєстрації включає в себе заповнення обов’язкових полів форми, таких як: «Ім’я користувача», «Електронна пошта», «Пароль».

Таблиця 1.6

Варіант використання «Реєстрація в системі»

| | |
|---|---|
| Контекст використання | Реєстрація в системі |
| Дійові особи | Користувач |
| Передумова | - |
| Тригер | Відкрита сторінка реєстрації в системі |
| Сценарій | 1. Перехід на сторінку реєстрації. 2. |

| | |
|---|---|
| | Заповнення полів форми. 3. Підтвердження. |
| Пост-умова | |

Функція “Авторизація користувача в системі”. Для того щоб зайти в свій аккаунт на сайті користувач повинен заповнити форму авторизації, а саме два її поля: «Ім’я користувача», «Пароль». Після цього авторизований користувач буде мати доступ до функції «Перегляд попередніх талонів пацієнта».

Таблиця 1.7

Варіант використання «Авторизація»

| | |
|---|---|
| Контекст використання | Авторизація в системі |
| Дійові особи | Користувач, адміністратор |
| Передумова | Реєстрація в системі |
| Тригер | Відкрита сторінка авторизації |
| Сценарій | 1. Перехід на сторінку авторизації. 2. Заповнення полів форми. 3. Підтвердження. |
| Пост-умова | Авторизований користувач |

На сторінці авторизації знаходиться форма введення авторизаційних даних, яка складається з двох полів (ім'я користувача, пароль) та кнопки підтвердження. Розкадровка сторінки з формою авторизації показана на рисунку 1.14.

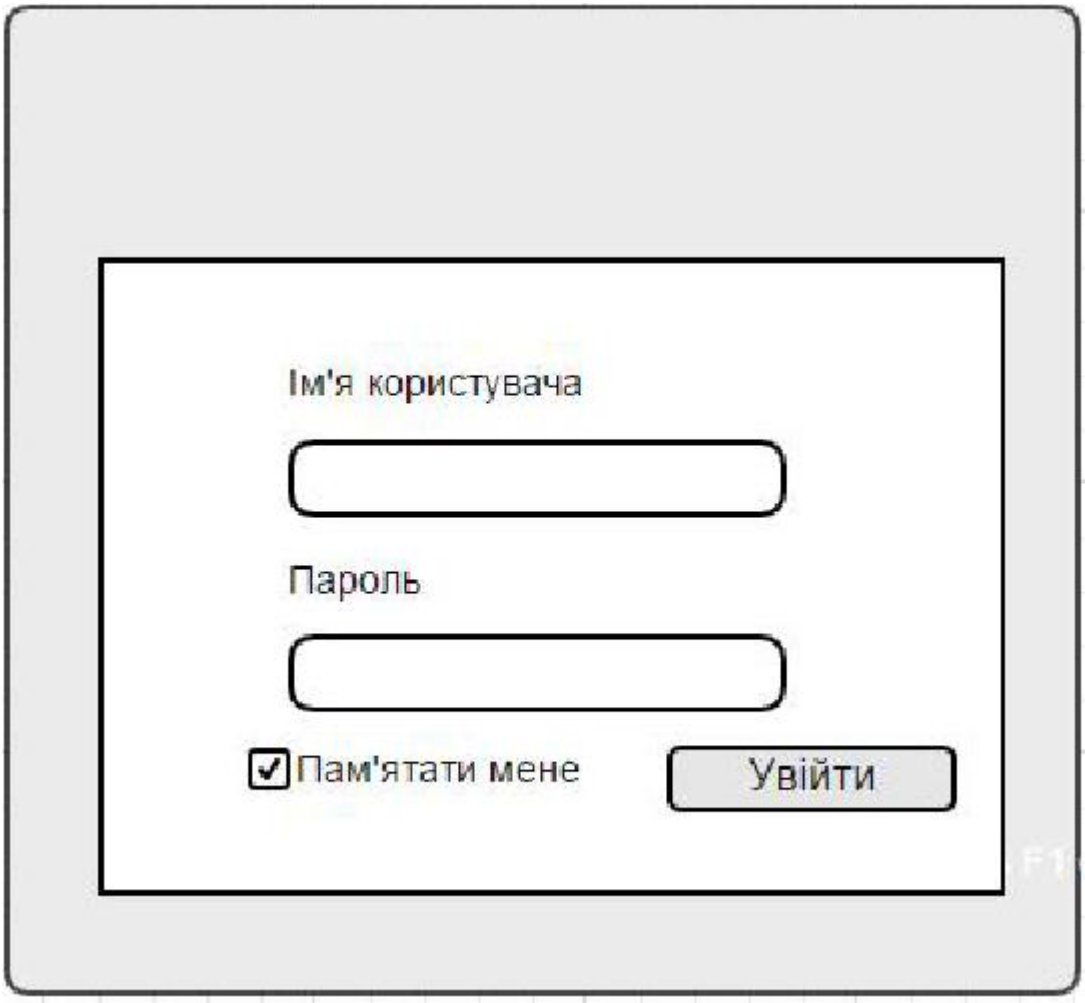


Рис. 1.14. Розкадровка сторінки авторизації

Функція “Перегляд списку медичних закладів” дає можливість користувачу переглянути список медичних закладів наявних в системі. На сторінці буде знаходитися список з назвами медичних закладів. Обравши якийсь з них користувач зможе переглянути детальну інформацію про медичний заклад та почати реєстрацію на прийом до лікаря обраного закладу.

Таблиця 1.8

Варіант використання «Перегляд списку медичних закладів»

| | |
|---|---|
| Контекст використання | Перегляд списку медичних закладів |
| Дійові особи | Користувач |
| Передумова | Авторизація в системі |

| | |
|---|---|
| Тригер | Відкрита головна сторінка сайту |
| Сценарій | 1. Перехід на головну сторінку сайту |
| Пост-умова | - |

Розкадровка сторінки зі списком медичних закладів зображена на рисунку 1.15.

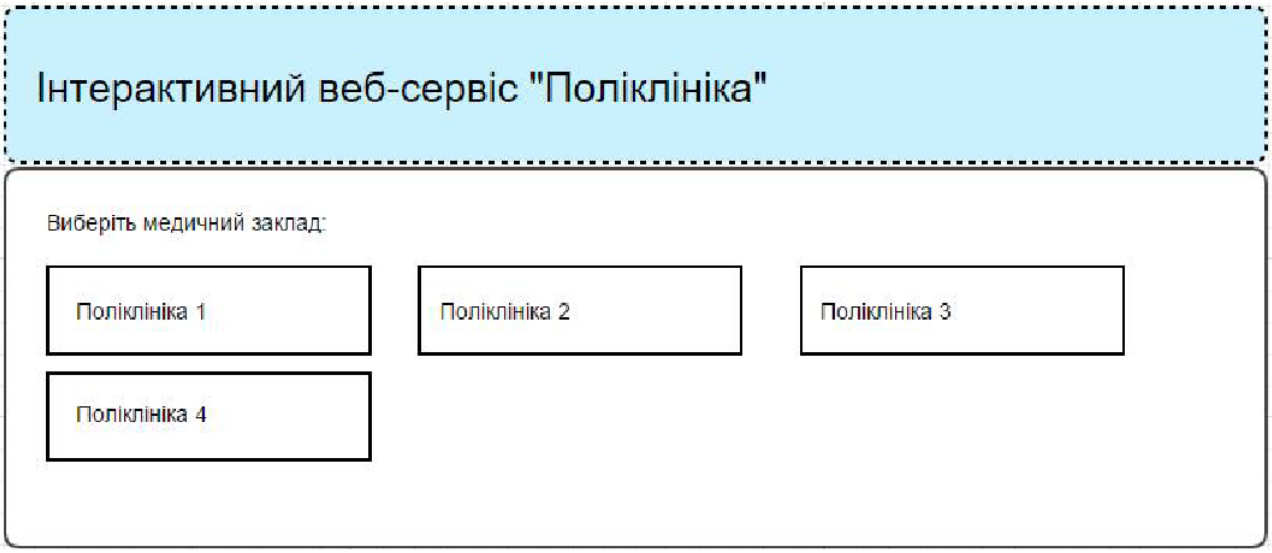


Рис. 1.15. Розкадровка сторінки зі списком медичних закладів

Адміністратор матиме можливість переглядати список медичних закладів в адміністративній панелі сайту.

Таблиця 1.9

Варіант використання «Перегляд списку медичних закладів»

| | |
|---|---|
| Контекст використання | Перегляд списку медичних закладів |
| Дійові особи | Адміністратор |
| Передумова | Авторизація в системі |
| Тригер | Відкрита сторінка зі списком медичних закладів в адмінпанелі сайту |

| | |
|---|---|
| Сценарій | 1. Перехід в адмінпанель сайту. 2. Вибір посилання на сторінку зі списком медичних закладів. |
| Пост-умова | - |

Функція “Реєстрація на прийом до лікаря” дає змогу користувачу записатися на прийом до лікаря обравши потрібний йому медичний заклад, після чого перед користувачем з’явиться список спеціальностей лікарів даного закладу. Після обрання спеціальності користувачу буде показано список лікарів обраної спеціальності з подальшим вибором конкретної особи лікаря. На наступному кроці користувач заповнює форму, де потрібно заповнити такі поля: «Прізвище», «Ім’я», «По-батькові», «Рік, місяць та день народження», «Населений пункт», «Вулиця», «Будинок», «Квартира». Після заповнення останньої форми буде сформовано талон на прийом до лікаря з вказаною інформацією.

Таблиця 1.10

Варіант використання «Реєстрація на прийом до лікаря»

| | |
|---|---|
| Контекст використання | Реєстрація на прийом до лікаря |
| Дійові особи | Користувач |
| Передумова | Авторизація в системі |
| Тригер | Відкрита сторінка реєстрації на прийом |
| Сценарій | 1. Перехід на сторінку реєстрації на прийом. 2. Вибір медичного закладу. 3. Вибір спеціальності лікаря. 4. Вибір особи лікаря. 5. Вибір дати і часу прийому. 6. Введення особистих даних 7. Підтвердження. |

| | |
|---|---|
| Пост-умова | - |

В частині контенту на сторінці зі списком лікарів медичного закладу знаходиться кнопка повернення на попередню сторінку, назва кроку реєстрації на прийом там список спеціальностей лікарів. Розкадровка сторінки зі списком лікарів показана на рисунку 1.16.

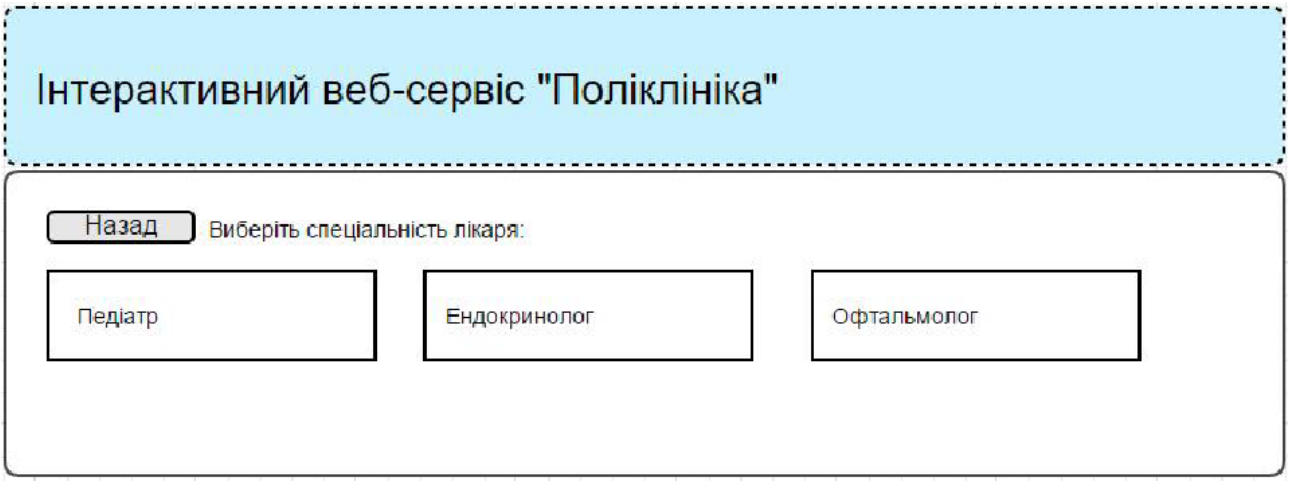


Рис. 1.16. Розкадровка сторінки зі списком спеціальностей лікарів

На сторінках вибору дати та часу прийому в частині контенту перелічені дати та час прийому відповідно. Сірим кольором позначені дати та час, які вже минули. Записи з білим фоном та чорним текстом означають дати та час, вільні для реєстрації на прийом. Розкадровка сторінок вибору дати та часу прийому показана на рисунках 1.17 та 1.18.

