МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ШЕРШЕНЬ Назар Вікторович

Інтерактивний веб-сервіс "Поліклініка"/ Interactive web-service "Polyclinic"

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконав студент групи ПЗС-41
Н. В. Шершень

_____

Науковий керівник:
викладач ПОРПЛИЦЯ Н.П.

_____

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

РЕЗЮМЕ

Дипломна робота містить 104 сторінок, 16 таблиць, 40 рисунків, список використаних джерел із 20 найменувань та 7 додатків.

Метою дипломної роботи є розробка веб-сайту для реєстрації різнотипних медичних закладів. 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 можливістю подальшого запису на прийом до обраного спеціаліста.

Об'єктом дослідження є процес реєстрації пацієнтів на прийом в реєстратурі медичного закладу.

Предметом дослідження є застосування сучасних інформаційних технологій для розробки програмного продукту, який забезпечуватиме можливість реєстрації пацієнтів на прийом до лікаря в обраному медичному закладі зі списку заздалегідь зареєстрованих в системі.

Методи розробки базуються на використанні сучасних веб-технологій, системи управління контентом та системи управління реляційними базами даних.

Одержані результати полягають в розробці інтерактивного веб-сервісу "Поліклініка".

Ключові слова: електронний талон, адміністративна панель, веб-сервіс, система управління контентом, система управління реляцінийми базами даних, тестові випадки.

SUMMARY

Thesis contains 104 pages, 16 tables, 40 drawings, list of used sources with 20 titles and 7 annexes.

The aim of the thesis is to develop a website, which will contain different types of medical institutions. This website will provide to a wide range of users opportunity to view information about any medical institutions registered in system (general information about medical institution, list of doctors and doctor’s schedule) with a further appointment to selected specialist.

The object of research is the process of registering patients at the reception desk in the clinic.

The subject of research is the use of modern information technologies to create software for registration patients to the doctor at the selected medical institution from the list of medical institutions previously registered in the system.

Methods of developing based on usage of a web technologies, content management system and relational database management system.

The resulting is creating an interactive web service "Polyclinic".

Keywords: electronic ticket, admin panel, web service, content management system, relational database management system, test cases.

ЗМІСТ

ВСТУП

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.

Для досягнення мети дипломної роботи, необхідно вирішити такі завдання:

1. дослідити та проаналізувати веб-сайти, які надають користувачам можливість запису на прийом до лікаря. Порівняти їхній функціонал, а також виділити переваги та недоліки;
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 «Поліклініка» для запису на прийом в медичні заклади, згідно проведених робіт, перерахованих в пунктах 1-4.

Об’єкт дослідження – процес запису пацієнта на прийом до лікаря в медичному закладі.

Предмет досліджень – застосування технологій створення веб-орієнтованого програмного забезпечення для розробки інтерактивного веб-сервісу «Поліклініка».

В процесі розробки використовуватимуться наступні мови програмування та технології:

PHP – інтерпретована мова програмування для виконання скріптів та генерування HTML сторінок на стороні сервера.

JavaScript – об’єктна мова програмування, що виконується на стороні клієнта в браузері.

CMS Wordpress – система управління контентом веб-сайту.

MySql – система управління реляційними базами даних.

SQL – мова запитів для взаємодії з сервером баз даних.

Ajax – технологія створення асинхронних запитів до веб-сервера.

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 “Поліклініка” 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

Продовження таблиці 1.3

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.

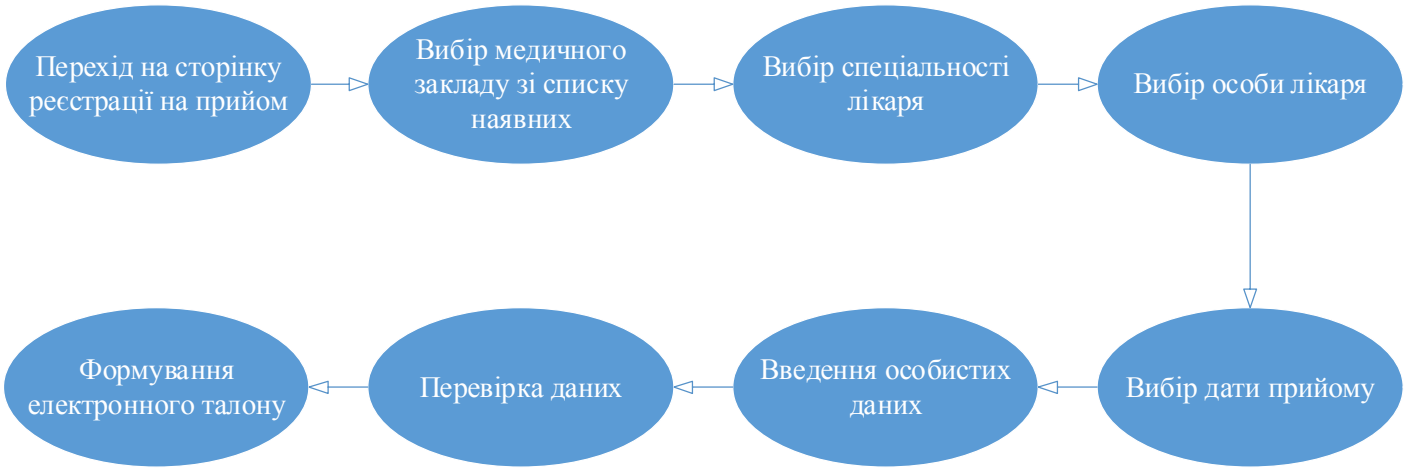

Рис. 1.4. Діаграма функцій процесу реєстрації на прийом

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 Архангельської області”, 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. На рисунку 1.5 показано вигляд головної сторінки сайту.

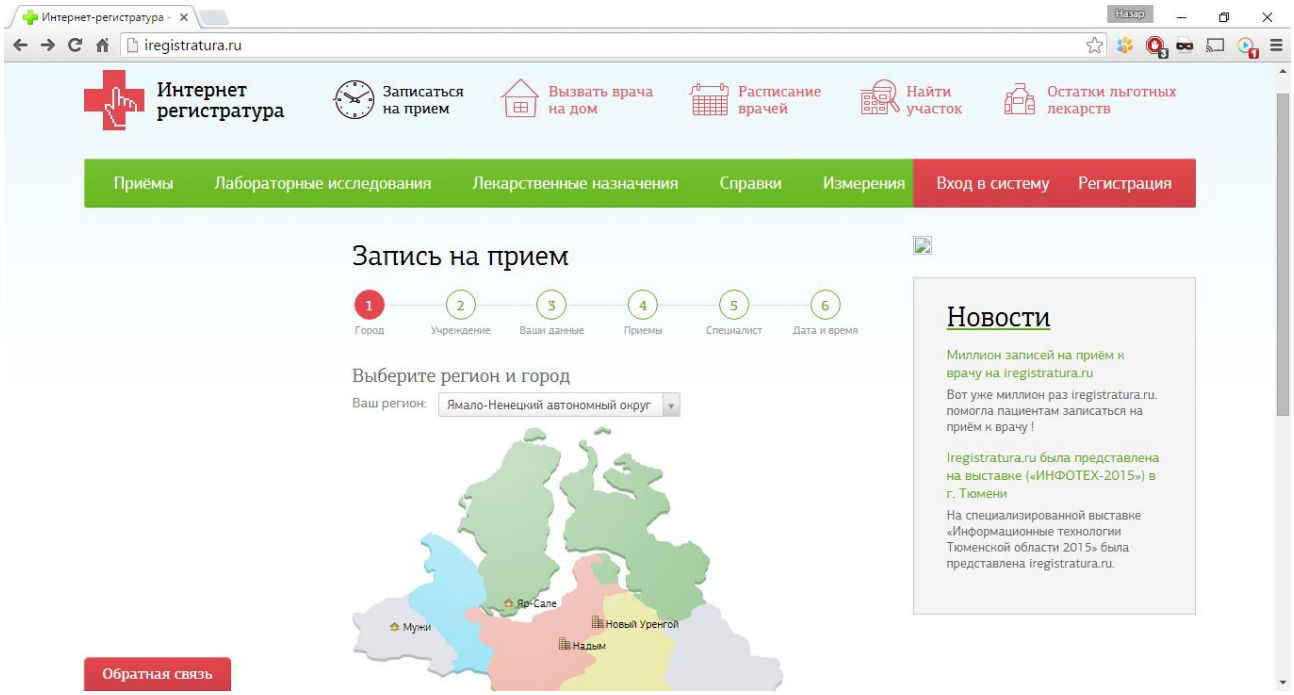

Рис. 1.5. Головна сторінка веб-сайту системи “Веб-реєстратура”

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, як показано на рисунку 1.6.

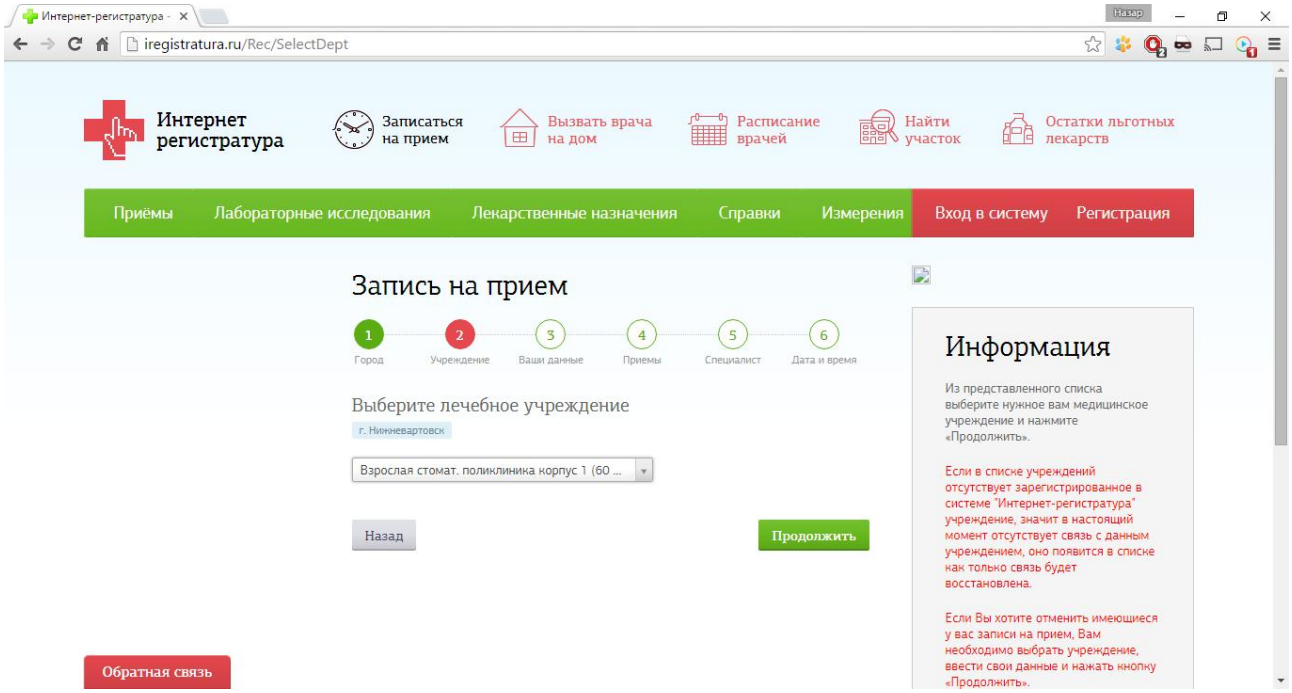

Рис. 1.6. Сторінка вибору медичного закладу.

Після підтвердження вибору медичного закладу користувач заповнює форму, де необхідно вказати особисті дані. Форма для заповнення особистих даних зображена на рисунку 1.7.