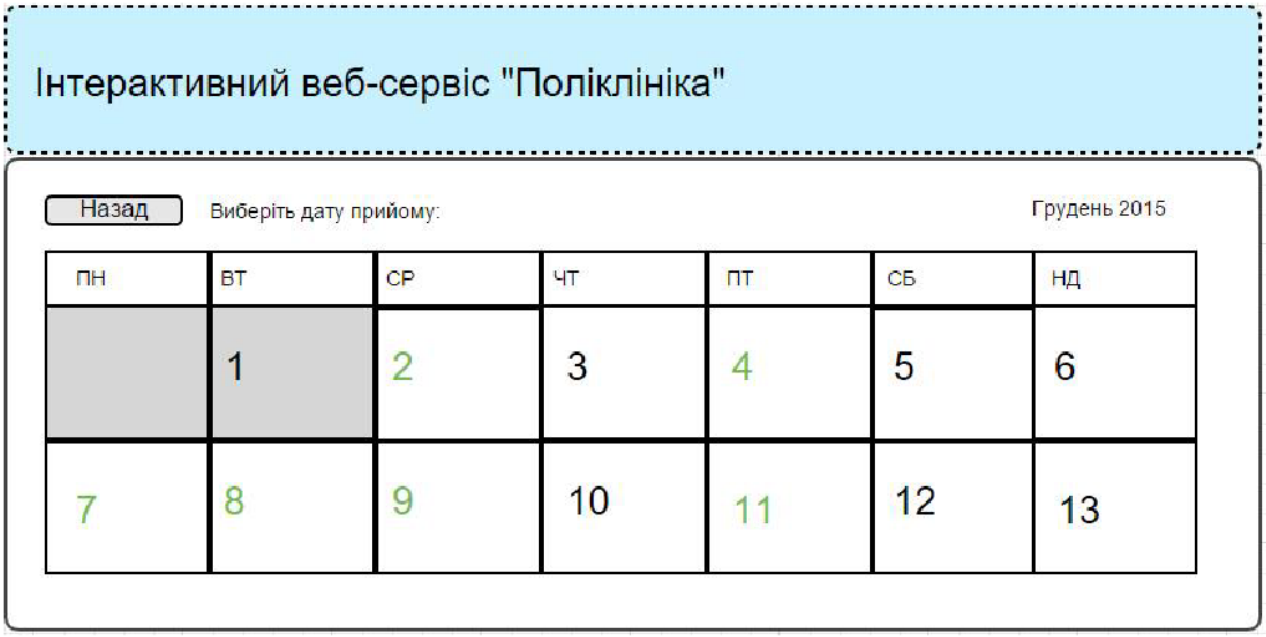


Рис. 1.17. Розкадровка сторінки вибору дати прийому


Рис. 1.18. Розкадровка сторінки вибору часу прийому

На сторінці для введення особистих даних знаходиться форма введення цих даних, яка в свою чергу складається з таких полів як: прізвище, ім’я, по-батькові, дата народження, вулиця, будинок, квартира та кнопки підтвердження. Розкадровка описаної сторінки показана на рисунку 1.19.

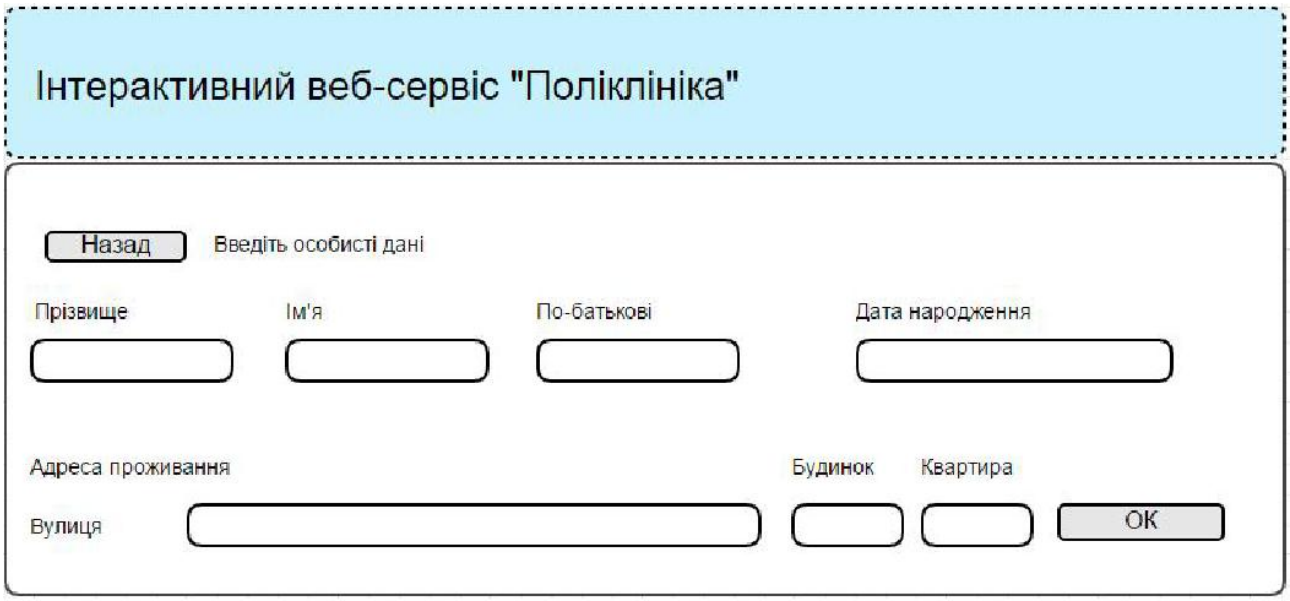


Рис. 1.19. Розкадровка сторінки введення особистих даних

Функція “Перегляд попередніх записів на прийом” дає можливість користувачу переглянути список талонів, які були замовлені на сайті з його аккаунта. На сторінці буде відображатися список талонів, посортованих по даті, від новіших до старіших. Обравши якийсь з них користувач зможе переглянути інформацію з талону.

Таблиця 1.11

Варіант використання «Перегляд попередніх записів на прийом»

| | |
|---|---|
| Контекст використання | Перегляд попередніх записів на прийом |
| Дійові особи | Користувач |
| Передумова | Авторизація в системі |
| Тригер | Відкрита особиста сторінка користувача |
| Сценарій | 1. Перехід на особисту сторінку користувача 2. Вибір посилання на список попередніх записів |
| Пост-умова | - |

Функція “Додавання медичного закладу в систему” дає змогу додати новий медичний заклад в базу даних системи. Дана функція доступна тільки

адміністратору. Адміністратор в адміністративній панелі сайту при додаванні нового медичного закладу повинен вказати наступну інформацію: назву, адресу, контактні дані, також додати всіх лікарів даного медичного закладу з графіками робіт.

Таблиця 1.12

Варіант використання «Додавання медичного закладу в систему»

| Контекст використання | Додавання медичного закладу в систему |
|---|---|
| Дійові особи | Адміністратор |
| Передумова | Авторизація в системі |
| Тригер | Відкрита сторінка додавання медичного закладу в адмінпанелі сайту |
| Сценарій | 1. Перехід в адмінпанель сайту. 2. Вибір посилання на сторінку додавання медичного закладу. 3. Введення інформації про медичний заклад. 4. Підтвердження. 5. Додавання персоналу медичного закладу. 6. Збереження. |
| Пост-умова | |

Функція “Видалення медичного закладу з системи”. Також адміністратор має можливість видаляти медичні заклади з системи, в разі необхідності.

Таблиця 1.13

Варіант використання «Видалення медичного закладу з системи»

| Контекст використання | Видалення медичного закладу з системи |
|---|---|
| Дійові особи | Адміністратор |
| Передумова | Авторизація в системі |
| Тригер | Відкрита сторінка додавання медичного закладу в |

| | |
|---|---|
| | адмінпанелі сайту |
| Сценарій | 1. Перехід в адмінпанель сайту. 2. Вибір посилання на сторінку зі списком медичних закладів. 3. Вибір медичного закладу. 4. Видалення. |
| Пост-умова | - |

Функція “Редагування інформації медичного закладу в системі”. Адміністратор при потребі має змогу редагувати інформацію про наявний в системі медичний заклад.

Таблиця 1.14

Варіант використання «Редагування даних медичного закладу»

| Контекст використання | Редагування даних медичного закладу |
|---|---|
| Дійові особи | Адміністратор |
| Передумова | Авторизація в системі |
| Тригер | Відкрита сторінка додавання медичного закладу в адмінпанелі сайту |
| Сценарій | 1. Перехід в адмінпанель сайту. 2. Вибір посилання на сторінку зі списком медичних закладів. 3. Вибір медичного закладу. 4. Перехід на сторінку редагування. 5. Внесення потрібних змін та збереження. |
| Пост-умова | - |

Специфікація функціональних вимог наведена у таблиці 1.15

Таблиця 1.15

Специфікація функціональних вимог

| Ідентифікатор вимог | Назва вимоги | Атрибути вимог | | |
|---|---|---|---|---|
| | | Пріоритет | Складність | Контакт |
| 1 | Реєстрація користувача | Обов’язкове | Висока | Замовник |
| 2 | Авторизація користувача | Обов’язкове | Середня | Замовник |
| 3 | Додавання медичного закладу | Обов’язкове | Середня | Замовник |
| 4 | Редагування медичного закладу | Обов’язкове | Висока | Замовник |
| 5 | Видалення медичного закладу | Обов’язкове | Середня | Замовник |
| 6 | Перегляд списку медичних закладів | Обов’язкове | Середня | Замовник |
| 7 | Реєстрація на прийом | Обов’язкове | Висока | Замовник |

Специфікація нефункціональних вимог наведена у таблиці 1.16

Таблиця 1.16

Специфікація нефункціональних вимог

| №. | Назва вимоги | Характеристики |
|---|---|---|
| 1. | Застосовність | |
| 1.1 | Час для навчання користувачів | Не більше 60 хв |

| | | |
|---|---|---|
| 1.2 | Вимірюваний час відгуку для типових завдань | |
| 2. | Надійність | |
| 2.1 | Середній час безвідмовної роботи | |
| 2.2 | Середнє напрацювання до ремонту | 1 рік |
| 2.3 | Максимальна норма помилок або дефектів | 100 |
| 3 | Робочі характеристики | |
| 3.1 | Місткість (максимальне значення) | 1000 користувачів |
| 4 | Проектні обмеження | |
| 4.1 | Мова програмування | PHP, JavaScript |
| 5 | Вимоги до документації, призначеної для користувача, і до системи допомоги | Наявність інструкції користувача |
| 6 | Інтерфейси | |
| 6.1 | Інтерфейс користувача | Веб-додаток |
| 6.2 | Програмні інтерфейси | jQuery RDBMS MySql PhpMyAdmin |
| 6.3 | Комунікаційні інтерфейси | Доступ до мережі інтернет |

**Висновки до першого розділу**

У даному розділі було проаналізовано предметну область для розроблюваного продукту. Досліджено структуру та напрями діяльності об’єкту управління “Реєстратура поліклініки”. Також були проаналізовані усі бізнес-процеси, які відбуваються у даному об’єкті управління. Для кращого розуміння розроблюваного продукту було порівняно три аналогових продукти: “Веб-реєстратура”, “Інтернет-портал охорони здоров’я” та “Інтернет реєстратура”. Після порівняння функціоналу систем-аналогів було визначено

переваги та недоліки, які потрібно враховувати в процесі розробки веб-сервісу “Поліклініка”.

Для кращого орієнтування у предметній області був створений глосарій, у якому подані усі терміни та визначення. Виявлені актори та варіанти використання. Ці варіанти використання були детально проаналізовано та описано. Також були описані функціональні та нефункціональні вимоги розроблюваного продукту.

# РОЗДІЛ 2. ПРОЕКТУВАННЯ

## 2.1. Розробка архітектури програмної системи

Для розробки архітектури інтерактивного веб-сервісу була взята клієнт-серверна архітектура додатку. Обрана архітектура найчастіше використовується в роботі з базами даних та мережі і забезпечує обмін даними між вказаними компонентами. Архітектура клієнт-сервер передбачає такі три основні компоненти: ·сервери, що обробляють отримані запити та видають відповідний результат; клієнти, що звертаються до серверів з запитами про дані; мережа, що забезпечує обмін даними між клієнтами та серверами.

Обробка та збереження даних відбувається на боці сервера, відображення даних і надсилання запитів на сервер виконується на боці клієнта. На рисунку 2.1 зображена трирівнева схема архітектури веб-додатку.

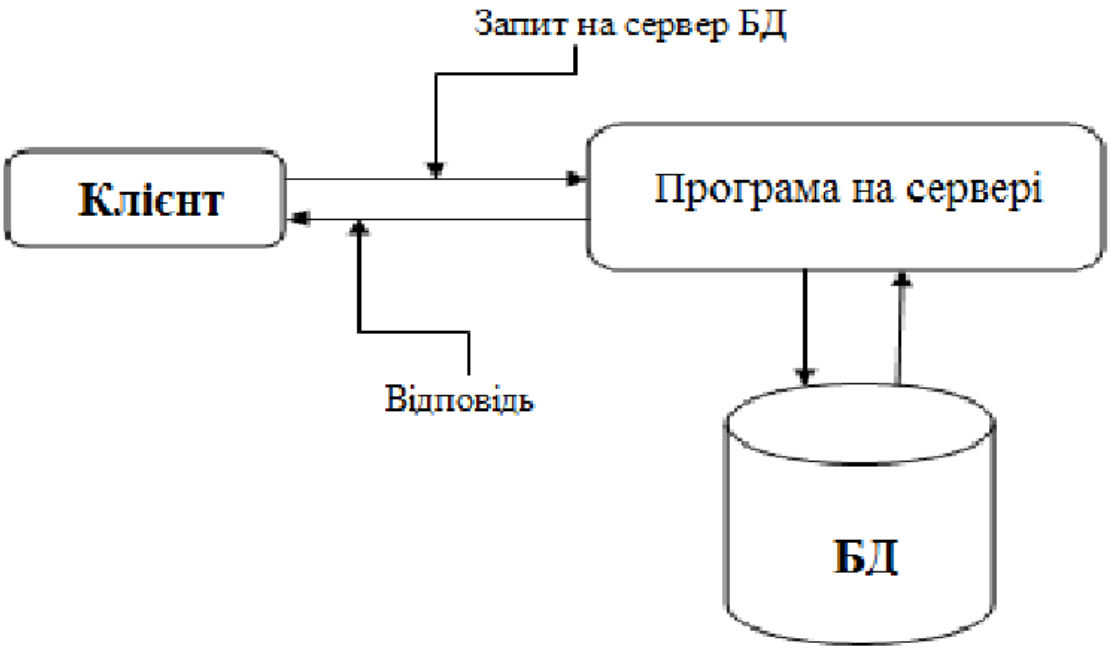


Рис. 2.1. Схема клієнт-серверної архітектури

Перший рівень – це клієнт, який відсилає запити на сервер та приймає результат обробки запитів. Клієнтом є браузер користувача. Другий рівень – це бізнес-логіка додатку. Це логіка, за якою веб-сервер обробляє отримані від клієнта запити. Третій рівень – це сама СУБД, яка отримує запити від сервера і повертає потрібні дані на сервер або зберігає їх. На рисунку 2.2 зображено діаграму ієрархії функцій системи.