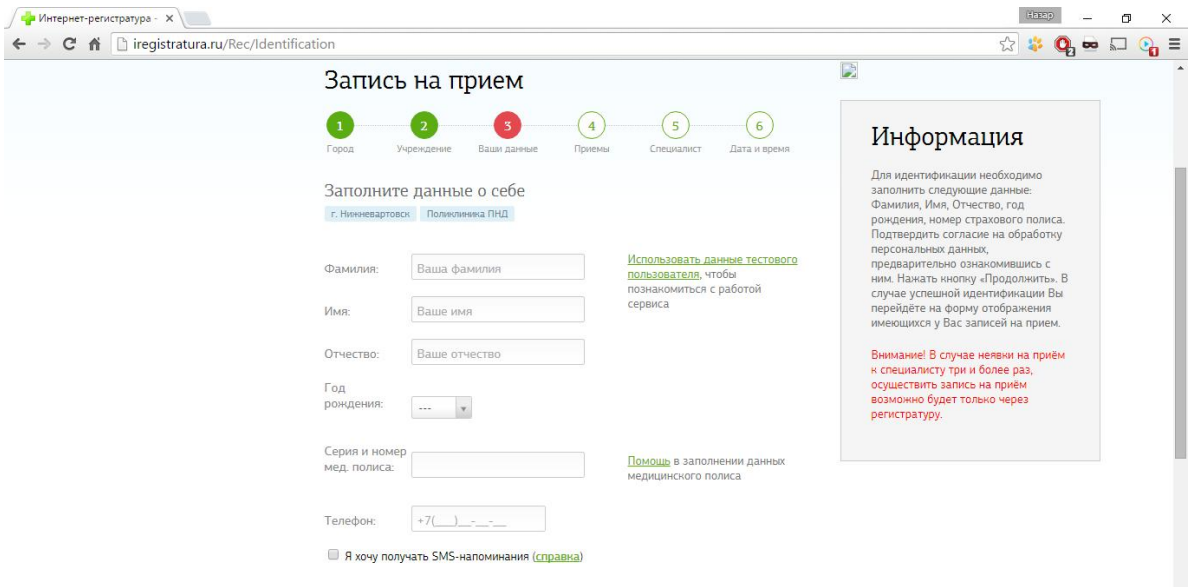

Рис. 1.7. Сторінка з формою для введення особистих даних

Вказавши особисті дані, користувачу залишається обрати потрібного йому спеціаліста, дату та час прийому.

Інтернет-портал для самостійного запису населення на прийом до медичних установ Архангельської області (рисунок 1.8), режим доступу: (https://www.zdrav29.ru/).

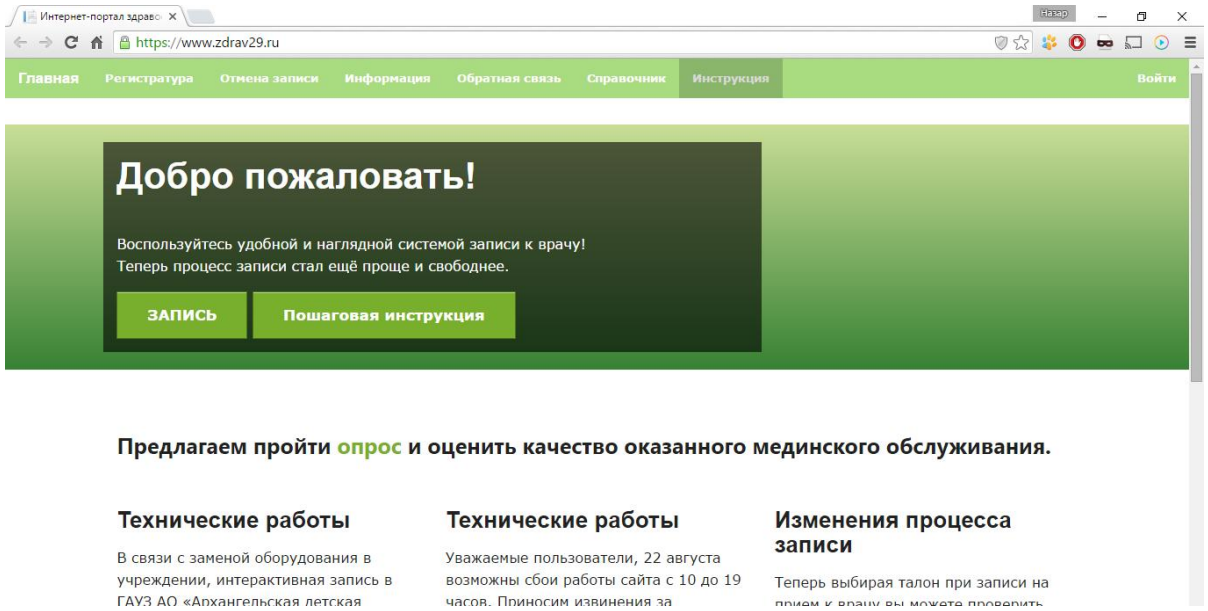

Рис. 1.8. Головна сторінка порталу охорони здоров’я Архангельської області

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. Сторінку зі сформованим талоном зображено на рисунку 1.9.

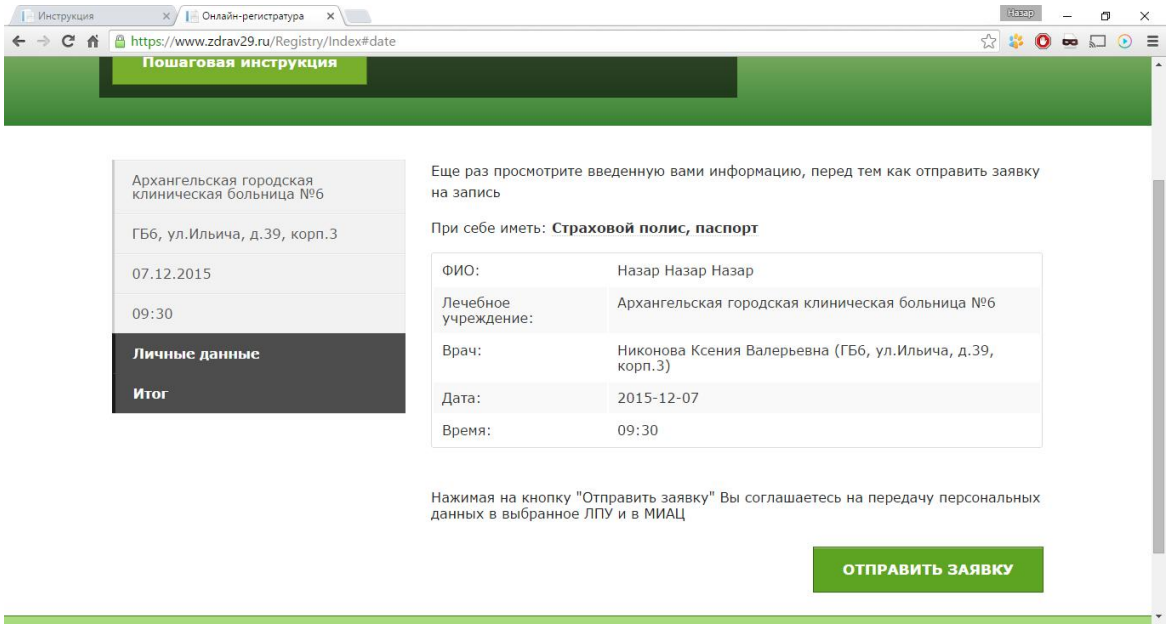

Рис. 1.9. Сторінка з талон на прийом

“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. Головна сторінка сайту “Веб-реєстратура” зображена на рисунку 1.10.

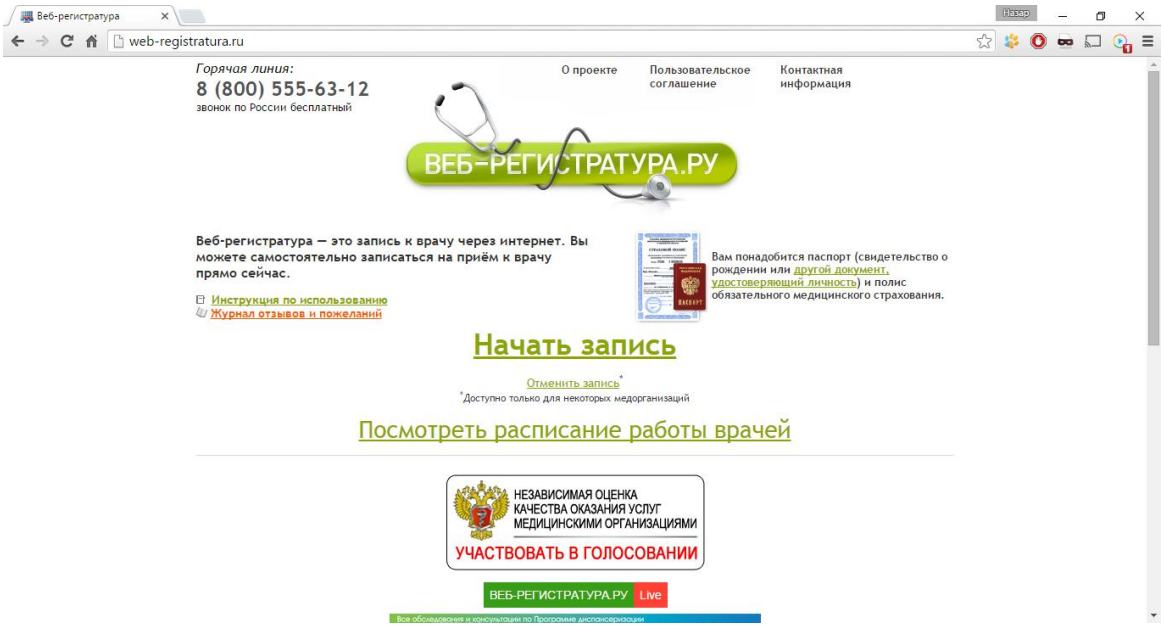

Рис. 1.10. Головна сторінка сайту “Веб-реєстратура”

Для того, щоб записатися на прийом з допомогою даного сайту потрібно перейти за посиланням “Почати запис”. Після цього на сторінці з’явиться спадний список, в якому потрібно буде обрати населений пункт та вибрати медичний заклад з переліку. Сторінка вибору медичного закладу зображена на рисунку 1.11.

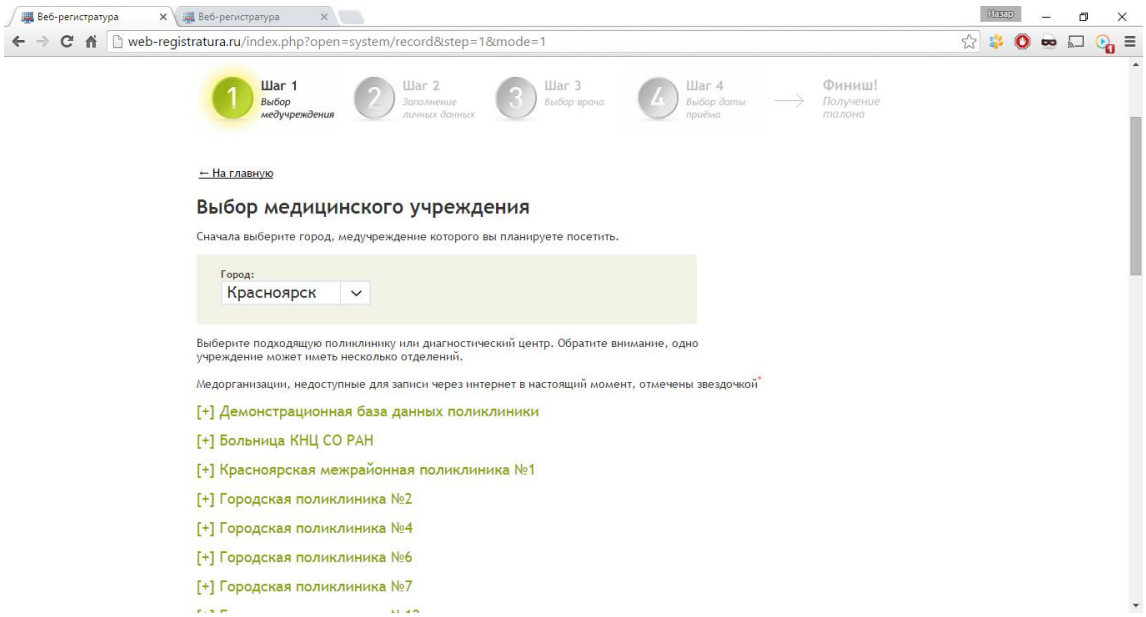

Рис. 1.11. Сторінка вибору медичного закладу

Другим кроком є введення номеру паспорта пацієнта та номеру страхового полісу. Форма вводу цих даних зображена на рисунку 1.12.