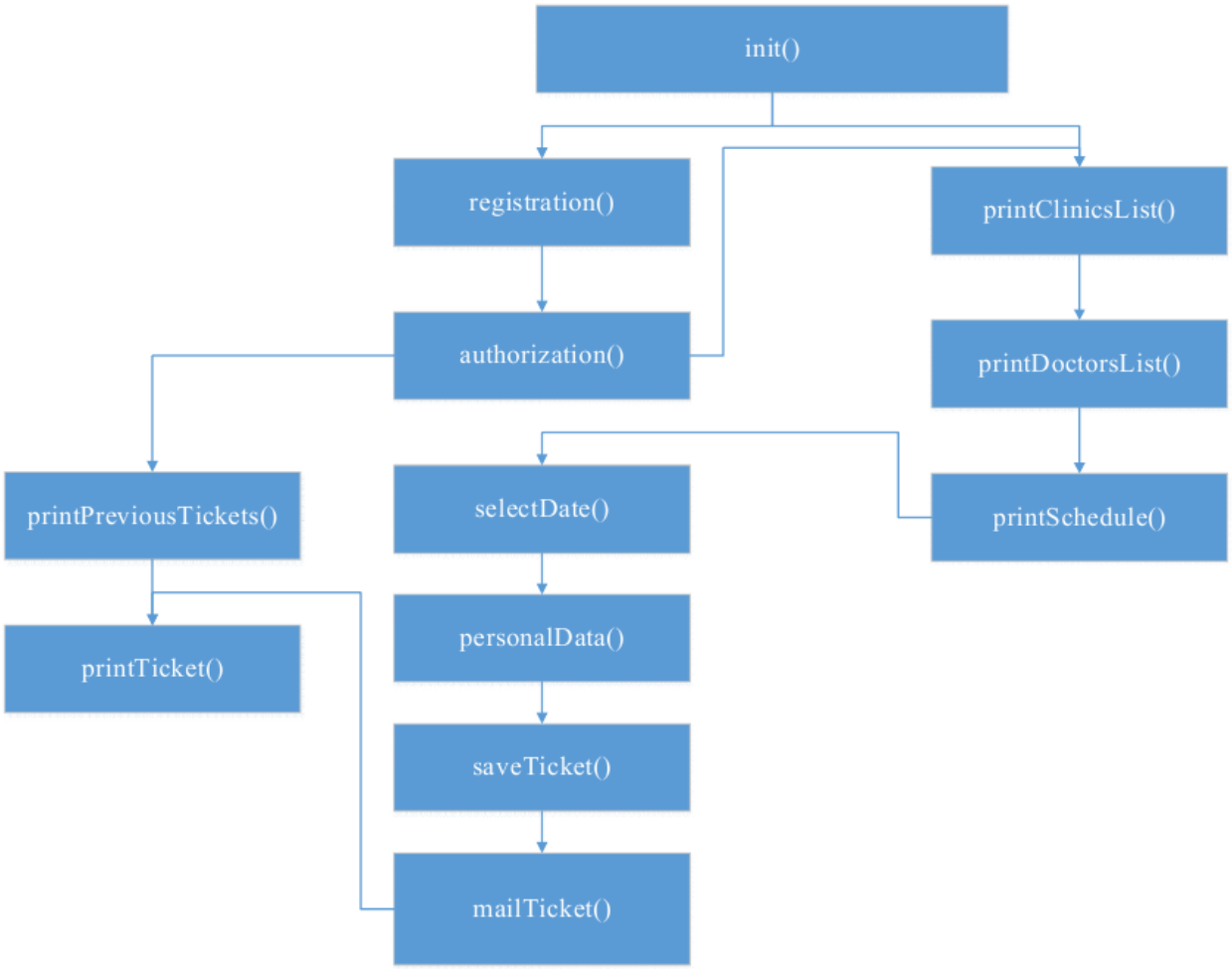


Рис. 2.2. Діаграма ієрархії функцій системи

На діаграмі ієрархії функцій показано назви функцій системи та послідовність їхнього виклику, відповідно до дій користувача в системі. Наприклад функція printPreviousTickets() не спрацює, якщо користувач перед тим не пройшов процес авторизації, тобто не виконалася функція authorization().

Діаграма станів показує те, що система може бути описана як скінченне число станів [1]. Така діаграма використовується для того, щоб надати абстрактний опис поведінки системи. Ця поведінка — це проаналізована та відтворена послідовність подій, які відбуваються в одному і більше можливих станах системи. Зазвичай, одна діаграма описує один об’єкт і відслідковує зміну станів цього об’єкта в системі. Діаграма станів процесу реєстрації в системі зображена на рисунку 2.3.

Рис. 2.3. Діаграма станів процесу реєстрації в системі

Рис. 2.4. Діаграма станів процесу реєстрації на прийом

В процесі реєстрації система отримує реєстраційні дані з форми на сторінці сайту, перевіряє їх чи дані введені в правильному форматі, а також перевіряє чи не існує користувача з вказаною електронною адресою. Після успішного проходження перевірки дані користувача зберігаються в систему і на електронну пошту користувачу відправляється лист про успішну реєстрацію на сайті. На рисунку 2.4 зображено діаграму станів, що описує процес реєстрації пацієнта на прийом.

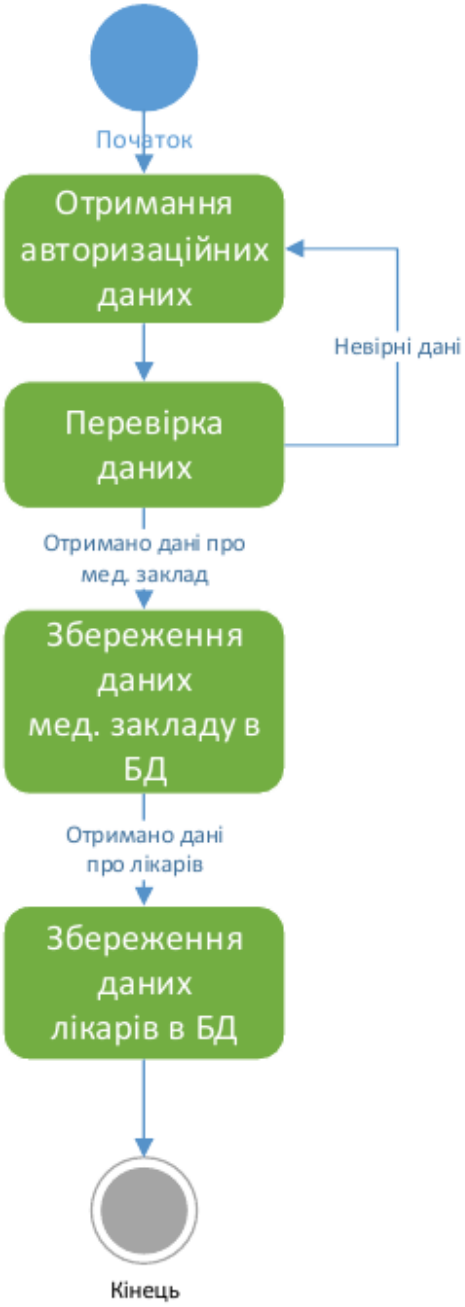


Рис. 2.5. Діаграма станів процесу “Управління даними медичного закладу”

Для того, щоб мати можливість зареєструватися на прийом до лікаря, користувач повинен бути авторизованим в системі. Система виводить список наявних медичних закладів та очікує на вибір конкретного медичного закладу. Відповідно до отриманих даних система виводить список лікарів обраного медичного закладу. Користувач обирає лікаря і система виводить графік роботи лікаря і очікує на отримання даних про дату прийому. Після отримання дати

прийому, користувач вводить особисті дані, відбувається їх перевірка, генерування та збереження талону в базі даних. Діаграма станів процесу “Управління даними медичного закладу” зображена на рисунку 2.5.

Система отримує авторизаційні дані та перевіряє чи вони відповідають авторизаційним даним користувача типу адміністратор. Після успішної авторизації система очікує на дані про медичний заклад, перевіряє їх та зберігає в базі даних. Далі отримує дані про лікаря медичного закладу, також перевіряє та зберігає їх в базі даних.

### 2.2. Проектування структури бази даних.

Для розробки моделі бази даних обрана реляційна модель даних. Вона найкраще підходить для вирішення цієї задачі, адже вона має ряд наступних переваг:

- незалежність програм від даних. Ідея використання баз даних та систем управління базами даних передбачає використання додаткового рівня між прикладними програмами та власне даними, завдяки чому прикладні програмісти можуть абстрагуватися від реалізації самої бази даних, а зосередити свою увагу на логіці обробки даних;

- простота розробки та моделювання інформаційного ресурсу як плата за деякі обмеження та уніфікацію на рівні реалізації операцій над даними;

- наявність умов керування даними за допомогою операцій над множинами.

В процесі проектування структури бази даних потрібно створити діаграму корпоративної моделі даних та на основі визначених елементів і зв’язків створити ER – діаграму. Діаграма корпоративної моделі даних представлена на рисунку 2.6 [3].

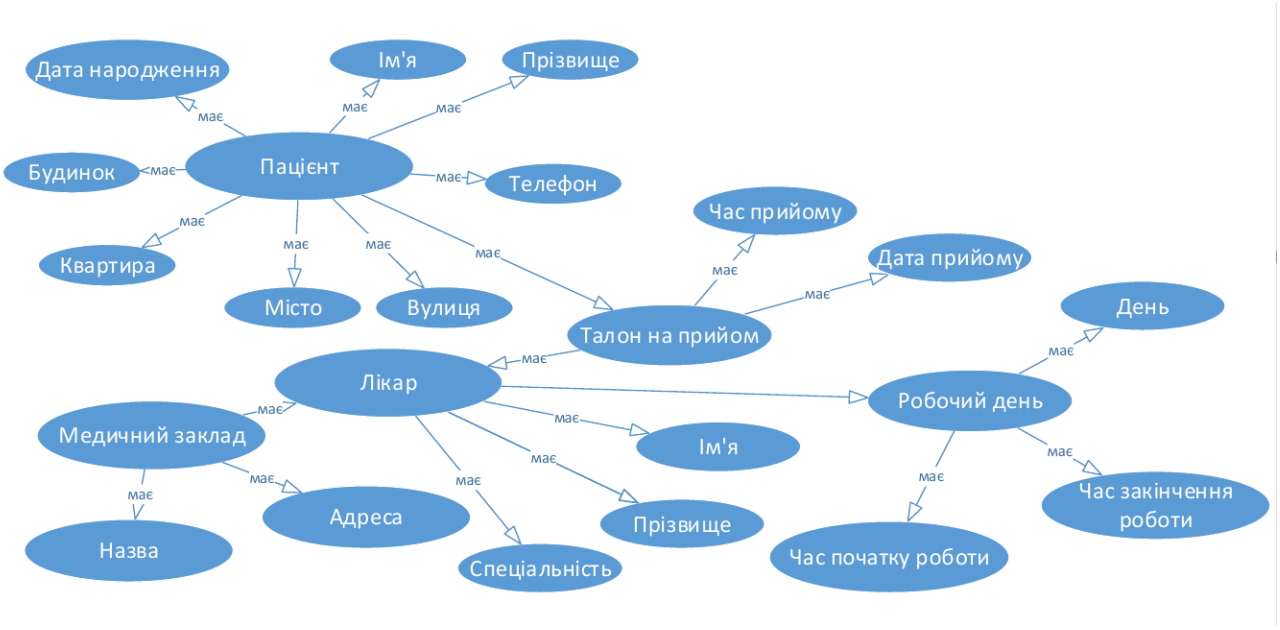


Рис. 2.6. Діаграма корпоративної моделі даних

ERD (Діаграма зв’язків сутностей) описує взаємопов’язані предмети інтересу в певній сфері знань. ER – модель складається з типів сутностей, які описують предмети інтересу, та визначає зв’язки, які можуть бути між двома екземплярами цих типів сутностей. ER – діаграма інтерактивного веб-сервісу «Поліклініка» зображена на рисунку 2.7 [3].

У фізичне проектування бази даних входить створення таблиць у відповідній БД, згідно ER – діаграми [3]. Для цього було обрано СУБД MySql.

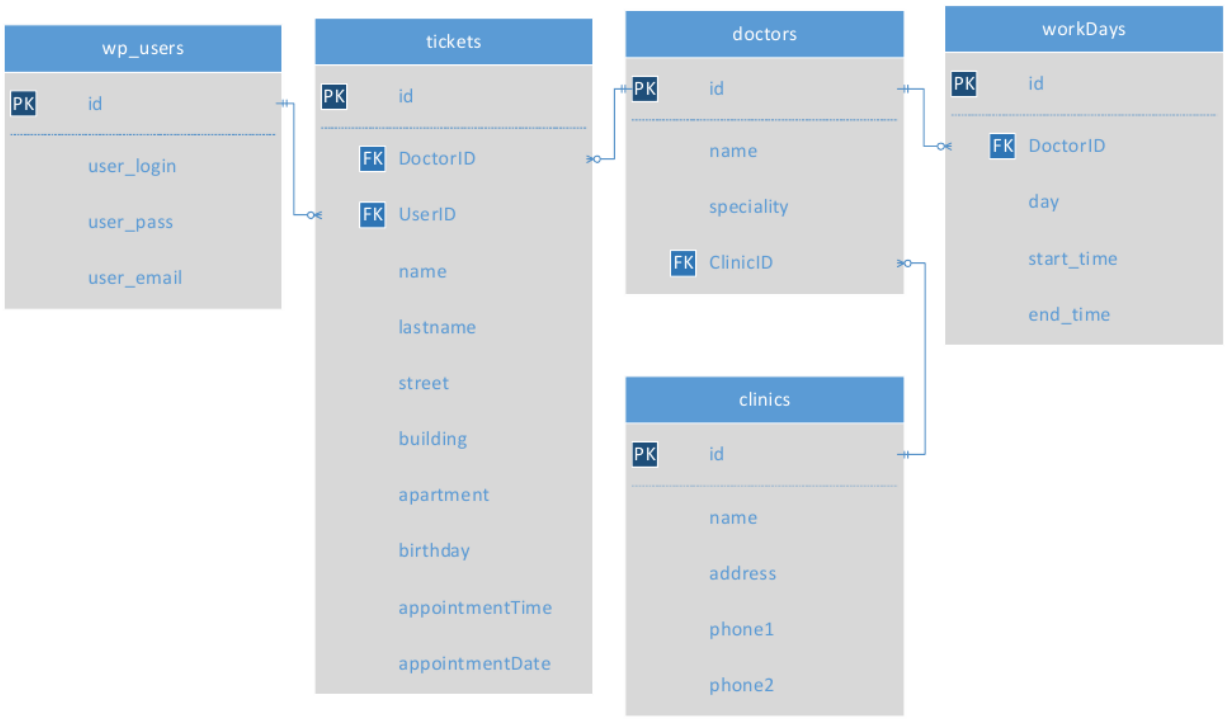


Рис. 2.7. ER – діаграмаНа ER – діаграмі зображені наступні відношення: User, Ticket, Doctor, Clinic, WorkDay.

Структура таблиці «clinics» показана на рисунку 2.8.

| | # | Назва | Тип | Порівняння | Атрибути | Нуль | За замовчуванням | Додатково |
|---|---|---|---|---|---|---|---|---|
| ☐ | 1 | id | int(11) | | | Ні | *Немає* | AUTO_INCREMENT |
| ☐ | 2 | name | varchar(50) | utf8_general_ci | | Ні | *Немає* | |
| ☐ | 3 | address | varchar(100) | utf8_general_ci | | Ні | *Немає* | |
| ☐ | 4 | phone1 | varchar(20) | utf8_general_ci | | Ні | *Немає* | |
| ☐ | 5 | phone2 | varchar(20) | utf8_general_ci | | Ні | *Немає* | |

Рис. 2.8. Структура таблиці «clinics»

Таблиця «clinics» складається з таких полів:

1. id – унікальний ідентифікатор. Тип даних integer;

2. name – поле для збереження назви медичного закладу. Тип даних varchar(50);

3. address – поле для збереження адреси медичного закладу. Тип даних varchar(100);

4. phone1, phone2 – поля для збереження телефонів медичного закладу. Тип даних varchar(20).

Структура таблиці «doctors» показана на рисунку 2.9.

| | # | Назва | Тип | Порівняння | Атрибути | Нуль | За замовчуванням | Додатково |
|---|---|---|---|---|---|---|---|---|
| ☐ | 1 | id | int(11) | | | Ні | *Немає* | AUTO_INCREMENT |
| ☐ | 2 | name | varchar(50) | utf8_general_ci | | Ні | *Немає* | |
| ☐ | 3 | speciality | varchar(50) | utf8_general_ci | | Ні | *Немає* | |
| ☐ | 4 | clinicID | int(11) | | | Ні | *Немає* | |

Рис. 2.9. Структура таблиці «doctors»

Таблиця «doctors» складається з таких полів:

1. id – унікальний ідентифікатор. Тип даних integer;

2. name – поле для збереження імені лікаря. Тип даних varchar(30);

3. speciality – поле для збереження спеціальності лікаря. Тип даних varchar(50);

4. clinicID – поле для збереження ідентифікатора медичного закладу в якому працює лікар. Тип даних integer.

Структура таблиці «tickets» показана на рисунку 2.10.

| | # | Назва | Тип | Порівняння | Атрибути | Нуль | За замовчуванням | Додатково |
|---|---|---|---|---|---|---|---|---|
| ☐ | 1 | id | int(11) | | | Ні | *Немає* | AUTO_INCREMENT |
| ☐ | 2 | doctorID | int(11) | | | Ні | *Немає* | |
| ☐ | 3 | userID | int(11) | | | Ні | *Немає* | |
| ☐ | 4 | name | varchar(30) | utf8_general_ci | | Ні | *Немає* | |
| ☐ | 5 | lastname | varchar(30) | utf8_general_ci | | Ні | *Немає* | |
| ☐ | 6 | street | varchar(30) | utf8_general_ci | | Ні | *Немає* | |
| ☐ | 7 | building | varchar(10) | utf8_general_ci | | Ні | *Немає* | |
| ☐ | 8 | appartment | varchar(10) | utf8_general_ci | | Ні | *Немає* | |
| ☐ | 9 | birthday | date | | | Ні | *Немає* | |
| ☐ | 10 | appointmentDate | date | | | Ні | *Немає* | |
| ☐ | 11 | appointmentTime | time | | | Ні | *Немає* | |

Рис. 2.10. Структура таблиці «tickets»

Таблиця «tickets» складається з таких полів:

1. id – унікальний ідентифікатор. Тип даних integer;2. doctorID – поле для збереження ідентифікатора лікаря, на прийом до якого зареєструвався користувач. Тип даних integer.

3. userID – поле для збереження ідентифікатора лікаря, на прийом до якого зареєструвався користувач. Тип даних integer.

4. name – поле для збереження імені користувача, зареєстрованого на прийом. Тип даних varchar(30);

5. lastname – поле для збереження прізвища користувача, зареєстрованого на прийом. Тип даних varchar(30);

6. street – поле для збереження вулиці проживання користувача. Тип даних varchar(30);

7. building – поле для збереження номера будинку проживання користувача. Тип даних varchar(10);

8. apartment – поле для збереження номера квартири проживання користувача. Тип даних varchar(10).

9. birthday – поле для дати народження пацієнта. Тип даних date;

10. appointmentDate – поле для дати прийому. Тип даних date;

11. appointmentTime – поле для часу прийому. Тип даних time.

Структура таблиці «wp_users» показана на рисунку 2.11.

| # | Назва | Тип | Порівняння | Атрибути | Нуль | За замовчуванням | Додатково |
|---|---|---|---|---|---|---|---|
| 1 | ID | bigint(20) | | UNSIGNED | Ні | *Немає* | AUTO_INCREMENT |
| 2 | user_login | varchar(60) | utf8mb4_unicode_ci | | Ні | | |
| 3 | user_pass | varchar(255) | utf8mb4_unicode_ci | | Ні | | |
| 4 | user_email | varchar(100) | utf8mb4_unicode_ci | | Ні | | |

Рис. 2.11. Структура таблиці «wp_users»

Таблиця «wp_users» складається з таких полів:

1. ID – унікальний ідентифікатор. Тип даних biginteger;

2. user_login– поле для збереження логіна користувача. Тип даних Varchar(255)

3. user_pass– поле для збереження пароля користувача. Тип даних varchar(255);

4. user_email – поле для збереження електронної пошти користувача. Тип даних varchar(100).

Структура таблиці «workdays» показана на рисунку 2.12.

| # | Назва | Тип | Порівняння | Атрибути | Нуль | За замовчуванням | Додатково |
|---|---|---|---|---|---|---|---|
| 1 | id | int(11) | | | Ні | *Немає* | AUTO_INCREMENT |
| 2 | doctorID | int(11) | | | Ні | *Немає* | |
| 3 | day | tinyint(1) | | | Ні | *Немає* | |
| 4 | start_time | time | | | Ні | *Немає* | |
| 5 | end_time | time | | | Ні | *Немає* | |

Таблиця «workdays» складається з таких полів:

1. id – унікальний ідентифікатор. Тип даних integer;

2. doctorID – поле для збереження ідентифікатора лікаря до якого відноситься даний робочий графік. Тип даних integer.

3. day – поле для збереження порядкового номеру дня тижня. Тип даних tinyint.

4. start_time – поле для збереження часу початку роботи лікаря. Тип даних time;

5. end_time – поле для збереження часу закінчення роботи лікаря. Тип даних Integer;

**Висновки до другого розділу**

У цьому розділі була розроблена архітектура веб-додатку. Спершу було обрано одну із загальних архітектур розробки програмного забезпечення, яка найкраще підходила для вирішення поставленої задачі. Для кращого розуміння архітектури продукту було розроблено діаграми, які показують взаємодію користувачів та їх функцій між собою та у системі.

Оскільки дана система оперує даними, було спроектовано архітектуру бази даних. Проаналізовано потоки даних у системі. Усі ці потоки були представлені на діаграмі потоків даних. Оскільки база даних є реляційною, була створена діаграма корпоративної моделі даних, яка показує об’єкти системи та зв’язки між ними, а також ER-діаграма. Вона показує взаємодію між відношеннями і дозволяє приступити до програмування бази даних

# РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

## 3.1. Програмна реалізація проекту

Для реалізації проекту було обрано мову PHP. PHP - мова програмування, яка виконує програмний код та генерує HTML-сторінки на стороні веб-сервера. PHP є найбільш використовуваною мовою програмування для створення веб-орієнтованого програмного забезпечення [16].

В PHP є готові бібліотеки для роботи з багатьма базами даних, такими як: PostgreSQL, Oracle, mSQL, MySQL Informix. Використовуючи прикладний програмний інтерфейс для доступу до систем управління базами є можливість з'єднатися з будь-якою базою даних, якщо до неї існує драйвер.

Для керування данми використовується система управління реляційними базами даних MySQL. MySQL є системою з відкритим кодом та є повністю безкоштовною. Це одна з найбільш використовуваних систем управління базами даних в програмних продуктах базованих на архітектурі клієнт-сервер. Має хорошу підтримку з боку мови програмування PHP.

Для адміністрування системи управління базами даних MySQL використовується веб-застосунок з графічним веб-інтерфейсом, розроблений на мові програмування PHP – phpMyAdmin. В якості середовища для написання коду було обрано програмний продукт Brackets. Brackets — безкоштовний редактор з відкритим програмним кодом, використовується для редагування коду JavaScript, HTML, CSS, PHP.

Програмна реалізація веб-сервісу почалася з верстки шаблону сайту. Верстка виконувалася мовою розмітки гіпертексту HTML5. Для надання розмітці стилів використовувалися каскадні таблиці стилів CSS3. Було обрано варіант блочної верстки з використанням фреймворку Bootstrap 3, а саме системи сітки даного фреймворку. Використання сітки Bootstrap полегшує створення адаптивного дизайн сайту, тобто такого дизайну, який буде коректно відображати вміст веб-ресурсу, на різних пристроях, масштабуючись

відповідно до розмірів пристрою або до ширини екрану пристрою. Також при розробці шаблону була використана JavaScript бібліотека – jQuery.

Для керування даними інтерактивного веб-сервісу “Поліклініка” було створено нові користувацькі типи записів, а саме: “Клініка”, “Лікар” та “Талон”. Код створення типу записів Клініка наведено нижче

```
function register_post_type_clinic() {
    $singular = __('Clinic');
    $plural = __('Clinics');
    $plural_slug = str_replace( ' ', '_', $plural );
    $labels = array(
            'name'                              => $plural,
            'singular_name'          => $singular,
            'add_new'                       => 'Add New',
            'add_new_item'                  => 'Add New ' . $singular,
            'edit_item'              => 'Edit ' . $singular,
            'new_item'               => 'New ' . $singular,
            'parent'                        => 'Parent ' . $singular
      );
      $args = array(
            'labels'                  => $labels,
            'public'                => true,
        'publicly_queryable'  => true,
        'show_in_nav_menus'   => true,
        'show_ui'             => true,
        'show_in_admin_bar'   => true,
        'menu_position'       => 6,
        'menu_icon'           => 'dashicons-building',
        'query_var'           => true,
        'capability_type'     => 'page',
                'rewrite'               => array(
            'slug' => strtolower( $plural_slug ),
            'with_front' => true,
            'pages' => true,),
        'supports'            => array( 'title'   ));

      register_post_type( 'clinic', $args);
} add_action( 'init', 'register_post_type_clinic' );
```

Новий тип записів створюється з використанням функції register_post_type ($id, $args ), першим параметром якої є назва створюваного типу записів, а другим є масив параметрів [15]. Після створення типу записів

“Клініка”, в адміністративній панелі сайту, в навігаційному меню з’являється новий пункт з назвою створеного типу.