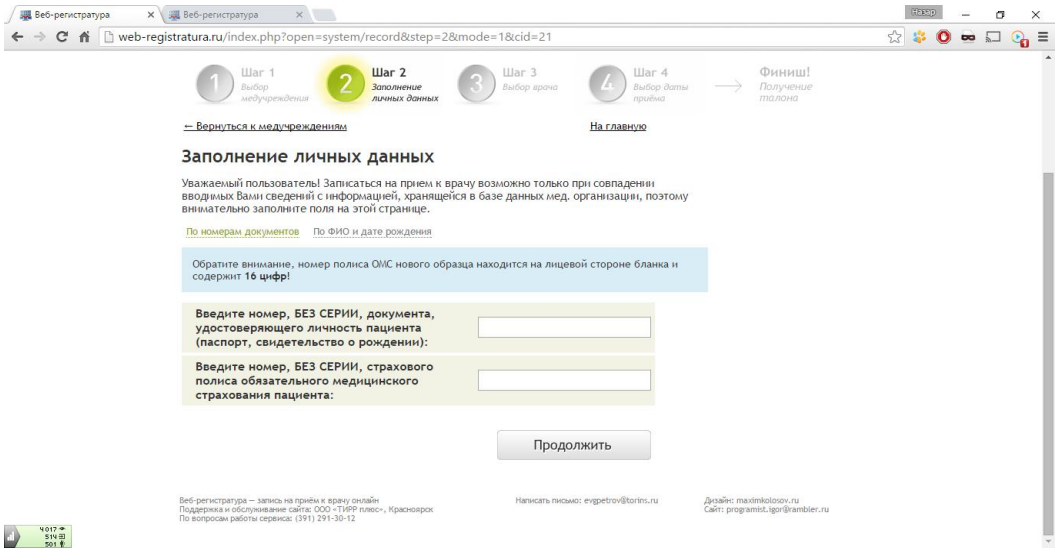

Рис. 1.12. Сторінка для вибору медичного закладу

Після цього користувач обирає спеціальність та особу потрібного йому лікаря і переходить на сторінку вибору дати прийому. На останньому кроці користувач перевіряє та підтверджує правильність інформації, після чого йому видається талон, який він може зберегти та роздрукувати.

Порівняльна характеристика оглянутих програмних продуктів знаходиться в таблиці 1.5.

Таблиця 1.5

Порівняльна характеристика програмних продуктів

Назва програмного продукту	“Інтернет-реєстратура – запис на прийом до лікаря”	“Інтернет-портал охорони здоров’я Архангельської області”	“Веб-реєстратура”
Фірма розробник	“КОМТЕК”	Отдел программного обеспечения, МИАЦ Архангельск	“ТИРР плюс”
Реєстрація в системі	+	+	-
Перегляд новин	+	+	-
Перегляд графіку роботи лікарів	+	-	+

Продовження таблиці 1.5

Запис на прийом	+	+	+
Пошук медичного закладу	+	-	-
Зворотній зв’язок	+	+	-
Інтерфейс користувача	Інтерфейс користувача є простим для розуміння та інтуїтивно зрозумілим.	Інтерфейс користувача є простим та зручним у використанні.	Інтерфейс користувача не дуже зручний для використання.
Допомога користувачу	-	+	+

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 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.

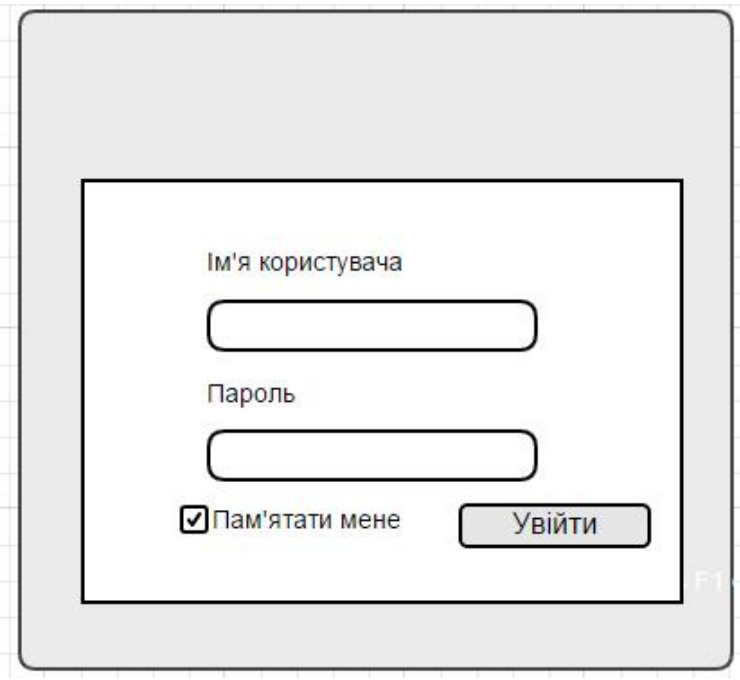

Рис. 1.14. Розкадровка сторінки авторизації

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.

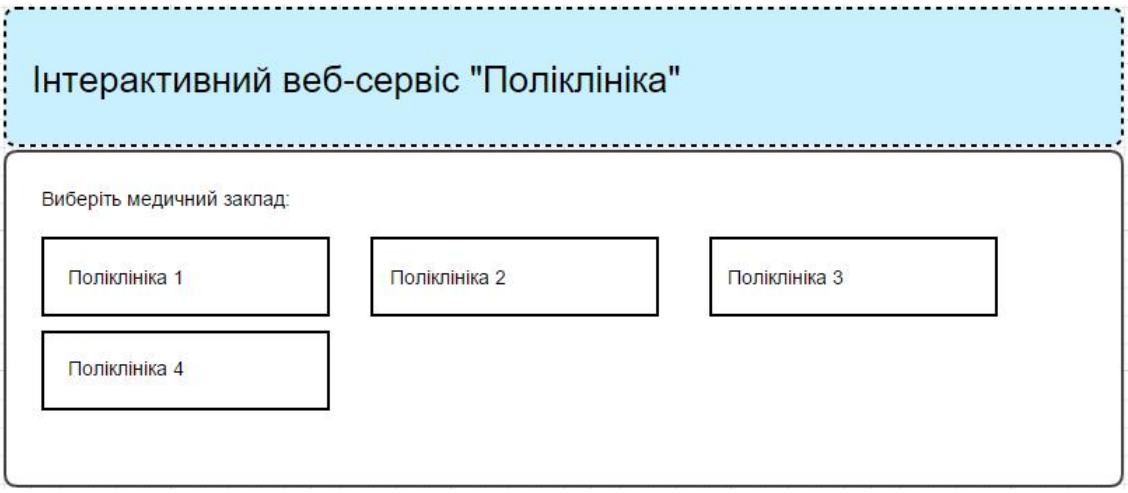

Рис. 1.15. Розкадровка сторінки зі списком медичних закладів

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

Продовження таблиці 1.10

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.

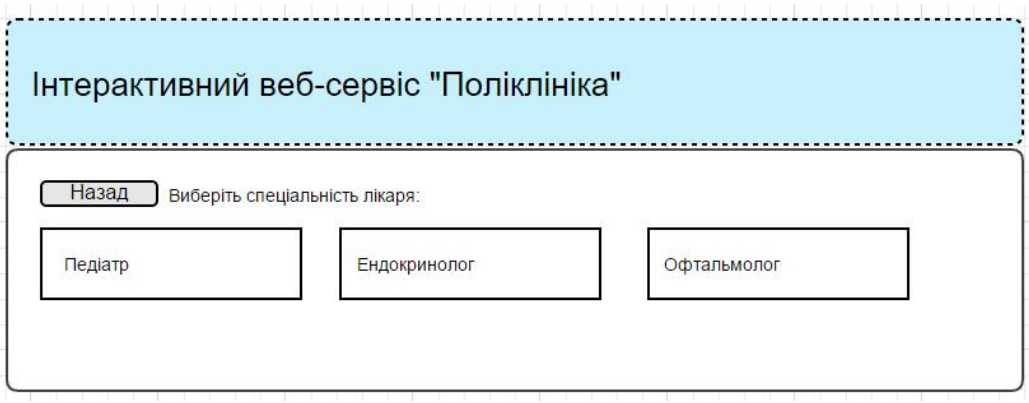

Рис. 1.16. Розкадровка сторінки зі списком спеціальностей лікарів

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

Продовження таблиці 1.13

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	Не більше 30 с
2.	Надійність	
2.1	Середній час безвідмовної роботи	100 год
2.2	Середнє напрацювання до ремонту	1 рік
2.3	Максимальна норма помилок або дефектів	100
3	Робочі характеристики	

Продовження таблиці 1.16

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	Бібліотека 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.

Дла кращого розуміння розроблюваного продукту було порівняно три аналогових продукти: “Веб-реєстратура”, “Інтернет-портал охорони здоров’я Архангельської області” 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 «Поліклініка» 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.

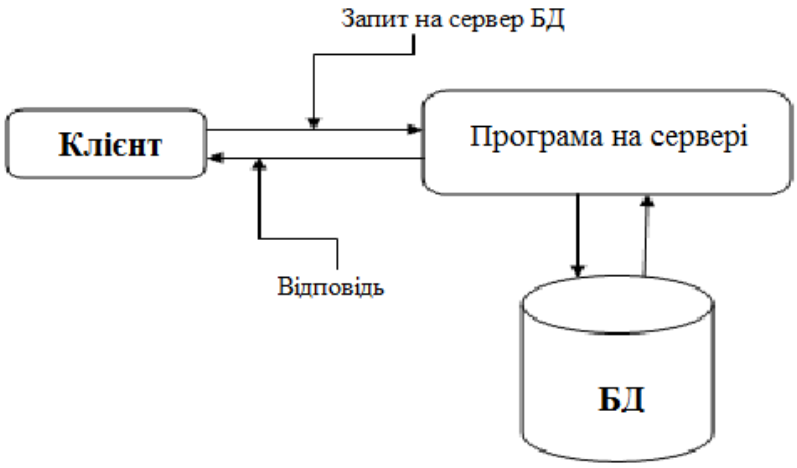

Рис. 2.1. Схема клієнт-серверної архітектури

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.