В системі повинні зберігатися назва медичного закладу, адреса, телефони та електронна пошта. Для цього було створено метабокс з полями для введення даних медичного закладу, перерахованих вище. Для запобігання несанкціонованого доступу до даних, а саме збереження та редагування даних про медичний заклад було використано метод wp_nonce_field(), який створює приховане перевірочне поле зі згенерованим кодом для перевірки джерела переданих даних, а точніше для того, щоб переконатися, що дані відправлені саме з поточного сайту, а не з іншого місця. Також було використано метод get_post_meta(), який повертає масив значень всіх полів запису. Код реалізації збереження та можливості редагування даних медичних закладів надано нижче.

```
function clinic_meta_save ($post_id) {
    $is_autosave = wp_is_post_autosave($post_id);
    $is_revision = wp_is_post_revision($post_id);
    $is_valid_nonce          =          (isset($_POST['clinic_nonce'])          &&
wp_verify_nonce($_POST['clinic_nonce'], basename(__FILE__))) ? 'true' : 'false';
    if ( $is_autosave || $is_revision || !$is_valid_nonce ) {
        return;
    }
    if ( isset( $_POST[ 'clinic_id' ] ) ) {
      update_post_meta(   $post_id,   'clinic_id',   sanitize_text_field(   $_POST[
'clinic_id' ] ) );
    }
    if ( isset( $_POST[ 'clinic_name' ] ) ) {
      update_post_meta(   $post_id,   'clinic_name',   sanitize_text_field(   $_POST[
'clinic_name' ] ) );
    }
    if ( isset( $_POST[ 'clinic_address' ] ) ) {
      update_post_meta(  $post_id,  'clinic_address',  sanitize_text_field(  $_POST[
'clinic_address' ] ) );
    }
    if ( isset( $_POST[ 'clinic_email' ] ) ) {
      update_post_meta(   $post_id,   'clinic_email',   sanitize_text_field(   $_POST[
'clinic_email' ] ) );
    }
    if ( isset( $_POST[ 'clinic_phone1' ] ) ) {
      update_post_meta(  $post_id,  'clinic_phone1',  sanitize_text_field(  $_POST[
'clinic_phone1' ] ) );
    }
```

Після виконання наведеного коду на сторінці додавання лікаря з правого боку з’являється метабокс, з можливістю обрати медичний заклад в якому

працює лікар. Для реалізації процесу реєстрації пацієнта на прийом до лікаря, першим кроком було створено шаблон сторінки з виводом всіх медичних закладів наявних в системі.

```
$clinics = new WP_Query( array( 'post_type' => 'clinic', 'posts_per_page' => -1 )
);
<?php if ($clinics->have_posts()) : ?>
<?php _e('<h1 style="text-align: center;">Оберіть медичний заклад</h1> ',
'polyclinic'); ?>
<ul id="clinics-list">
<?php
while ( $clinics->have_posts() ) : $clinics->the_post();
$address = get_post_meta( get_the_ID(), 'clinic_address', true);
$phone1 = esc_html(get_post_meta( get_the_ID(), 'clinic_phone1', true));
$phone2 = esc_html(get_post_meta( get_the_ID(), 'clinic_phone2', true));
$slug = get_permalink();
?>
<a href="<?php echo (esc_url($slug)); ?> ">
<li id="<?php esc_attr( the_ID() ); ?>" class="clinic">
<h5><?php esc_html(the_title()); ?></h5>
<i class="icon-location"></i> <span><?php echo (esc_html($address)); ?></span><br>
<i class="icon-phone"></i><span>Тел: <?php echo "$phone1   $phone2" ?></span>
</li>
</a>
<?php endwhile; ?>
</ul>
<?php else: ?>
<p><?php _e( 'Немає медичних закладів.', 'polyclinic' ); ?></p>
<?php endif; ?>
```

Після вибору медичного закладу було створено шаблон сторінки для виводу всіх лікарів обраного медичного закладу (single-clinic.php). Вивід лікарів реалізовано аналогічними методами, що і вивід медичних закладів. Для сортування списку лікарів по назві спеціальності в алфавітному порядку було використано php-функцію array_multisort [16, 19]. Код сортування масиву з даними про лікарів наведено нижче.

```
$specialities = array();
foreach ($doctors as $key => $row) {
        $specialities[$key] = $row['speciality'];
}
array_multisort($specialities, SORT_ASC, $doctors);
$previousSpeciality;
foreach ($doctors as $doctor) {
if($previousSpeciality != $doctor["speciality"]) {
echo "<h2 style='margin: 10px 0;'>".$doctor["speciality"]."</h2>";
$previousSpeciality = $doctor["speciality"];
}
$displayItem="<a
href=".$doctor["link"]."?clinic=".$clinic_name."&speciality=".$doctor["speciality"]
."&room=".$doctor["room"].">";
$displayItem .="<li class='doctor'>";
$displayItem .= "<h5>" .$doctor["name"]. "</h5>";
$displayItem     .=     "<span>Спеціальність:     "     .$doctor["speciality"].
"</span><br/>";
$displayItem .= "<span>Кабінет №" .$doctor["room"]. "</span></li></a>";
echo $displayItem; }
```

Після вибору лікаря, користувач переходить на детальну сторінку лікаря (single-doctor.php). З використанням http методу передачі даних Get, з попередньої сторінки відправляються ідентифікатор медичного закладу, до якого належить лікар та його спеціальність. На цій сторінці для неавторизованих користувачів відображається тільки графік роботи лікаря. Для авторизованих користувачів також форма для вибору дати і часу прийому. Графік роботи лікаря підтягується з сервера після завантаження сторінки, завдяки технології ajax, яка дозволяє відправляти асинхронні запити на сервер. Код ajax-запиту наведено нижче [14, 19].

```
jQuery.ajax({
    url: PICK_A_DATE.ajax_url,
    method: 'POST',
    dataType: 'json',
    data: {
        action: 'calculate_dates',
        doctorId: doctorID,
        security: PICK_A_DATE.security
    },
    success: function(data){
        if(data) {
            console.log(PICK_A_DATE.success);
            schedule_table(data.doctor_schedule);;
            schedule = data.doctor_schedule;
            disabledDates = data.ticketsDates;
            console.log(disabledDates);
           if(PICK_A_DATE.authorized{ jQuery("div.container.pickers").show();}
        }
    },
    error: function(error) {
        console.log(error);
        console.log(PICK_A_DATE.fail); }});
```

Php функція для обробки ajax запиту, на основі переданого методом Post ідентифікатора лікаря, повертає графік роботи лікаря та дати прийому, вказані в талонах, які відносяться до даного лікаря. Дані повертаються у форматі json. Код функції, яка обробляє ajax запит наведено нижче.

```
function clinic_calculate_dates() {
    if ( !check_ajax_referer( 'nonce-for-date', 'security' ) ) {
        return wp_send_json_error( 'Invalid Nonce' );
    }
    $doctorID = $_POST['doctorId'];
    $doctor_scedule = get_post_meta($doctorID);
    $schedule = array(
```

```
'mon_start_time'       =>       date('H:i',       strtotime(get_post_meta($doctorID,
'doctor_mon_start', true))),
'mon_end_time'              =>    date('H:i',    strtotime(get_post_meta($doctorID,
'doctor_mon_end', true))),
'tue_start_time'       =>       date('H:i',       strtotime(get_post_meta($doctorID,
'doctor_tue_start', true))),
'tue_end_time'              =>    date('H:i',    strtotime(get_post_meta($doctorID,
'doctor_tue_end', true))),
'wed_start_time'       =>       date('H:i',       strtotime(get_post_meta($doctorID,
'doctor_wed_start', true))),
'wed_end_time'              =>    date('H:i',    strtotime(get_post_meta($doctorID,
'doctor_wed_end', true))),
'thu_start_time'       =>       date('H:i',       strtotime(get_post_meta($doctorID,
'doctor_thu_start', true))),
'thu_end_time'              =>    date('H:i',    strtotime(get_post_meta($doctorID,
'doctor_thu_end', true))),
'fri_start_time'       =>       date('H:i',       strtotime(get_post_meta($doctorID,
'doctor_fri_start', true))),
'fri_end_time'              =>    date('H:i',    strtotime(get_post_meta($doctorID,
'doctor_fri_end', true))),);
```

```
$ticket_args = array(
            'post_type'  => 'ticket',
            'meta_query' -> array(
                array(
                    'key'   -> 'doctor_id',
                    'value' => $doctorID
                ),
                array(
                    'key'       => 'visiting_date',
                    'value'     -> date("Y-m-d"),
                    'compare'   => '>=',
                    'type'      => 'date'
                )
            )
        );
        $queryTickets = new WP_Query($ticket_args);
        $ticketsDates = array();
        foreach($queryTickets->posts as $post) {
            $date = esc_html(get_post_meta( $post->ID, 'visiting_date', true));
```

Для реалізації можливості вибору дати та часу прийому було використано плагін pickadate.js [14, 17]. Також було створено функцію, яка обчислює вільні та зайняті години прийому лікаря після вибору дати. Після підтвердження вибраної дати, користувач перенаправляється на сторінку з формою для введення персональних даних. Після введення коректних даних на основі всіх попередньо введених даних користувача генерується електронний талон на прийом та зберігається в базі даних. Код генерування та збереження талону наведено нижче. Також були реалізовані функції відправлення талону на електронну пошту користувача та функція друку талону. Ще було реалізовано можливість перегляду попередніх талонів. Створено шаблон для виводу всіх попередніх талонів користувача (archive-ticket.php) та шаблон для виводу детальної інформації конкретного талону (single-ticket.php).

### 3.2. Програмна реалізація бази даних

Налаштування з'єднання з базою даних виконуються в файлі wp-config.php. Використовуючи PHP-функцію define, ініціалізуються значення констант, значеннями яких є ім'я бази даних, ім'я користувача бази даних, пароль до бази даних, ім'я сервера та кодування даних. Код ініціалізації параметрів підключення до бази даних наведено нижче.

```
/** The name of the database for WordPress */
define('DB_NAME', 'clinic');
/** MySQL database username */
define('DB_USER', 'root');
/** MySQL database password */
define('DB_PASSWORD', '');
/** MySQL hostname */
define('DB_HOST', 'localhost');
define('DB_CHARSET', 'utf8mb4');
```

Використовуючи значення змінних з файлу wp-config.php засобами мови програмування PHP підключається до сервера бази даних та обирає вказану базу даних. Для роботи з даними які є в таблицях бази даних реалізований PHP-клас wpdb. Даний клас дозволяє виконувати довільні операції з базою даних: додавати, оновлювати, отримувати та видаляти дані. Для виконання операцій з базою даних потрібно лише знати методи цього класу, тобто звичайні PHP-функції, тільки всередині класу. Звернення до методів класу wpdb відбувається через глобальну змінну $wpdb. Також з допомогою цього класу відбувається керування користувацькими таблицями в базі даних, а не тільки тими які були створені самою системою управління контентом. Приклад виконання вибірки записів з таблиці в якій зберігається інформація про медичні заклади з використанням класу wpdb наведено нижче.

```
$clinics = $wpdb->get_results( "SELECT id, address, name, phone1, phone2 FROM
clinics" );
```

Код додавання нової клініки в базу даних з використанням методу query, класу wpdb.

```
$wpdb->query( “INSERT INTO clinics (name, address, phone1, phone2) VALUES
(‘Поліклініка №1’, ‘вул. Острозького 7’, ‘0352436603’, ‘0352271689’) ” );
```

Код для збереження всіх записів з таблиці “doctors”, зовнішній ключ яких відповідає запису з таблиці “clinics” з ідентифікатором 4, в змінній $wp_query у вигляді масиву. Записи відсортовані в алфавітному порядку за значенням полів “speciality” та “name”.

```
$wp_query = $wpdb->query( “SELECT * FROM `doctors` WHERE `clinicID`= 4 ORDER BY
`speciality`, `name` ASC” );
```

На рисунку 3.1 зображено результат виконання запиту описаного вище, в PhpMyAdmin.

| id | name ▲ 2 | speciality ▲ 1 | clinicID |
|---|---|---|---|
| 4 | Назвальський Богдан Володимирович | Кардіолог | 4 |
| 5 | Проців Людмила Іванівна | Невролог | 4 |
| 3 | Ґреґорі Хаус | Терапевт | 4 |
| 6 | Амосов Микола Михайлович | Хірург | 4 |
| 7 | Шідловський Віктор Олександрович | Хірург | 4 |

Рис. 3.1. Вибірка лікарів в базі даних

Код для отримання дати та часу прийому вказаних у всіх дійсних талонах до лікаря з ідентифікатором 1. В блоці умови даного запиту використовується агрегатна функція NOW(), яка повертає поточну дату, для порівняння дати прийому з поточною датою. Це потрібно для того, щоб в результат вибірки не потрапляли талони, в яких вказана дата прийому вже минула.

```
$wpdb->query( “SELECT `appointmentDate`, `appointmentTime` FROM `tickets` WHERE
`doctorID` = 1 AND `appointmentDate` > NOW() GROUP BY `appointmentDate`,
`appointmentTime` ” );
```

## Висновки до третього розділу

В цьому розділі було описано та обґрунтовано технології обрані для програмної реалізації інтерактивного веб-сервісу “Поліклініка”. Також було реалізовано функціонал системи відповідно до вимог та наведено фрагменти програмного коду з описом їх роботи. Обґрунтовано вибір засобів для роботи з базою даних. Наведено приклади SQL-запитів та результати їх виконання.

# РОЗДІЛ 4. ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

## 4.1 Тестування

Функціональне тестування проводиться, для того, щоб перевірити чи відповідає розрозблеий програмний продукт функціональним вимогам або виявити невідповідності поведінки програми до очікуваної поведінки [5]. Для того, щоб провести функціональне тестування необхідно розробити тестові випадки у вигляді exel-таблиці, що містить такі дані:

- id - ідентифікатор тестового випадку;

- summary – опис об’єкту тестування;

- expected result – очікуваний результат;

- passed/failed – показує чи тест пройшов чи провалився;

- pre-condition – умова, яка необхідна для виконання, щоб виконати тестовий випадок;

- steps to reproduce – шлях проходження тестового випадку.

Розроблено 12 тестових випадків або test-cases з яких всі пройшли успішно. Текст тестових випадків наведено в додатку Д. Оскільки виконання всіх тестових випадків було успіше, то можна стверджувати, що програмний продукт відповідає поставленим функціональним вимогам. Для того, щоб GUI тестування дало більший ефект, застосовують автоматизоване тестування інтерфейсу [10]. Для проведення тестування користувацького інтерфейсу було обрано програмний продукт Selenium. Так як, розроблена програмна система є веб-сервісом, то для тестування було встановлено плагін для браузеру Firefox, Selenium IDE. Цей плагін дозволяє створювати тести, які являють собою записи дій користувача над сайтом. Результат проведення тестування користувацького інтерфейсу зображено на рисунку 4.1.

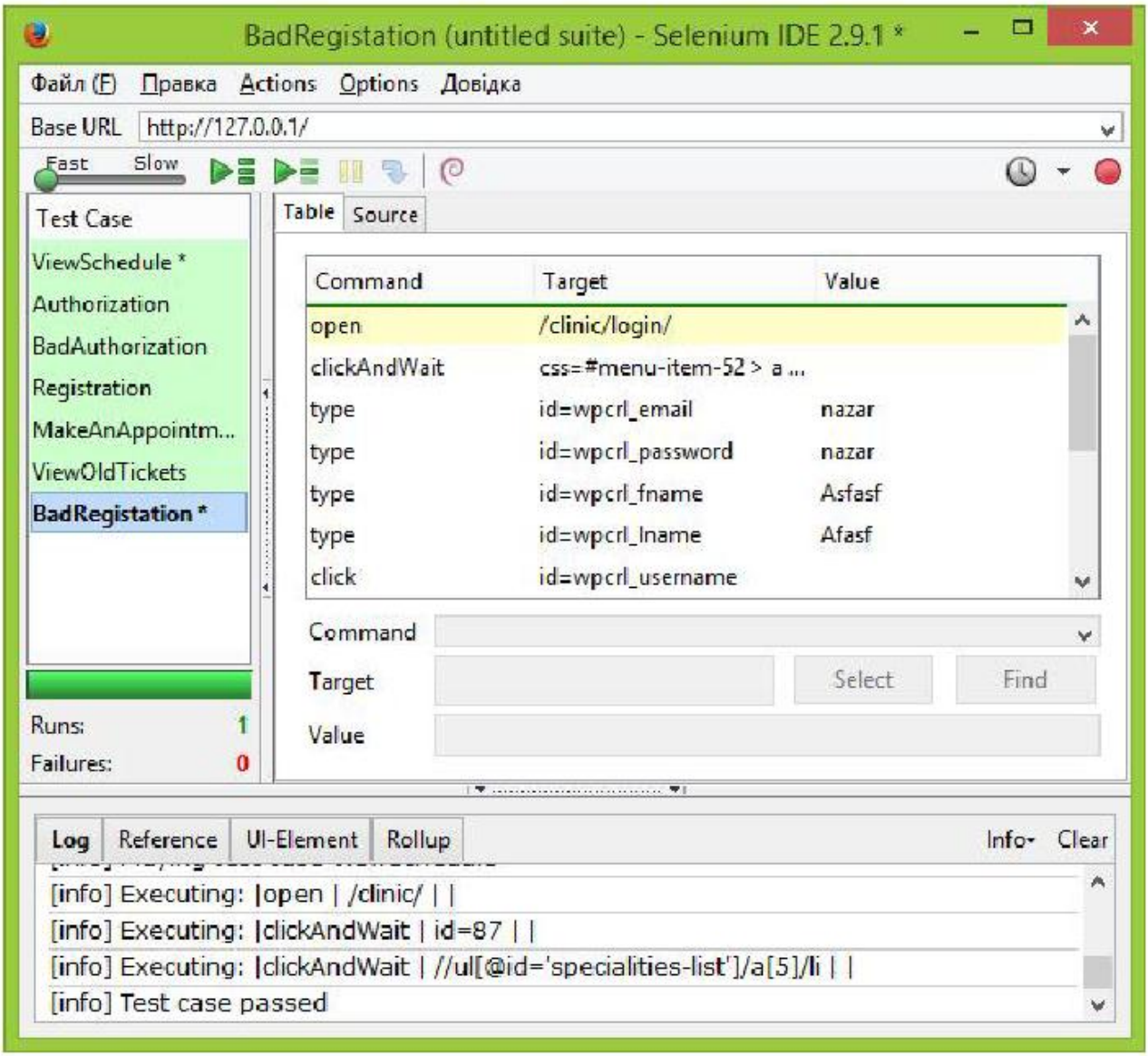


Рис. 4.1. Результат проведення GUI-тестування

Всі тестові випадки були пройдені успішно. Код тестів зображених на рисунку 4.1, наведений у додатку Е. Тест-план поданий у додатку Ж. Також важливим видом тестування є тестування безпеки. Цей вид відноситься до нефункціонального тестування. В процесі цього тестування перевіряється та аналізується поведінка програмного забезпечення в різних стресових ситуаціях. Особлива увага звертається на повідомлення про помилки, які виводить система. Для тестування безпеки були обрані наступні критерії: тестування прав доступу; перевірка значень, що вводяться в форми; коректність посилань.

1. Тестування прав доступу. Користувачу надається доступ до інформації відповідно до рівня його аккаунту. Звичайний користувач не може потрапити на сторінки адміністративної панелі сайту.

2. Перевірка значень. В будь-яке текстове поле можна вносити тільки ту інформацію, яка відповідає шаблону даних цього поля (наприклад у поле для введення номерного значення, можна вводити тільки числа). При введені некоректних даних система не приймає їх для подальшого використання, а натомість виводить повідомлення про введення помилкових даних. При введені коректних даних в поля вводу, перед тим як зберегти їх в базі даних система обробляє значення текстових полів функціями які видалять з тексту теги розмітки та програмний код.

3. Коректність посилань. Кожне посилання переправляє користувача на ту сторінку, якій воно відповідає. Після проведення тестування безпеки розроблюваного веб-сайту, можна зробити висновок, що він відповідає поставленим вимогам безпеки.

### 4.2 Розгортання програмного продукту

Для забезпечення повноцінного функціонування інтерактивного веб-сервісу “Поліклініка” потрібно розмістити його компоненти на веб-сервері, а саме файли сайту та базу даних. Для цього в панелі управління хостингом необхідно створити новий сайт та присвоїти йому доменне ім’я, після чого створити на сервері директорію з назвою, яка відповідає доменному імені, в ній створити папку www та завантажити всі файли сайту в цю папку. Створювати директорії на веб-хостингу та завантажувати файли можна використовуючи файловий менеджер, який підтримує ftp протокол або використовуючи файловий менеджер вбудований в систему управління веб хостингом. Також потрібно створити на сервері баз даних нову базу, а в файлі wp-config вказати коректні дані для доступу до сервера баз даних та до самої бази. Після цього виконати імпорт структури бази даних з sql файлу.

Для користування функціоналом сайту користувачу знадобиться доступ до Інтернету з комп’ютера, який використовується, та встановлений будь-який веб-браузер, що підтримує стандарти HTML5 та CSS3.

### 4.3 Інструкція користувача

Для того, щоб користувачу були доступні всі функції даного програмного продукту, перша за все він повинен пройти процес реєстрації в системі. Для цього потрібно перейти на сторінку “Зареєструватися”, посилання на яку знаходиться в головному меню сайту та заповнити реєстраційну форму. На рисунку 4.2 зображено сторінку реєстрації.

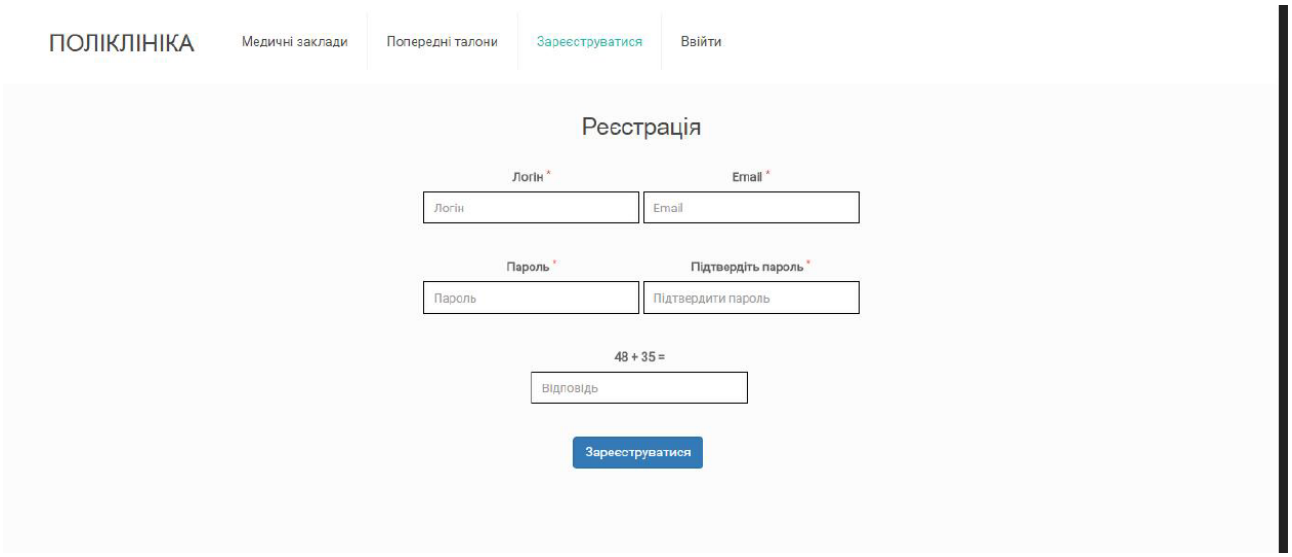


Рис. 4.2. Сторінка реєстрації

Після реєстрації користувач може здійснювати реєстрацію на прийом до лікаря. Для цього на сторінці “Медичні заклади” (рисунок 4.3) потрібно обрати один медичний заклад зі списку.


Рис. 4.3. Сторінка “Медичні заклади”

Обравши медичний заклад користувачу потрібно буде обрати лікаря. Сторінка зі списком лікарів показана на рисунку 4.4.

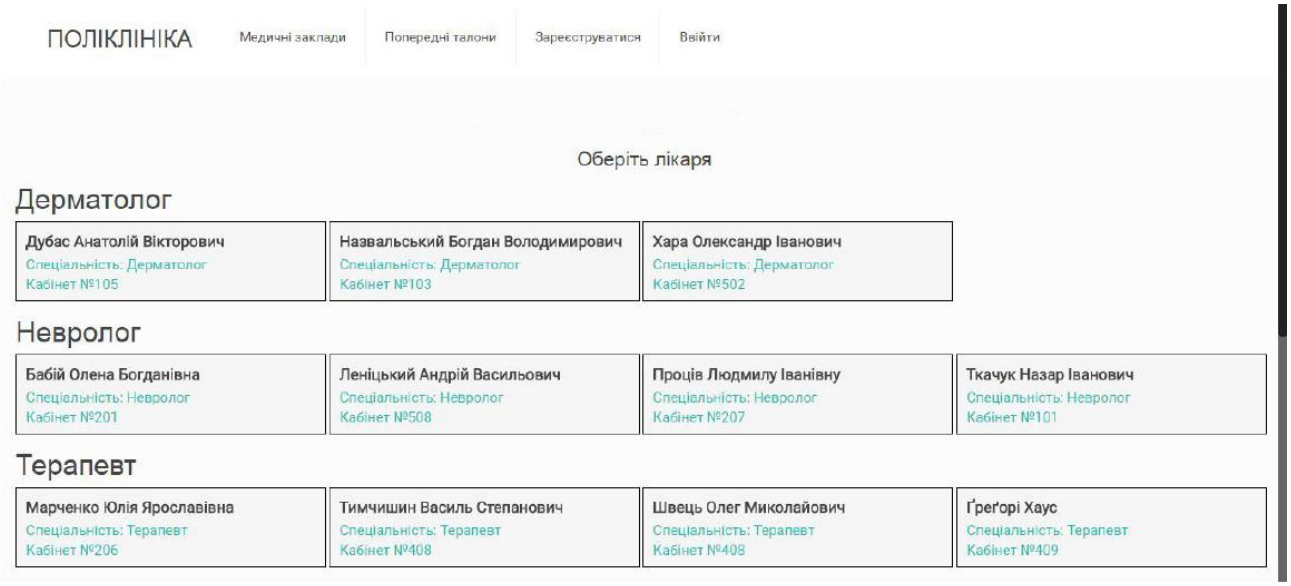


Рис. 4.4. Сторінка зі списком лікарів

Далі, після вибору лікаря, користувачу буде завантажено сторінку з графіком роботи обраного лікаря та форму для вибору дати та часу прийому (рисунок 4.5).

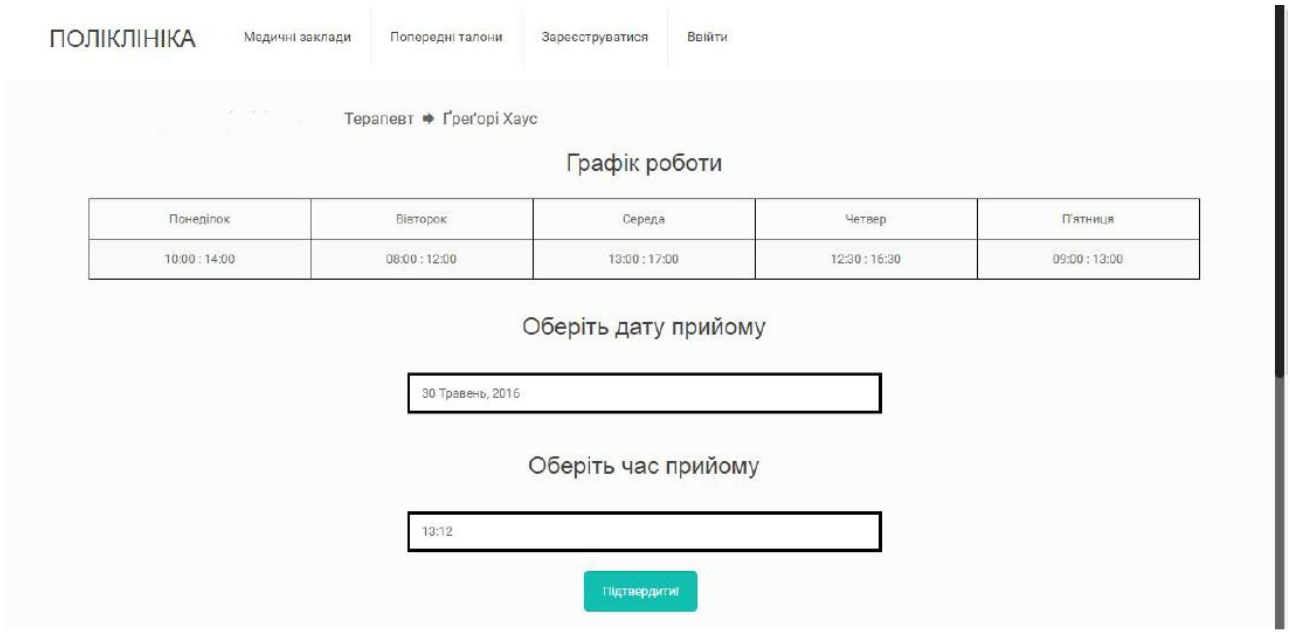


Рис. 4.5. Вибір дати і часу прийому

Підтвердивши обрану дату та час прийому, користувач повинен буде заповнити форму для особистих даних. Сторінка з формою введення особистих даних зображена на рисунку 4.6.