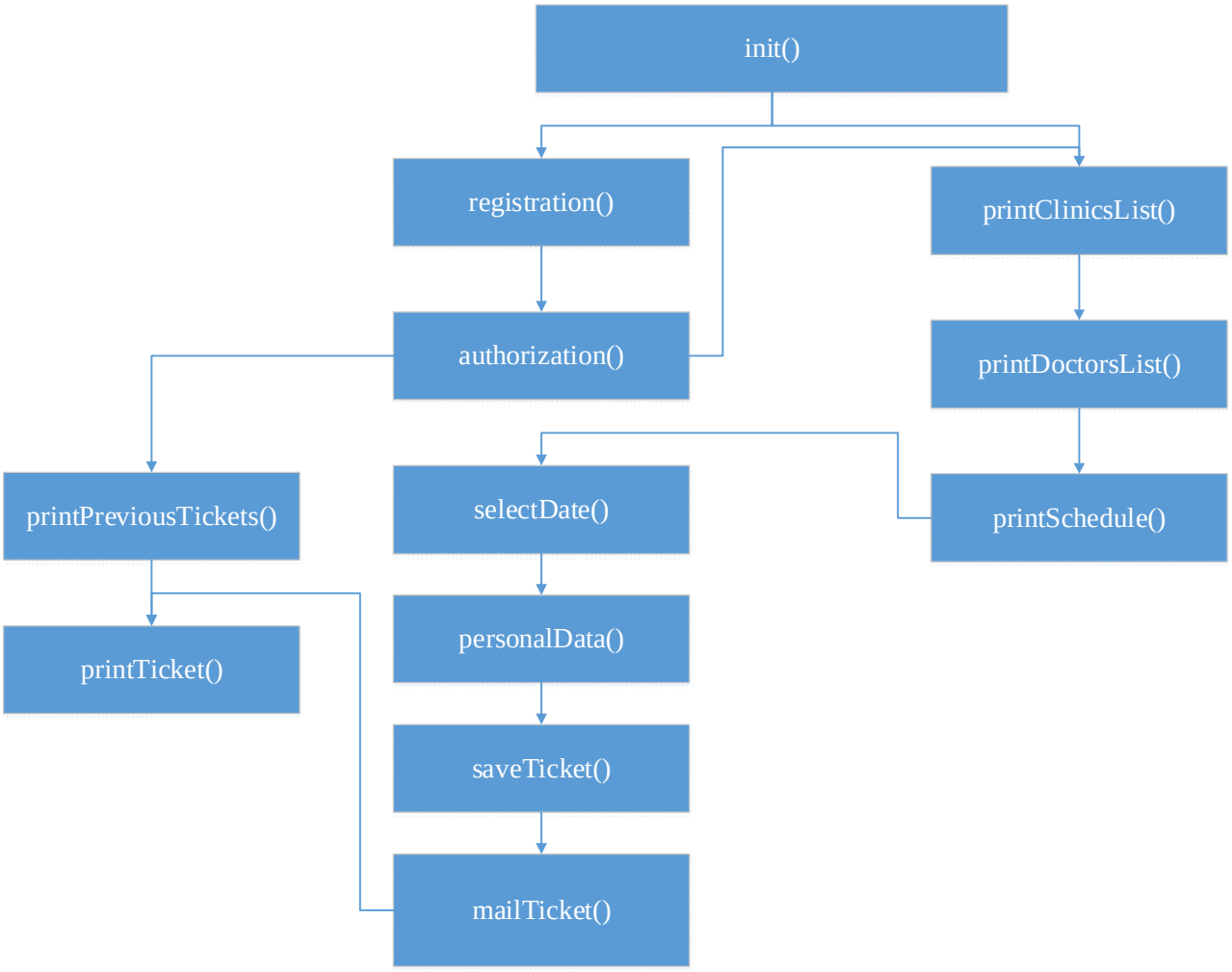

Рис. 2.2. Діаграма ієрархії функцій системи

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.

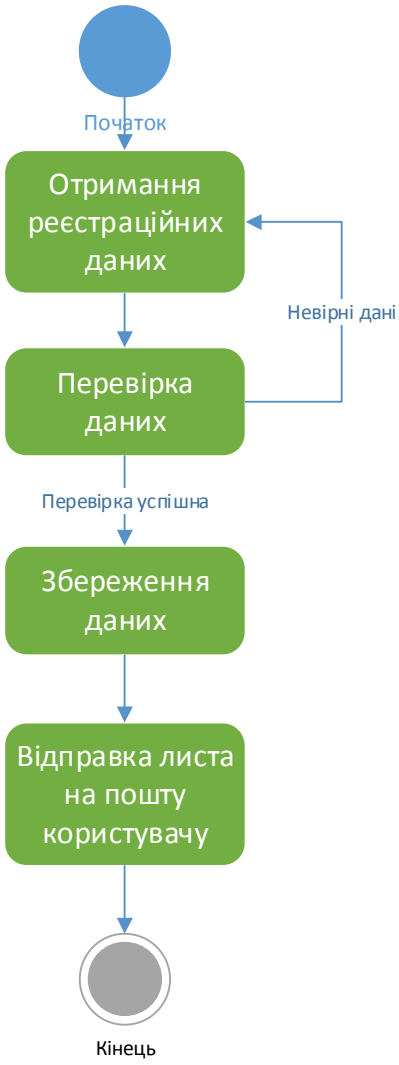

Рис. 2.3. Діаграма станів процесу реєстрації в системі

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.

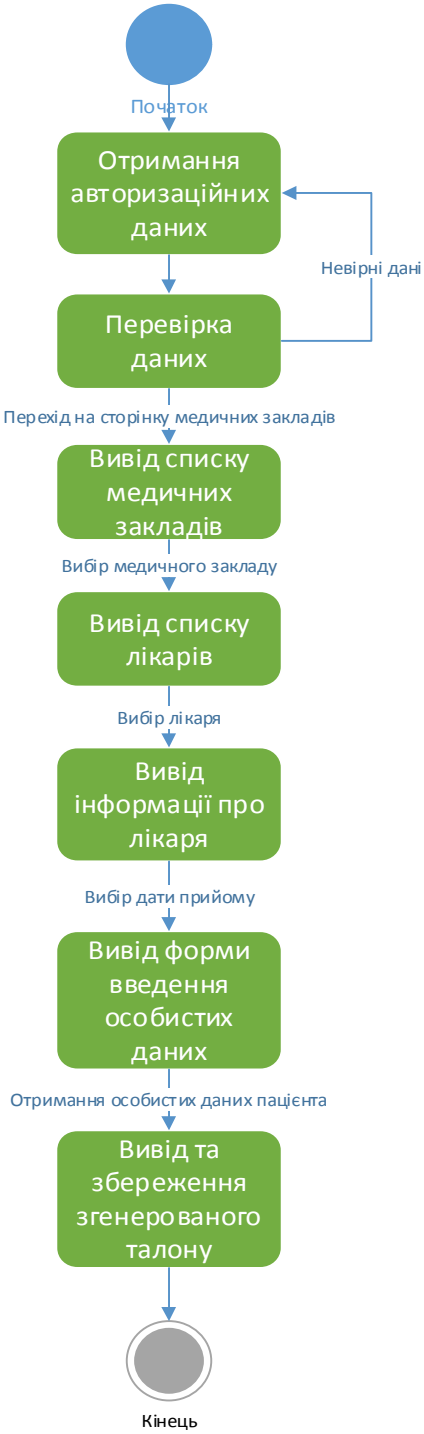

Рис. 2.4. Діаграма станів процесу реєстрації на прийом

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.

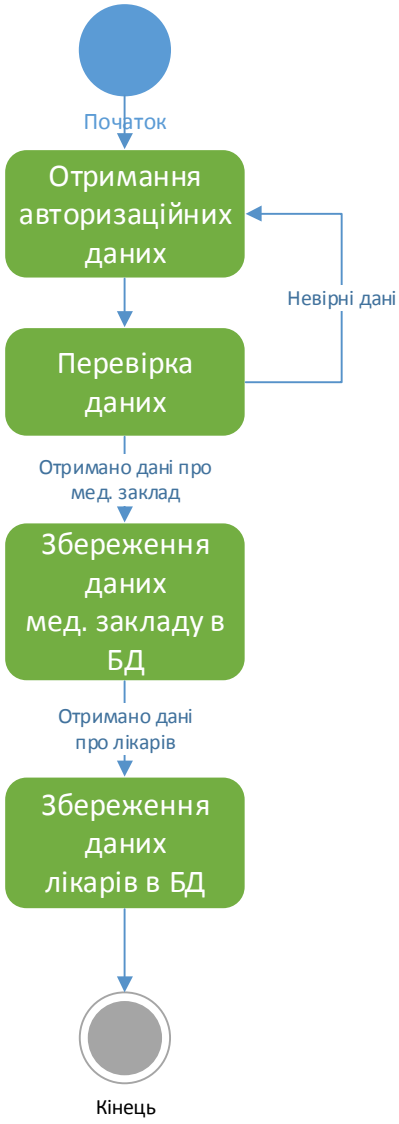

Рис. 2.5. Діаграма станів процесу “Управління даними медичного закладу”

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].

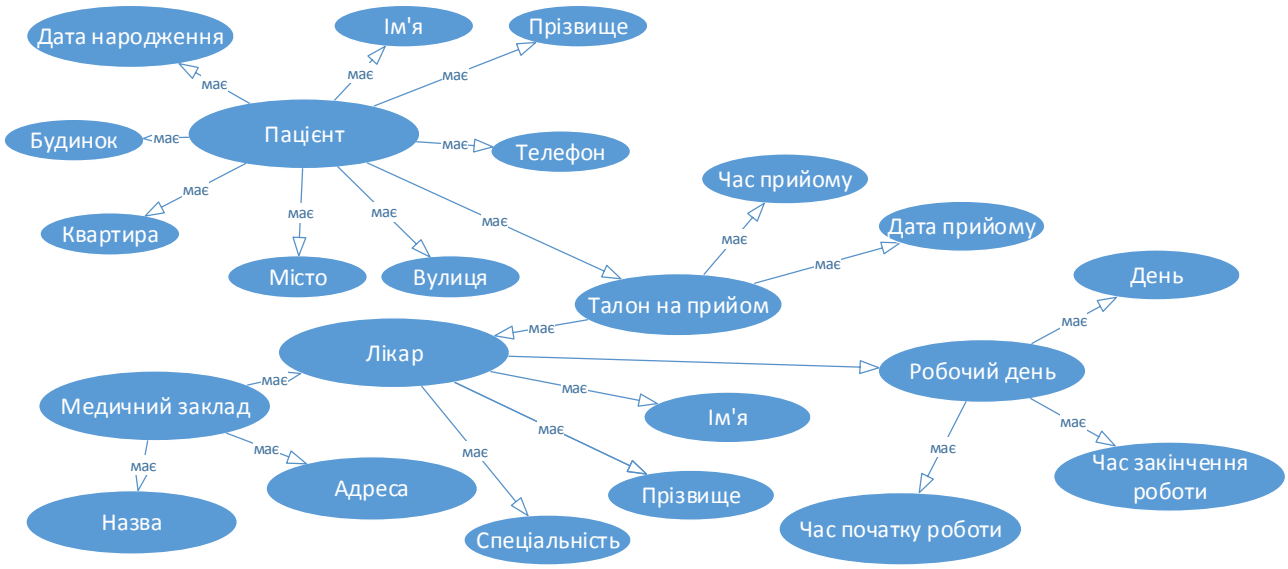

Рис. 2.6. Діаграма корпоративної моделі даних

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].

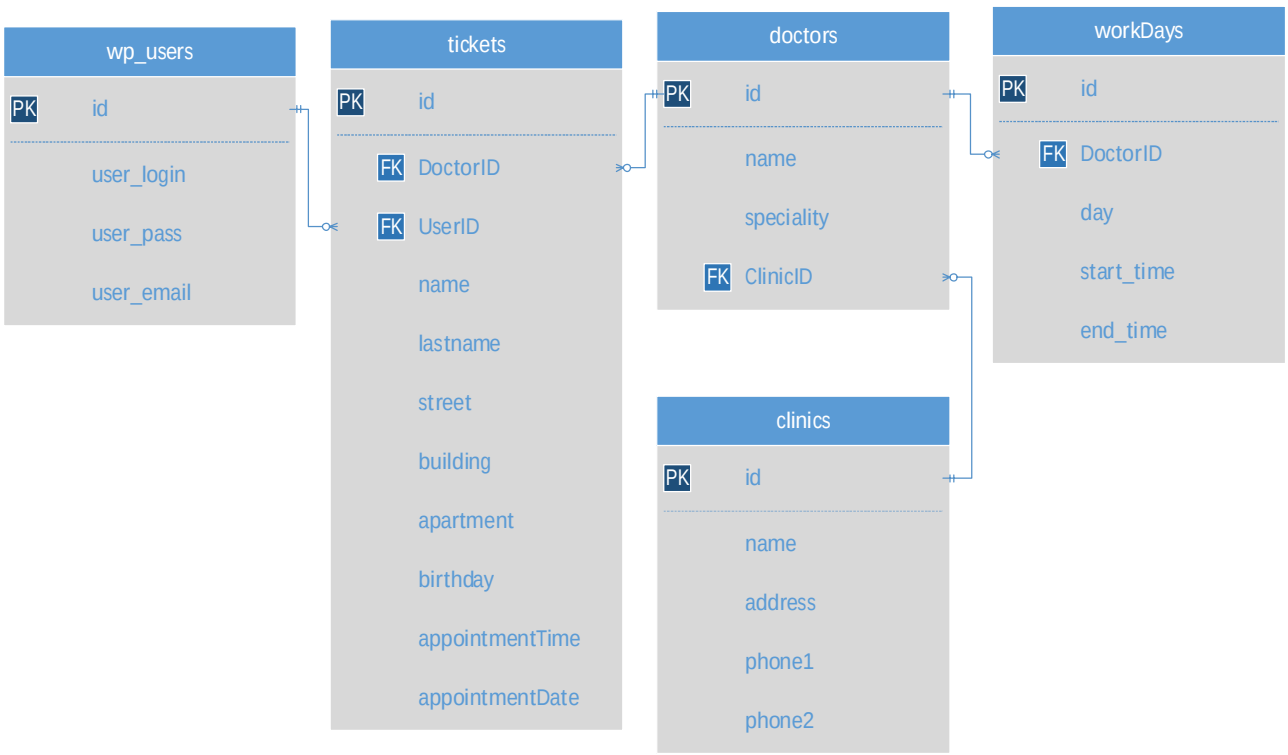

Рис. 2.7. ER – діаграма

На ER – діаграмі зображені наступні відношення: User, Ticket, Doctor, Clinic, WorkDay.

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

Нижче наведено DDL код для створення однієї таблиці в БД.

```
DROP TABLE IF EXISTS `clinics`;
CREATE TABLE IF NOT EXISTS `clinics` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `address` varchar(100) NOT NULL,
  `phone1` varchar(20) NOT NULL,
  `phone2` varchar(20) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;
```

Весь код для створення таблиць знаходиться у додатку В.

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(60);

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*	

Рис. 2.12. Структура таблиці «workdays»

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 фунцій 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

Оскільки розроблювана система є веб-орієнтованою, то в якості фундаменту розробки доцільно використати систему управління контентом. Для даного проекту мною було обрано систему управління контентом Wordpress [18].

WordPress — безкоштовна система управління контентом з відкритим кодом, створена на основі PHP та MySql. Wordpress встановлюється на веб-сервер, він може бути як частиною сервісу, який надається веб-хостингом так і мережевим хостом самостійно. Особливостями даної системи управління контентом є власна архітектура плагінів та багаторівнева ієрархія шаблонів [15].

Система управління контентом потрібна для того, щоб адміністратор веб-сервісу мав можливість керувати інформацією, не вносячи вручну зміни в програмний код, а використовуючи графічний інтерфейс. Тобто адміністратору не потрібно вміти програмувати, для управління інформацією в розроблюваній системі. Використовуючи адміністративну панель сайту, адміністратор зможе додавати, редагувати, видаляти дані медичних закладів, а також виконувати попередньо перераховані дії над даними лікарів.

PHP - мова програмування, яка виконує програмний код та генерує HTML-сторінки на стороні веб-сервера. PHP є найбільшвикористовуваною 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 в CMS Wordpress використовується система управління реляційними базами даних MySQL. MySQL є системоюз 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.

Після завершення верстки шаблону, потрібно його правильно підключити до системи управління контентом, в нашому випадку до Wordpress. Для цього використовуються функції з Wordpress API [15, 20].

В wordpress для підключення каскадних таблиць стилів та скріптів написаних на мові програмування JavaSript використовується наступний механізм. Файли стилів підключаються з використанням функції wp_enqueue_style(). Скріпти підключаються з допомогою функції

wp_enqueue_script(). Нижче подано код підключення файлів стилів та скріптів до шаблону сайту:

```
function clinic_scripts() {
          //add styles
      wp_enqueue_style('stylesheet',
get_template_directory_uri().'/css/theme.css');
          //add js
      wp_enqueue_script('custom-js',
get_template_directory_uri().'/js/custom.js' );
}

add_action('wp_enqueue_scripts', cilnic_scripts');
```

По замовчуванню в Wordpress є такі типи записів: запис, сторінка, прикріплення, редакція запису та елемент навігації [15]. 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' ] ) );
    }
```

Для того, щоб вказати медичний заклад в якому працює лікар, було використано плагін “Post2Post”, який дозволяє створити зв’язок між різними типами записів в CMS Wordpress, в тому числі і між користувацькими. Код створення зв’язку між типом записів “Клініка” та типом записів “Лікар” наведено нижче.

```
function my_connection_types() {
    p2p_register_connection_type( array(
        'name' => 'doctors_to_clinics',
        'from' => 'doctor',
        'to' => 'clinic'   ) ); }
```

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. Для вибірки медичних закладів було створено новий об’єкт класу WP_Query [15, 20]. Код обробки та виводу даних медичних закладів наведено нижче (template-clinics.php).

```
$clinics = new WP_Query( array( 'post_type' => 'clinic', 'posts_per_page' => -1 )
);
<?php if ($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));
```

```
            $date = date_format(date_create($date), "Y/m/d");
            $time = esc_html(get_post_meta( $post->ID, 'visiting_time', true));
            $minutes = date_format(date_create($time), "i");
            $hours = date_format(date_create($time), "H");
            array_push($ticketsDates,   array('date'   =>   $date,   'minutes'   =>
$minutes, 'hours' => $hours));
        }
        echo  json_encode(array('doctor_schedule'  =>  $schedule,  'ticketsDates'  =>
$ticketsDates));
        wp_die();
    }
    add_action('wp_ajax_calculate_dates', 'clinic_calculate_dates');
    add_action('wp_ajax_nopriv_calculate_dates', 'clinic_calculate_dates');
    ?>
```

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. Код функції наведено нижче.

```
function afterSetDate() {
    var date = jQuery("#datepicker_fieldset input[name='chosenDate']").val();
    var d = new Date(date);
    var startTime = ""; var endTime = "";

    if(date) {
        var chosenDay = d.getDay();
        var $input = jQuery('.timepicker').pickatime({
            container: '#container-for-time',
            interval: 12,
            clear: "",
            format: 'H:i',
            formatSubmit: 'H:i',
            hiddenName: true,
            onSet: function(){ jQuery("#submitDateBtn").show(); }
        });

        var picker = $input.pickatime('picker');
        picker.set({ enable: true });
        disabledTime = [];
```

```
            for(var i=0; i<disabledDates.length; i++)
            {
                if(date == disabledDates[i].date)
                {
               disabledTime.push(new  Date(""," "  ,""  ,  disabledDates[i].hours,
disabledDates[i].minutes));
                }
            }
            picker.set({ disable: disabledTime });
            console.log(disabledTime);
            switch(chosenDay) {
                case 1:
                    picker.set({      min:      schedule.mon_start_time,      max:
schedule.mon_end_time });
                    picker.start();
                    jQuery('#timeInputWrapper').show();
                    break;
                case 2:
                    picker.set({      min:      schedule.tue_start_time,      max:
schedule.tue_end_time });
                    jQuery('#timeInputWrapper').show();
                  break;
                case 3:
                    picker.set({      min:      schedule.wed_start_time,      max:
 schedule.wed_end_time });
                    jQuery('#timeInputWrapper').show();
                    break;
                case 4:
                    picker.set({      min:      schedule.thu_start_time,      max:
 schedule.thu_end_time });
                    jQuery('#timeInputWrapper').show();
                    break;
                case 5:
                    picker.set({      min:      schedule.fri_start_time,      max:
 schedule.fri_end_time });
                    jQuery('#timeInputWrapper').show();
                    break;
                default:
                    alert("Something wrong with ajax request! Try later!");
```

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.

```
    function saveTicket($clinicName, $doctorName, $chosenDate, $chosenTime,
$patientBirthday) {
        $ticketHtml = "<div id='ticket-wrapper'>";
        $ticketHtml .= "<h3 style='text-align: center;'>"
.get_the_title($_POST['clinicID']). "</h3>";
        $ticketHtml .= "<hr><ul class='ticket-details'>";
        $ticketHtml .= "<li>
    <span class='detail-title'>Дата та час<br/>прийому</span>
<span class='detail-content'>" .date_format($chosenDate, 'd.m.Y'). " "
.$chosenTime. "</span><li>";
$ticketHtml .= "<li>
    <span class='detail-title'>Спеціальність<br/>лікаря</span>
    <span class='detail-content'>" .$_POST['speciality']. "</span>
    </li>";
    $ticketHtml .= "<li>
    <span class='detail-title'>Ім'я лікаря</span>
    <span class='detail-content'>" .$doctorName. "</span></li>";
    $ticketHtml .= "<li>
     <span class='detail-title'>Кабінет</span>
     <span class='detail-content'>" .$_POST['room']. "</span></li>";
     $ticketHtml .= "<li>
     <span class='detail-title'>Ім'я пацієнта</span>
<span class='detail-content'>" .$_POST['patientName']. "</span></li>";
     $ticketHtml .= "<li>
    <span class='detail-title'>Дата<br/>народження</span>
    <span class='detail-content'>" .date_format($patientBirthday, 'd.m.Y').
"</span></li>";$ticketHtml .= "<li>
<span class='detail-title'>Адреса<br/>проживання</span>
<span class='detail-content'>" .$_POST['patientCity']. " "
.$_POST['patientStreet']. " " .$_POST['patientBuilding']. " "
.$_POST['patientAppartment']. "</span></li>";
$ticketHtml .= "<li>
```

```
<iclass='icon-location'></i><span>".esc_html(get_post_meta($_POST['clinicID'],
'clinic_address', true)). "</span>
<br>
     <iclass='icon-phone'></i><span>Тел:                                                    "
.esc_html(get_post_meta($_POST['clinicID'],         'clinic_phone1',         true)).
"; ".esc_html(get_post_meta($_POST['clinicID'],     'clinic_phone2',     true)).
"</span></li></ul></div>";
         echo $ticketHtml;
         $post_data = array(
             'post_title'    => wp_strip_all_tags($_POST['patientName']),
             'post_status'   => 'publish',
             'post_author'   => get_current_user_id(),
             'post_type'     => 'ticket'
         );

         $post_id = wp_insert_post($post_data, true);

         $current_user = wp_get_current_user();

         $email = $current_user->user_email;
         $to = $email;
         $subject = "Талон на прийом до лікаря";
       $headers = "MIME-Version: 1.0" . "\r\n";
       $headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";

       mail($email, $subject, $ticketHtml, $headers);
   }
```

Також були реалізовані функції відправлення талону на електронну пошту користувача та функція друку талону. Код функції, що відправляє талон на пошту наведено нижче.

```
$current_user = wp_get_current_user();
    $email = $current_user->user_email;
    $to = $email;
    $subject = "Талон на прийом до лікаря";
    $headers = "MIME-Version: 1.0" . "\r\n";
    $headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";
    mail($email, $subject, $ticketHtml, $headers);
```

Код функції друку талону наведено нижче.