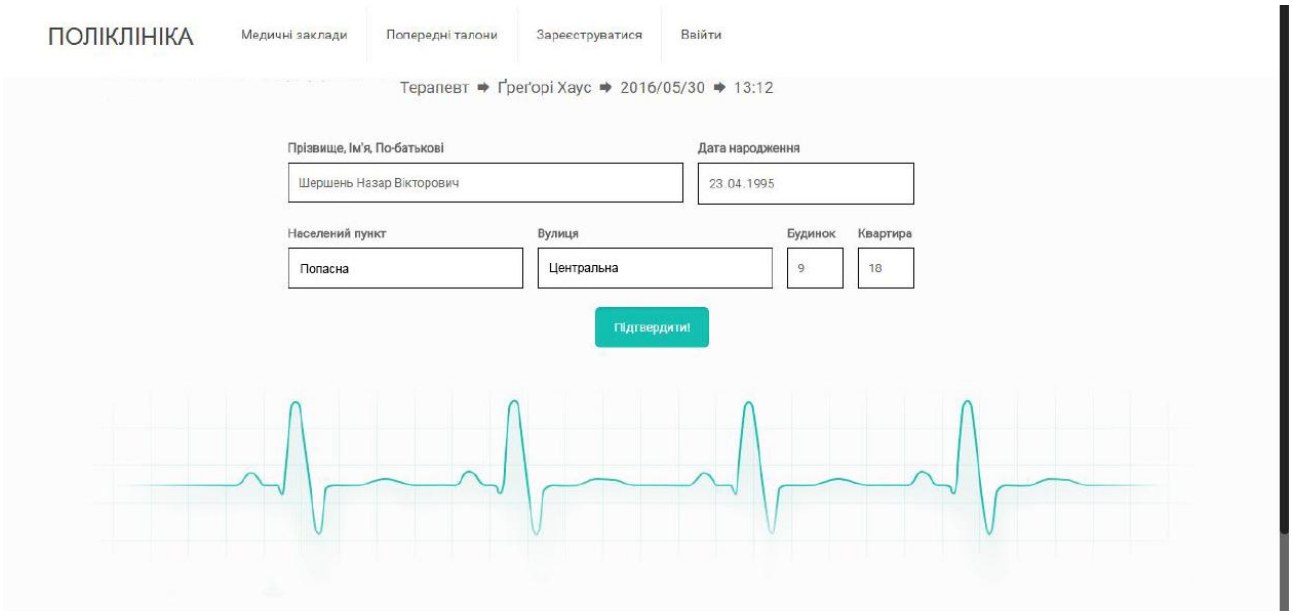


Рис. 4.6. Вибір дати і часу прийому

Після введення особистих даних, користувач натиснувши кнопку “Підтвердити” буде перенаправлений на сторінку, де буде виведено згенерований талон на прийом (рисунок 4.7).

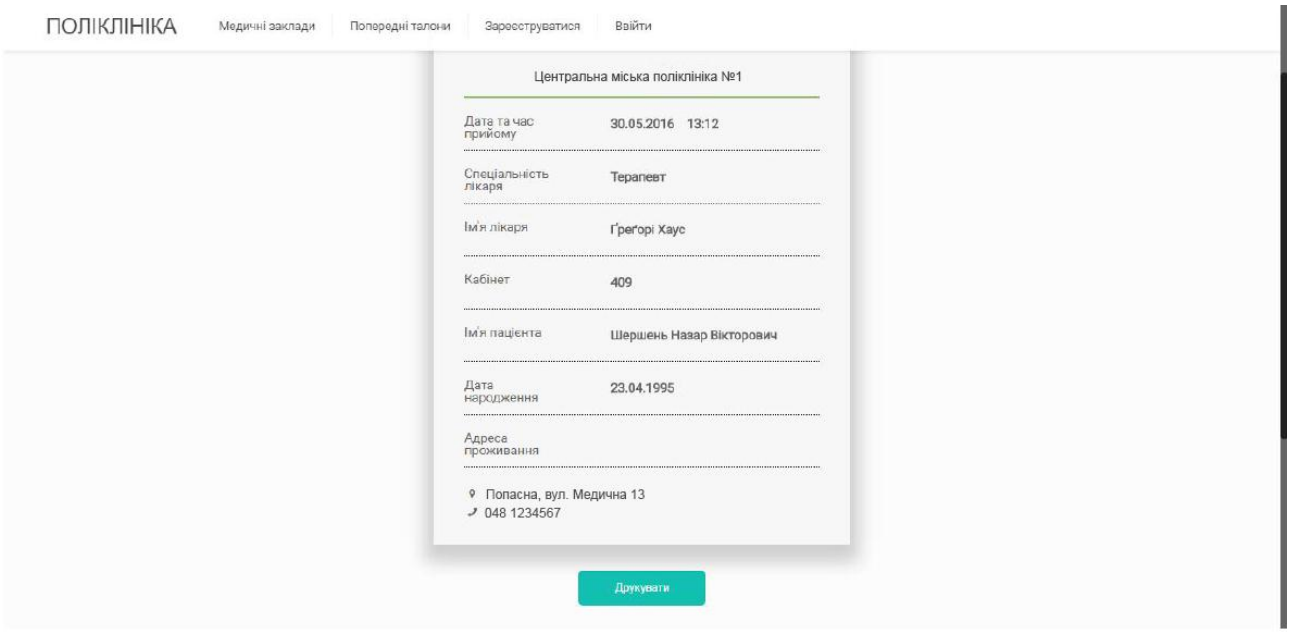


Рис. 4.7. Вибір дати і часу прийому

Також талон автоматично надсилається користувачу на електронну скриньку, вказану ним при реєстрації на сайті. Користувач має можливість роздрукувати згенерований талон натиснувши кнопку “Друкувати”. При потребі користувач може переглянути історію талонів, замовлених з його аккаунту на сайті, просто перейшовши на сторінку “Попередні талони” (рисунок 4.8).

Обравши певний талон користувач зможе переглянути його деталі. Під час використання розробленої системи можуть виникати помилки у випадку, якщо у веб-браузері користувача буде вимкнуто виконання коду написаного на мові програмування JavaSript. У такому випадку на сторінку не зможуть завантажитися дані про графік роботи лікаря, що спричинить неможливість виконання основного функціоналу системи, а саме реєстрації на прийом. Щоб

уникнути даних «неприємностей», потрібно увімкнути виконання JavaSript в налаштуваннях веб-браузера.

### 4.4 Адмін-панель для внесення результатів аналізів та діагнозів

Меню сторінки адміністрування містить такі групи посилань: «Редагування шаблонів», «Зберігання результатів аналізів», «Пошук», «Статистичні діаграми».

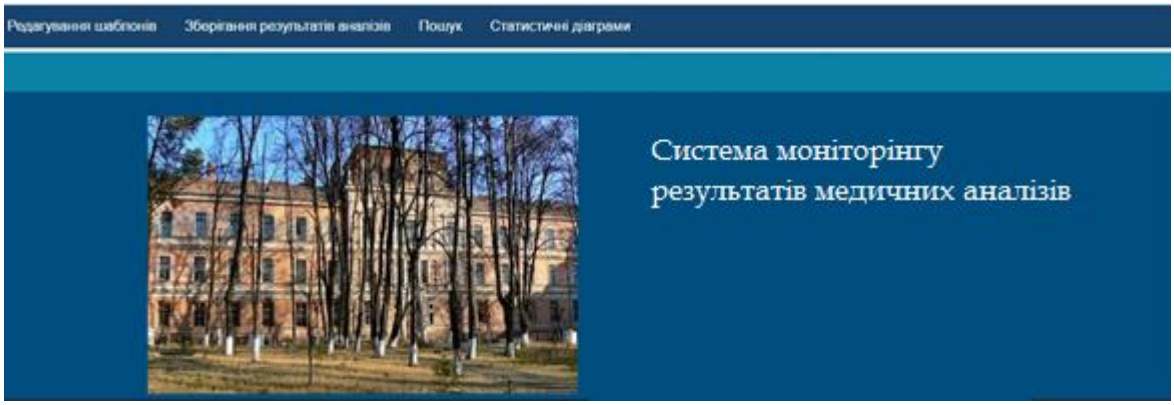


Рис. 4.8. Головна сторінка адмен-панелі

Група «Редагування шаблонів» містить посилання на сторінки, які призначені для створення та редагування шаблонів зберігання аналізів. Посилання: редагування шаблону додавання аналізу; редагування діагнозів.

Група «Зберігання результатів аналізів» містить посилання на сторінки з шаблонами, які створюються при редагуванні шаблонів (група «Редагування шаблонів»). При створенні нового шаблону, у списку посилань з'явиться посилання на новий шаблон додавання аналізу, названий як вид аналізу. Посилання:

- Аналіз мікрофлори.

Група «Пошук» містить такі посилання:

- Пошук за критеріями;
- Пошук аналізів пацієнта.

Група «Статистичні діаграми» містить такі посилання:

- Діаграма розподілу пацієнтів по діагнозу.
- Порівняльна діаграма розподілу захворюваності по районах.

Розглянемо сторінку «Редагування шаблону додавання аналізу». На ній розміщені форми, які дозволяють редагувати таблиці «Analyzes», «Parameters». Поля цих форм дають можливість додавати та видаляти типи аналізів та їх параметри. На сторінці розміщена підказка у формі таблиці, в якій вказано варіанти задання меж параметрів та вигляд референтного інтервалу, який отримаємо у результаті (Рис. 4.9.).

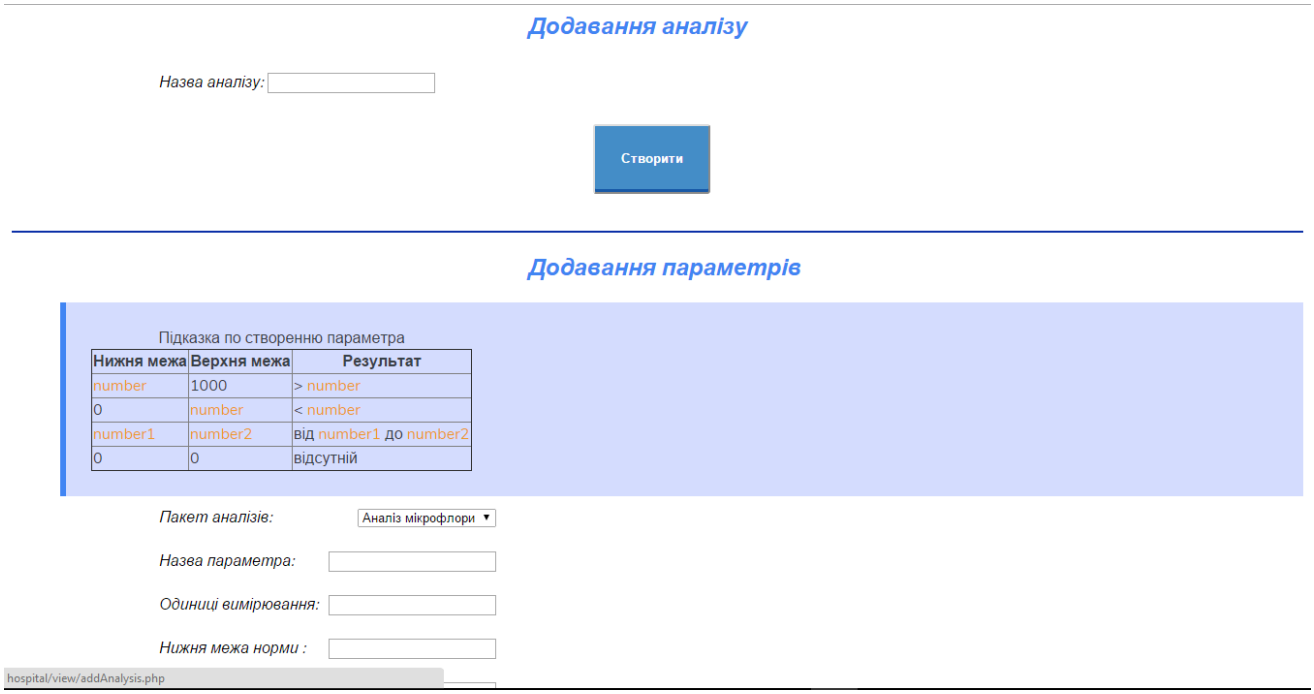


Рис. 4.9. Сторінка «Редагування шаблону додавання аналізу».

На сторінці «Редагування діагнозів» можна змінювати таблицю «Diagnoses», тобто редагувати список діагнозів (додавати та видаляти діагнози) (Рис. 4.10.).