```
function print(ele) {
    var openWindow = window.open("", "Талон", "width=1000, height=600");
    openWindow.document.write("<link rel='stylesheet'
href='http://127.0.0.1/clinic/wp-content/themes/betheme/css/custom.css?ver=7.0'
type='text/css'>");
    openWindow.document.write("<div id='ticket-wrapper' class='without-
shadow'>");
    openWindow.document.write(ele.previousSibling.innerHTML);
    openWindow.document.write("</div>");
    openWindow.document.close();
    openWindow.focus();
    setTimeout(function(){ openWindow.print(); openWindow.close(); }, 1);
}
```

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).

Весь програмний код подано у додатку З.

3.2. Програмна реалізація бази даних

Оскільки система управління контентом Wordpress для збереження даних використовує MySQL, то для управління даними веб-сервісу було взято саме цю систему управління реляційними базами 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 Wordpress засобами мови програмування PHP підключається до сервера бази даних та обирає вказану базу даних.

Для роботи з даними які є в таблицях бази даних в CMS Wordpress 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.

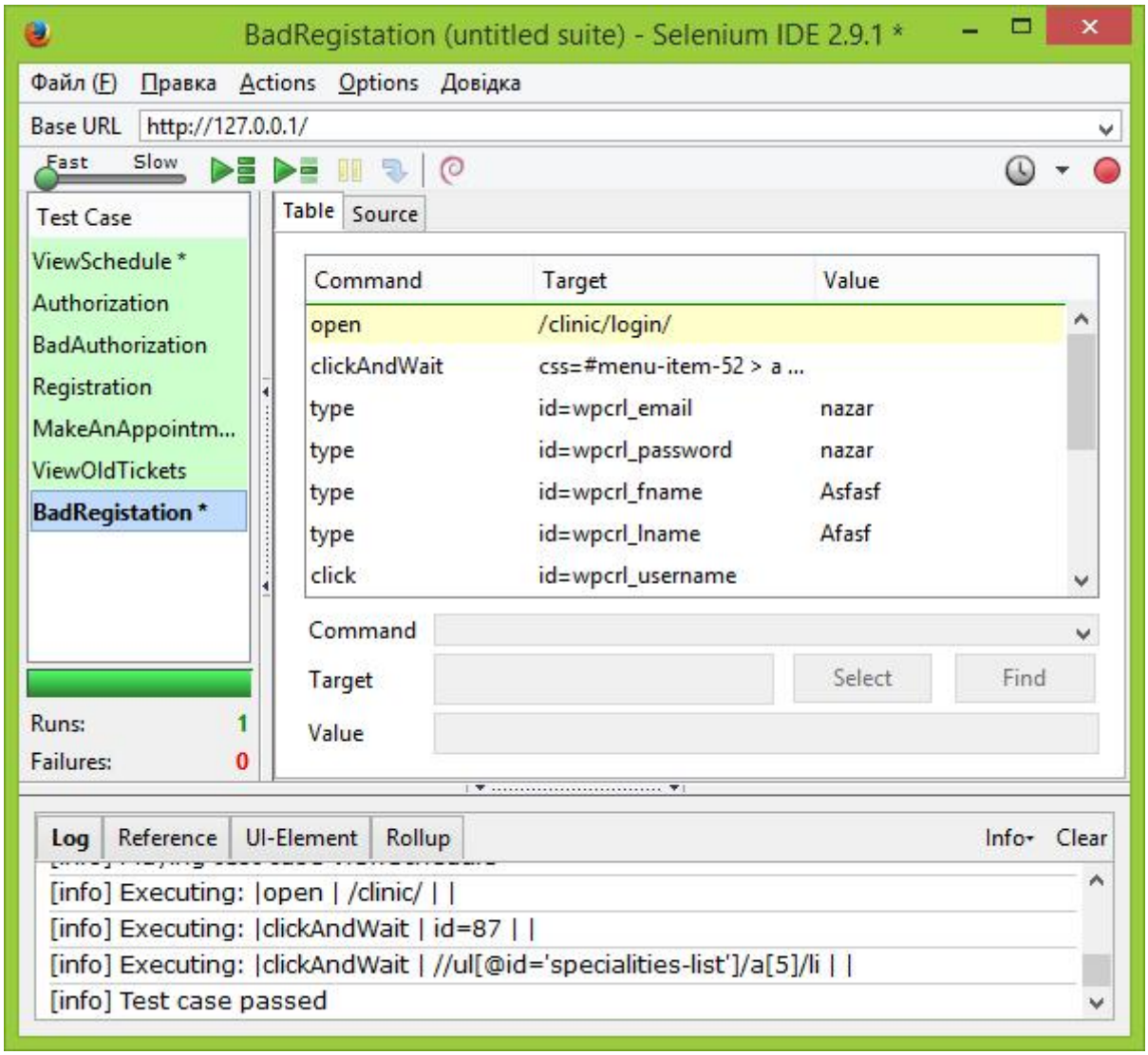

Рис. 4.1. Результат проведення GUI-тестування

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 некоретних 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.

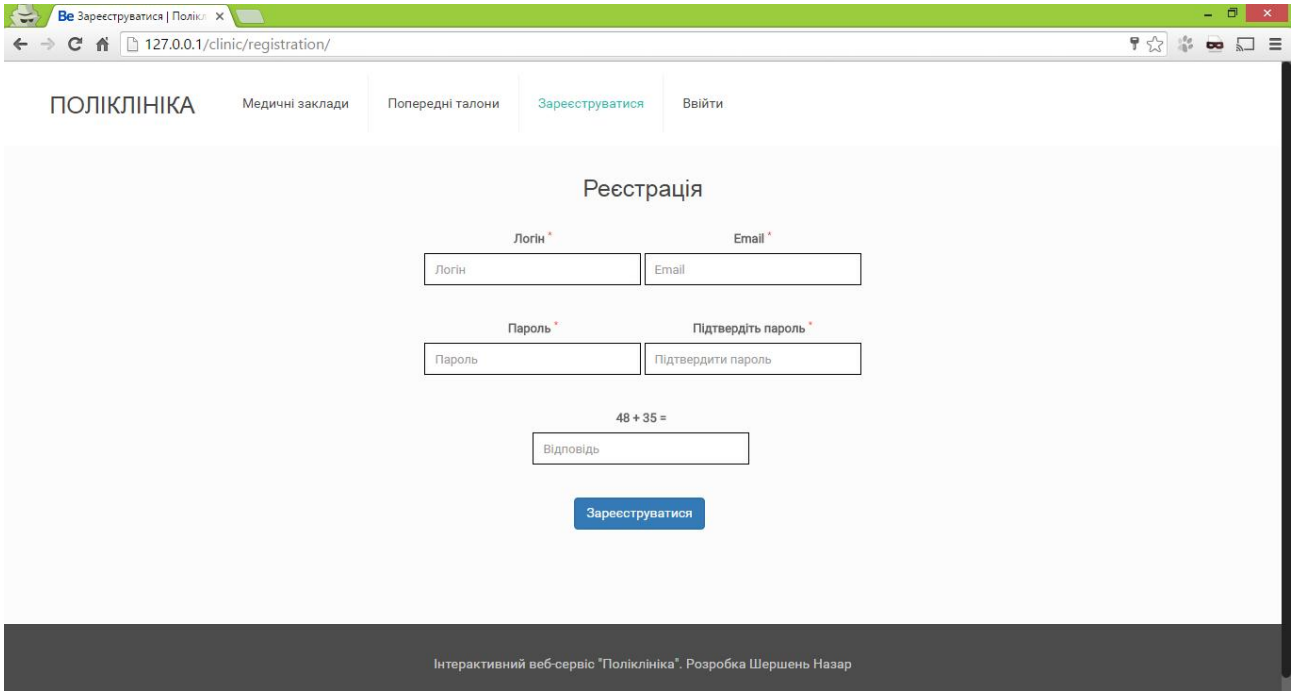

Рис. 4.2. Сторінка реєстрації

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.

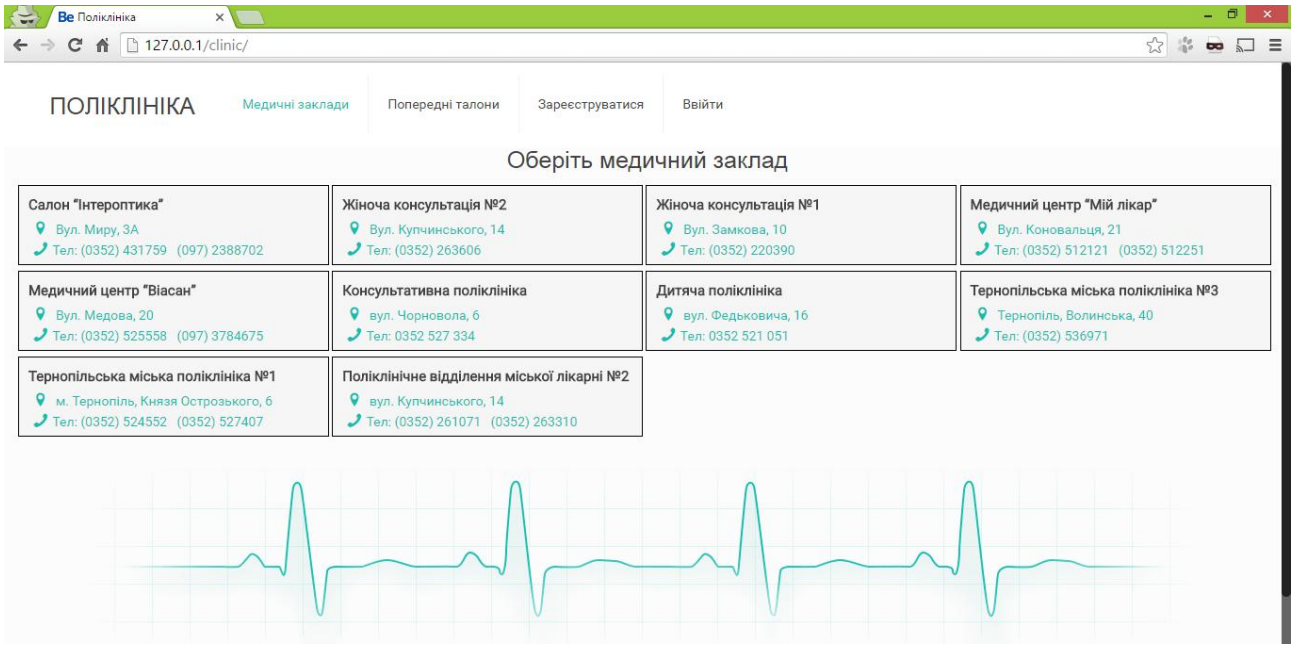

Рис. 4.3. Сторінка “Медичні заклади”

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

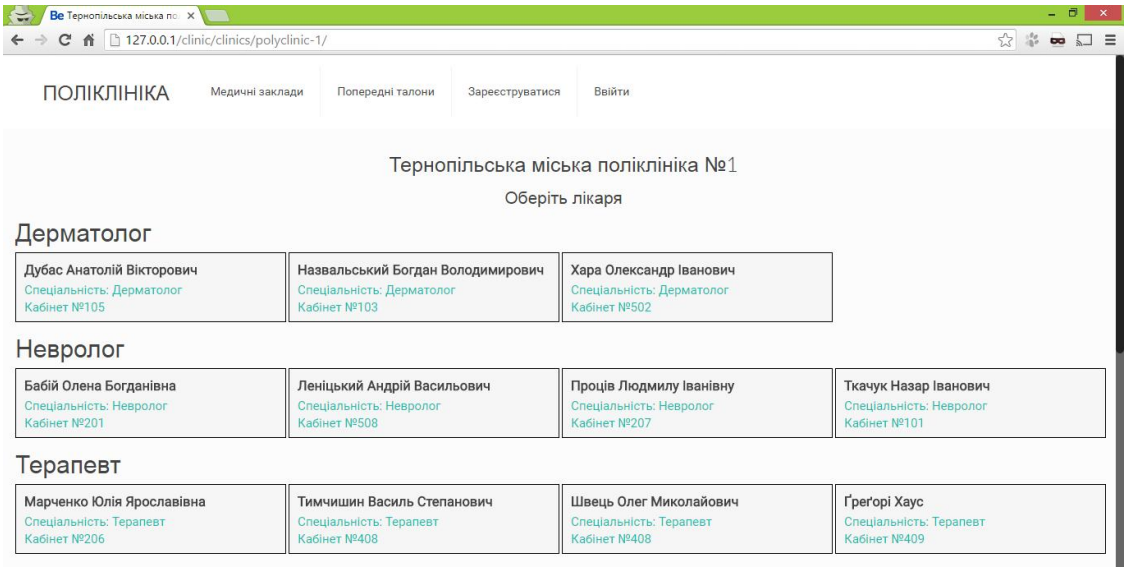

Рис. 4.4. Сторінка зі списком лікарів

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

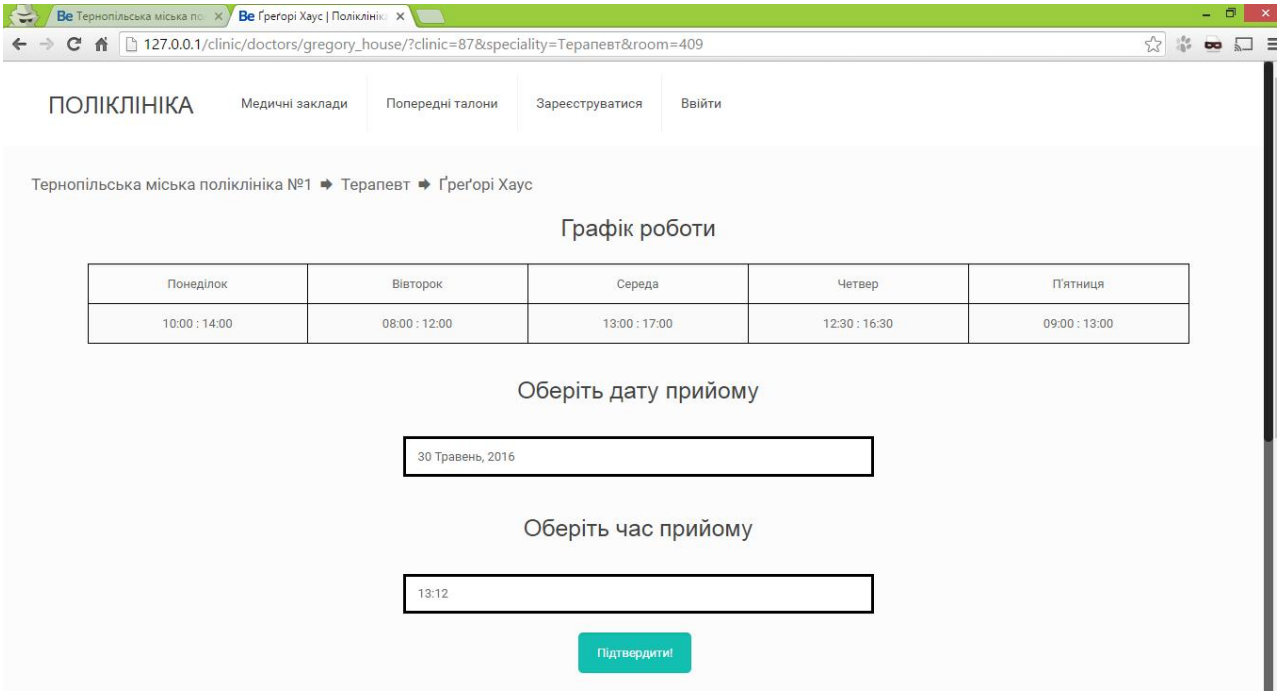

Рис. 4.5. Вибір дати і часу прийому

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.

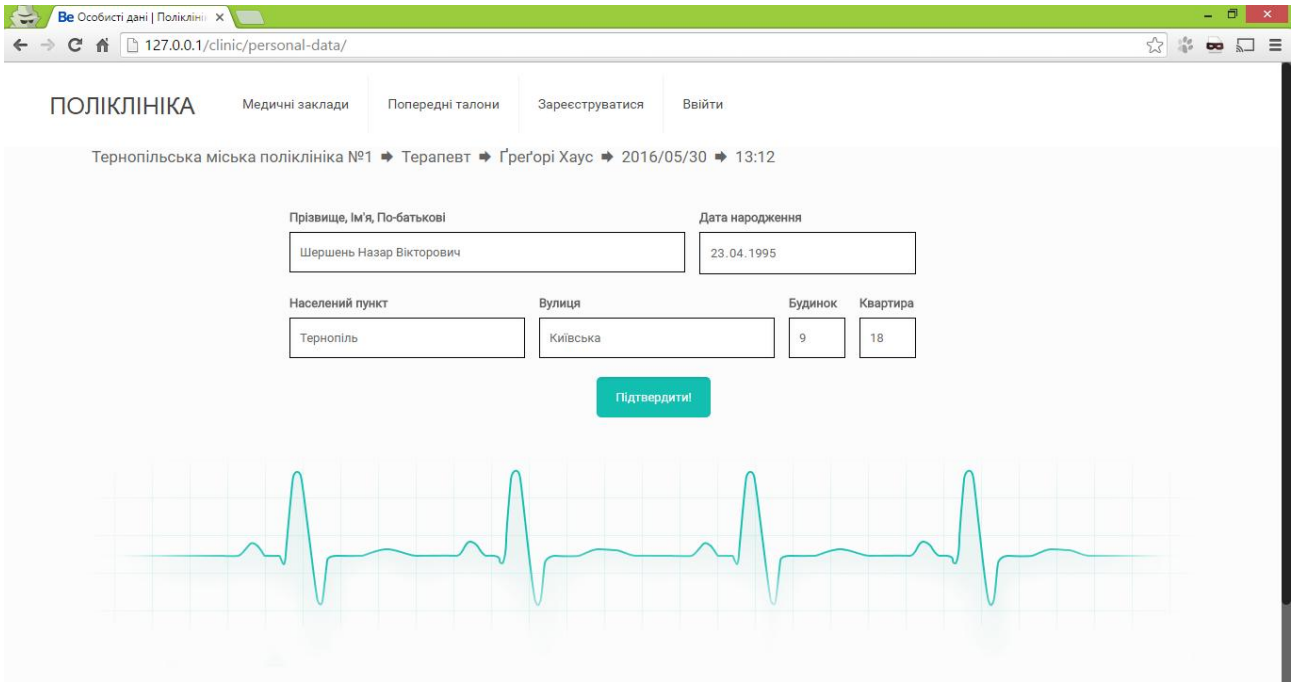

Рис. 4.6. Вибір дати і часу прийому

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

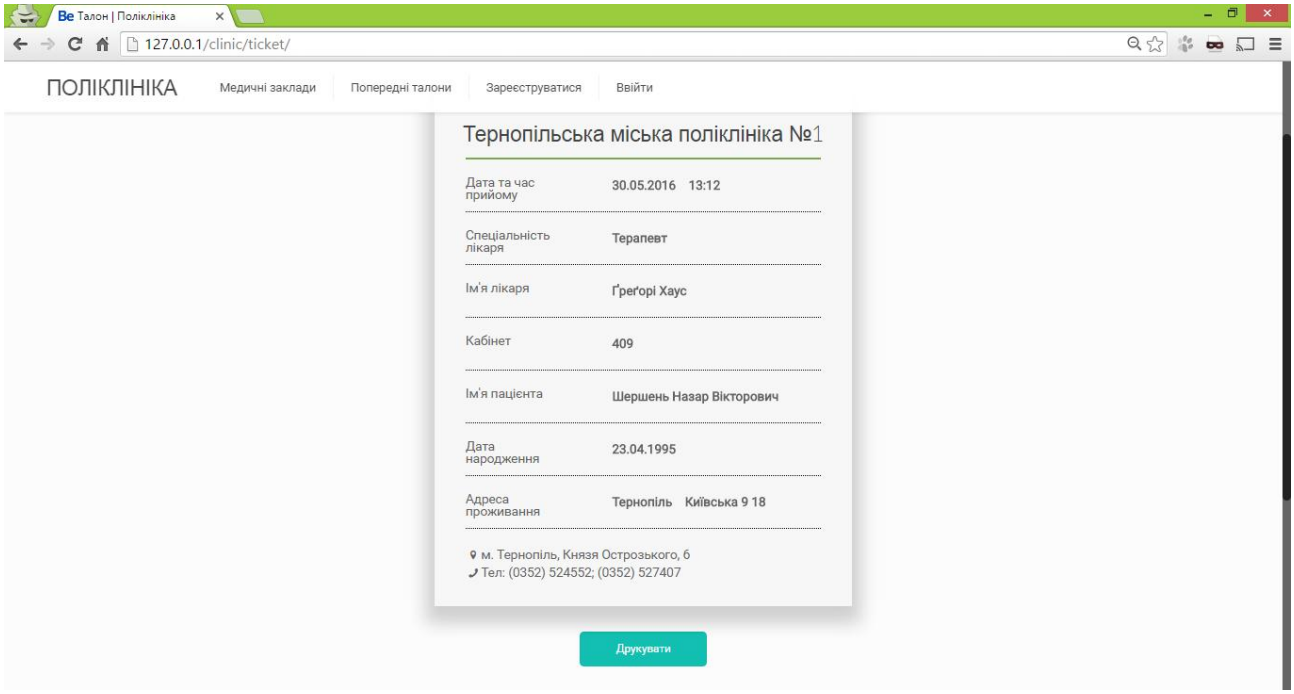

Рис. 4.7. Вибір дати і часу прийому

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).

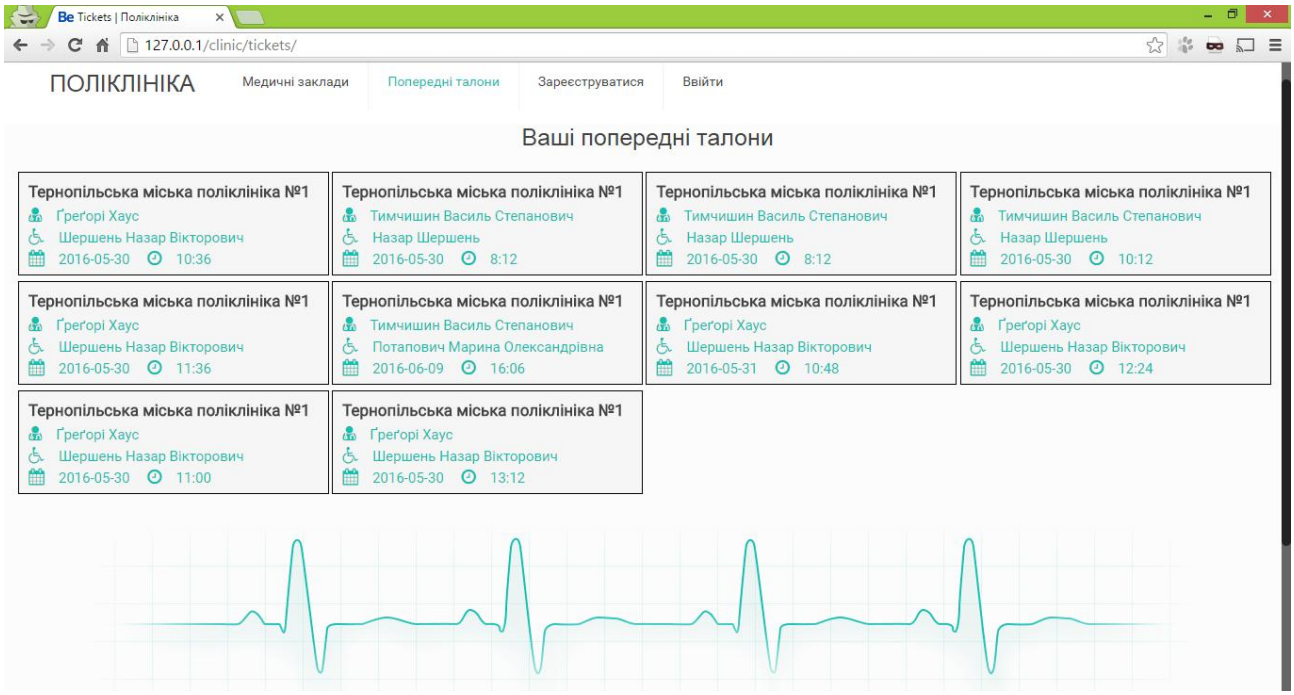

Рис. 4.8. Перегляд історії талонів

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.

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 “Поліклініка”. 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. Отже, інтерактивний веб-сервіс «Поліклініка» готовий до використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Буч Г. Язык UML: руководство пользователя / Г. Буч, Д. Рамбо, И. Якобсон // [пер. с англ. Н. Мухина]. – Москва. – 2007. – 493 с.
2. Громадське здоров'я та охорона здоров'я. Под ред. В.А. Міняєва, Н.І. Вишнякова М. «Вища школа»., 2002. - С. 173-194.
3. Дейт К. Введение в системы баз даных /К.Дейт// – К.; М.; СПб.: Изд.дом “Вильямс”. – 2000. –560 с.
4. Дивак М.П. Системний аналіз та проектування КІС /М.П.Дивак// Навчальний посібник – Т.: Економічна думка. – 2004.
5. Калбертсон Роберт, Браун Крис, Кобб Гэри Быстрое тестирование. — М.: «Вильямс», 2002. — 374 с.
6. Квентин Зервас. Web 2.0: создание приложений на PHP = Practical Web 2.0 Applications with PHP. — М.: «Вильямс», 2009. — С. 544.
7. Костарев А. Ф. PHP 5. — Спб.: «БХВ-Петербург», 2008. — С. 1104.
8. Кузнецов Максим, Симдянов Игорь. PHP на примерах. — 2-е изд. перераб. и доп. — Спб.: «БХВ-Петербург», 2011. — С. 400.
9. Кузнецов Максим, Симдянов Игорь. PHP. Практика создания Web-сайтов. — 2-е изд. перераб. и доп. — Спб.: «БХВ-Петербург», 2008. — С. 1264.
10. Лайза Криспин, Джанет Грегори Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд = Agile Testing: A Practical Guide for Testers and Agile Teams. — М.: «Вильямс», 2010. — 464 с
11. Мэтт Зандстра. PHP: объекты, шаблоны и методики программирования, 3-е издание = PHP Objects, Patterns and Practice, Third Edition. — М.: «Вильямс», 2010. — С. 560
12. Синицын С. В., Налютин Н. Ю. Верификация программного обеспечения. — М.: БИНОМ, 2008. — 368 с.
13. Юр'єв В.К., Куценко Г.І. Громадське здоров'я та охорона здоров'я. С-П, 2000. - С. 240-283.

14. jQuery API documentation [Електронний ресурс]. - Режим доступу: http://api.jquery.com/

15. Online manual for WordPress [Електронний ресурс]. - Режим доступу: https://codex.wordpress.org/

16. PHP documentation [Електронний ресурс]. - Режим доступу: http://php.net/docs.php/

17. Pickadate.js API documentation [Електронний ресурс]. - Режим доступу: http://amsul.ca/pickadate.js/api/

18. Scott, Adam D. WordPress for Education. — Birmingham: Packt Publishing Ltd, 2012. — 144 c.

19. Web developer information website [Електронний ресурс]. - Режим доступу: http://www.w3schools.com/

20. WordPress как на ладони [Електронний ресурс]. - Режим доступу: http://www.wp-kama.ru/

ДОДАТОК А ГЛОСАРІ ПРОЕКТУ

Таблиця А.1

Глосарій

Термін	Опис терміну
1. Основні поняття та категорії предметної області та проекту.	
Веб-сайт	Сукупність веб-сторінок, доступних у мережі Інтернет, які об'єднані як за змістом, так і за навігацією. Фізично сайт може розміщуватися як на одному, так і на кількох серверах.
Веб-система	Клієнт-серверний додаток, в якому клієнтом виступає браузер, а сервером - веб-сервер.
База даних	Впорядкований набір логічно взаємопов'язаних даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів.
Авторизація	Керування рівнями та засобами доступу до певного захищеного ресурсу (наприклад, автоматизована система контролю доступу) та ресурсів системи залежно від ідентифікатора і пароля користувача або надання певних повноважень (особі, програмі) на виконання деяких дій у системі обробки даних.
Реєстрація в системі	Зареєстрований користувач може обрати собі ім'я (логін), яким будуть підписуватись в системі всі його редагування.
2. Користувачі системи	
Користувач	Простий користувач, який може переглядати вміст сайту (системи), але не має права доступу до більшості функцій системи.
Адміністратор	Має повні права доступу до системи її функціональних можливостей, а також маніпуляції над даними, якими оперує система.
Авторизований користувач	Користувач, який має більше прав доступу до функцій системи, але деякий контент для нього закритий.
3.Вхідні та вихідні документи	
Талон	Документ, який засвідчує запис пацієнта на прийом до лікаря. Містить особисті дані пацієнта, інформацію про лікаря, дату, час та місце прийому.

ДОДАТОК Б СПЕЦИФІКАЦІЯ ВИМОГ

Специфікація вимог до інтерактивного веб-сервісу “Полкліника”

1. Вступ

1.1 Призначення, мета

Програмне забезпечення створене для реєстрації пацієнтів на прийом до лікаря через Інтернет. Мета даного ПЗ спростити та пришвидшити процес реєстрації пацієнта на прийом до лікаря в медичному закладі.

1.2 Продукти-аналоги

Схожими продуктами є:

- “Інтернет-реєстратура – запис на прийом до лікаря”, режим доступу (http://iregistratura.ru/);
- “Інтернет-портал охорони здоров’я Архангельської області”, режим доступу (https://www.zdrav29.ru/);
- “Веб-реєстратура”, режим доступу (http://web-registratura.ru/).

2. Загальний опис

2.1Характеристики продукту

Оскільки ПЗ повинно стандартизувати і полегшити процес реєстрації пацієнта на прийом в медичному закладі, воно має в собі ряд облікових характеристик та відповідно функції, що їх реалізовують:

- додавання медичного закладу в систему;
- редагування даних медичного закладу;
- видалення медичного закладу з системи;
- реєстрація користувача в системі;
- авторизація користувача в системі;
- перегляд списку медичних закладів;
- реєстрація користувача на прийом до лікаря;
- перегляд попередніх записів на прийом.

2.2 Класи користувачів та їх характеристики

У системі присутні два класи користувачів: адміністратори та користувачі. Функціонал даних користувачів відрізняється різними рівнями доступу та можливостей у системі.

2.3 Середовище функціонування

Оскільки система є веб-додатком, то вона повинна коректно функціонувати в усіх популярних веб-браузерах: Google Chrome, Safari, Mozila FireFox, Opera, Internet Explorer, Microsoft Edge.

3. Характеристики системи

3.1. Реєстрація користувача в системі

3.1.1 Опис і пріоритет

Функціонал, що дає користувачу можливість авторизації на сайті.

3.1.2 Послідовності дія/відгук

- Відкрити сторінку реєстрації.
- Заповнити реєстраційну форму.
- Натиснути кнопку «Підтвердити».

3.1.3 Функціональні вимоги

REQ-1.1: всі поля обов’язкові для заповнення;

REQ-1.2: відправка листа з реєстраційними даними на пошту пацієнту.

3.2. Авторизація

3.2.1 Опис і пріоритет

Функція, що дає можливість подальшої реєстрації на прийом до лікаря.

3.2.2 Послідовності дія/відгук

- Відкрити сторінку з формою авторизації.
- Заповнити поля форми.
- Натиснути кнопку «Підтвердити».

REQ-2.1: всі поля обов’язкові для заповнення;

REQ-2.2: користувач отримує доступ до сторінки реєстрації на прийом.

3.3. Додавання медичного закладу в систему

3.3.1 Опис і пріоритет

Функція, яка дає можливість адміністратору додавати новий медичний заклад в систему.

3.3.2 Послідовності дій/відгук

- Відкрити адміністративну панель сайту.
- Перейти на сторінку додавання медичного закладу.
- Ввести в форму інформацію про медичний заклад.
- Додати персонал медичного закладу.
- Натиснути кнопку «Додати»

3.4. Редагування даних медичного закладу

3.4.1 Опис і пріоритет

Функція дозволяє адміністратору редагувати дані медичного закладу через.

3.4.2 Послідовності дія/відгук

- Відкрити адміністративну панель сайту.
- Перейти на сторінку зі списком медичних закладів.
- Обрати медичний заклад.
- Внести потрібні зміни.
- Натиснути кнопку «Зберегти»

3.5. Видалення медичного закладу з системи

3.5.1 Опис і пріоритет

Функція дозволяє адміністратору видалити обраний медичний заклад з системи.

3.5.2 Послідовності дія/відгук

- Відкрити адміністративну панель сайту.
- Перейти на сторінку зі списком медичних закладів.
- Обрати медичний заклад.
- Натиснути кнопку «Видалити»

3.6. Перегляд списку медичних закладів

3.6.1 Опис і пріоритет

Функція дозволяє переглядати на сайті список медичних закладів доданих в систему.

3.6.2 Послідовності дія/відгук

- Відкрити головну сторінку сайту.

REQ-6.1: на головній сторінці сайту повинен бути список всіх медичних закладів, які є в системі

3.7. Реєстрація на прийом до лікаря

3.7.1 Опис і пріоритет

Функція дозволяє користувачу записати на прийом до потрібного йому лікаря.

3.7.2 Послідовності дія/відгук

- Відкрити головну сторінку сайту.
- Обрати медичний заклад.
- Обрати спеціальність лікаря.
- Обрати особу лікаря.
- Вказати дату прийому.
- Вказати час прийому.
- Ввести особисті дані.
- Натиснути кнопку «Підтвердити»

REQ-7.1: можливість вибору дати прийому має співпадати з графіком роботи лікаря.

3.8. Реєстрація на прийом до лікаря

3.8.1 Опис і пріоритет

Функція дозволяє користувачу переглядати історію записів на прийом зі свого аккаунта.

3.8.2 Послідовності дія/відгук

- Відкрити особисту сторінку користувача.
- Перейти на сторінку перегляду історії записів.
- Натиснути кнопку «Підтвердити»

REQ-8.1: показувати користувачу тільки ті записи на прийом які здійснювалися з його аккаунта.

ДОДАТОК В ФІЗИЧНА МОДУЛЬ БАЗИ ДАНИХ