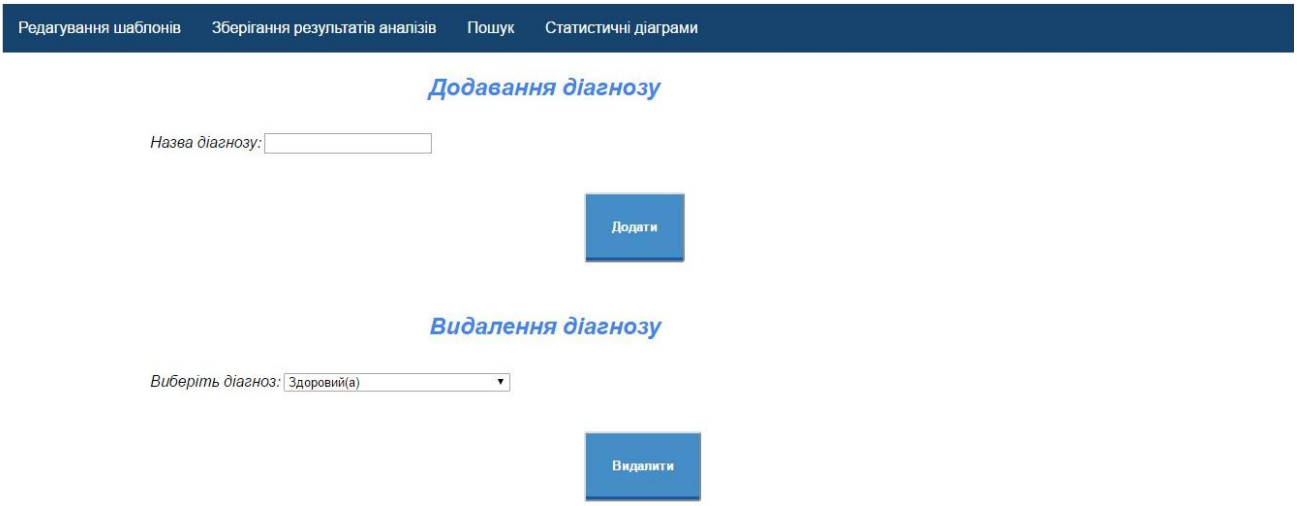


Рис. 4.10. Сторінка «Редагування діагнозів».

Сторінка «Аналіз мікрофлори» містить форму для додавання результату аналізу мікрофлори. На ній вводиться інформація про пацієнта, результати показників аналізу, заключення лікаря, проведене лікування тощо (Рис. 4.11.).

Посилання на цю сторінку розміщене у розділі «Зберігання результатів аналізів». Аналогічні сторінки, з відповідним видом аналізів та параметрами, створюються при додавання нового аналізу на сторінці «Редагування шаблону додавання аналізів».

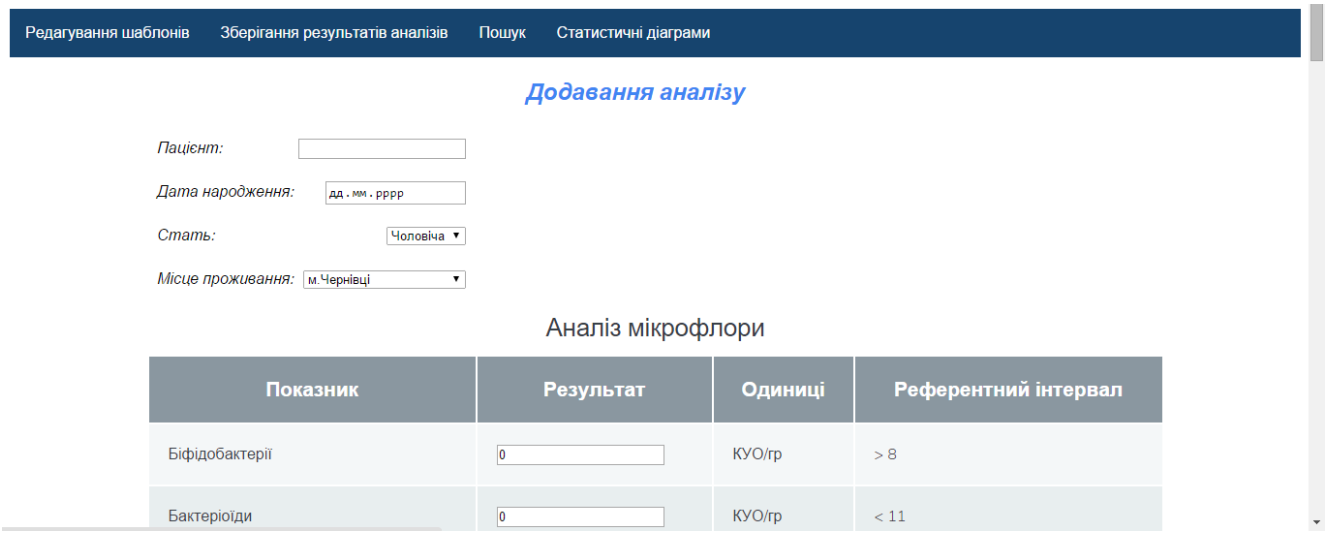


Рис. 4.11. Сторінка «Аналіз мікрофлори».

Сторінка «Пошук за критеріями» (Рис. 4.12.) дозволяє побачити результати пошуку аналізів за такими фільтрами:

- Вік;

- Стать;
- Місце проживання;
- Параметр не в нормі – назва параметра, результат якого у зоні підвищеної уваги;
- Заключення – діагноз, зроблений після проведеного аналізу;
- Проведене лікування – ключове слово у полі «Проведене лікування»;
- Вид аналізу – назва проведеного аналізу.

Таблиця результатів містить порядковий номер аналізу у пошуку, номер аналізу у базі даних, ім'я пацієнта, дата завершення аналізу, назва аналізу, посилання для перегляду аналізу.

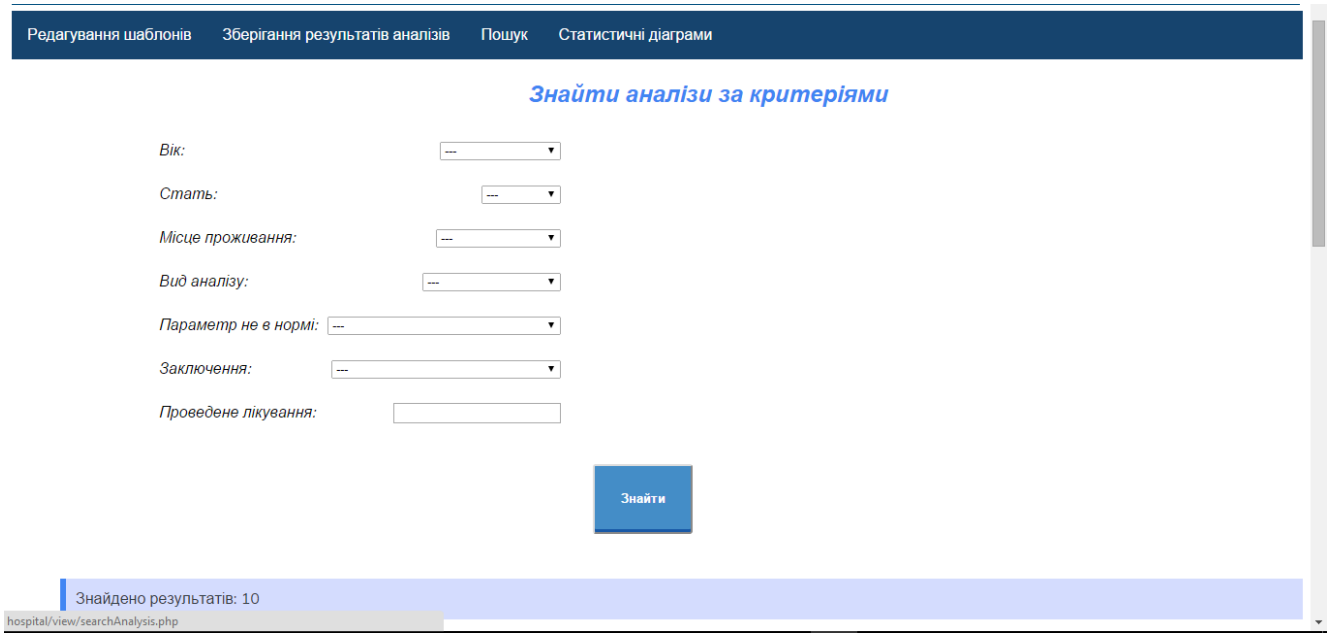


Рис. 4.12. Сторінка «Пошук за критеріями»

Сторінка «Пошук аналізів пацієнта» дозволяє знайти всі аналізи пацієнта за іменем людини.

На сторінці «Діаграма розподілу пацієнтів по діагнозу» (Рис. 4.13) знаходиться кругова діаграма, яка зображає відсотки кількості хворих на певну хворобу від загальної кількості пацієнтів.

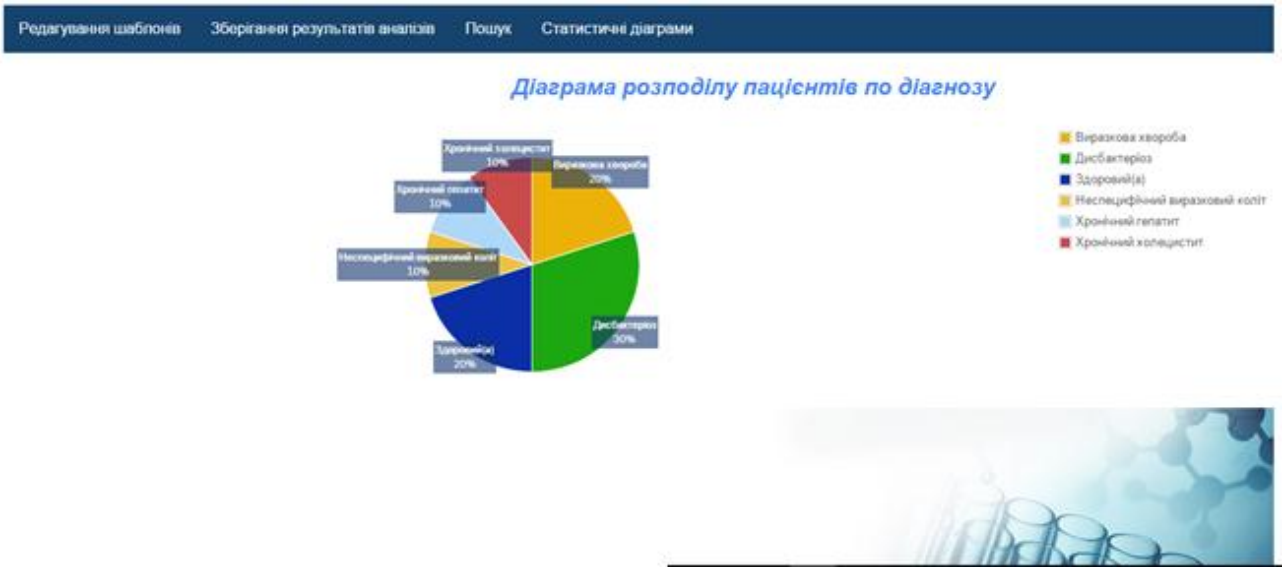


Рис. 4.13. Сторінка «Діаграма розподілу пацієнтів по діагнозу»

### 4.5. Функціонал сторінки додавання аналізу

Посилання на сторінки, призначені для додавання результатів певних аналізів, розміщені у розділі «Зберігання результатів аналізів».

Наприклад, сторінка «Аналіз мікрофлори» містить форму для додавання результату аналізу мікрофлори (Рис. 4.11.).


Рис. 4.14. Перегляд результатів аналізів та діагнозу

Для відображення сторінки з шаблоном додавання аналізу використовується PHP файл «addAnalysisResult.php», який відкривається із вказаним зовнішнім параметром - ідентифікаційний номер аналізу, який потрібно відобразити. Наприклад: «addAnalysisResult.php/analysis_id=1/».

**Висновки до четвертого розділу**

Було проведено функціональне тестування, автоматизоване тестування користувацького інтерфейсу з використанням програмного додатку для браузера Selenium IDE та тестування безпеки. Створено тест-план. Описано процес розгортання програмного продукту та створено інструкцію користувача.

# ВИСНОВКИ

Під час виконання дипломної роботи було спроектовано та розроблено інтерактивний веб-сервіс. Створений програмний продукт надає користувачу можливість зареєструватися на прийом до лікаря з обраного медичного закладу. Також переглядати графік роботи лікарів та історію попередніх записів. В повному обсязі було реалізовано функціонал системи, описаний в специфікації вимог.

В процесі проектування та розробки були взяті до уваги переваги та недоліки розглянутих систем з функціоналом, аналогічним до розробленої системи. Після завершення етапу програмної реалізації було проведено тестування системи. Згідно з результатами проведеного тестування програмний продукт повністю відповідає функціональним вимогам та виконує всі необхідні задачі.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Дейт К . Введение в системы баз даных /К.Дейт// – К. ; М. ; СПб.: Изд.дом “Вильямс”. – 2000. –560 с.
2. Дивак М.П. Системний аналіз та проектування КІС /М.П.Дивак// Навчальний посібник – Т.: Економічна думка. – 2004.
3. Калбертсон Роберт, Браун Крис, Кобб Гэри Быстрое тестирование. — М.: «Вильямс», 2002. — 374 с.
4. Квентин Зервас. Web 2.0: создание приложений на PHP = Practical Web 2.0 Applications with PHP. —М.:«Вильямс», 2009. — С. 544.
5. Костарев А. Ф. PHP 5. — Спб.: «БХВ-Петербург», 2008. — С. 1104.
6. Кузнецов Максим, Симдянов Игорь. PHP на примерах. — 2-е изд. перераб. и доп. — Спб.: «БХВ-Петербург», 2011. — С. 400.
7. Кузнецов Максим, Симдянов Игорь. PHP. Практика создания Web-сайтов. — 2-е изд. перераб. и доп. — Спб.: «БХВ-Петербург», 2008. — С. 1264.
8. Лайза Криспин, Джанет Грегори Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд = Agile Testing: A Practical
9. Guide for Testers and Agile Teams. — М.: «Вильямс», 2010. — 464 с
10. Мэтт Зандстра. PHP: объекты, шаблоны и методики программирования, 3-е издание = PHP Objects, Patterns and Practice, Third Edition. — М.: «Вильямс», 2010. — С. 560
11. Синицын С. В., Налютин Н. Ю. Верификация програмного обеспечения. — М.: БИНОМ, 2008. — 368 с.
12. jQuery API documentation [Електронний ресурс]. - Режим доступу: http://api.jquery.com/
13. PHP documentation [Електронний ресурс]. - Режим доступу: http://php.net/docs.php/
14. Pickadate.js API documentation [Електронний ресурс]. - Режим доступу: http://amsul.ca/pickadate.js/api/