```
DROP TABLE IF EXISTS `clinics`;
CREATE TABLE IF NOT EXISTS `clinics` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `address` varchar(100) NOT NULL,
  `phone1` varchar(20) NOT NULL,
  `phone2` varchar(20) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

DROP TABLE IF EXISTS `doctors`;
CREATE TABLE IF NOT EXISTS `doctors` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `speciality` varchar(50) NOT NULL,
  `clinicID` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `clinicID` (`clinicID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

ALTER TABLE `doctors`
  ADD CONSTRAINT `doctors_ibfk_1` FOREIGN KEY (`clinicID`) REFERENCES `clinics` (`id`);

DROP TABLE IF EXISTS `tickets`;
CREATE TABLE IF NOT EXISTS `tickets` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `doctorID` int(11) NOT NULL,
  `userID` int(11) NOT NULL,
  `name` varchar(30) NOT NULL,
  `lastname` varchar(30) NOT NULL,
  `street` varchar(30) NOT NULL,
  `building` varchar(10) NOT NULL,
  `appartment` varchar(10) NOT NULL,
  `birthday` date NOT NULL,
  `appointmentDate` date NOT NULL,
  `appointmentTime` time NOT NULL,
  PRIMARY KEY (`id`),
  KEY `doctorID` (`doctorID`),
  KEY `userID` (`userID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

ALTER TABLE `tickets`
  ADD CONSTRAINT `tickets_ibfk_1` FOREIGN KEY (`doctorID`) REFERENCES `doctors` (`id`),
  ADD CONSTRAINT `tickets_ibfk_2` FOREIGN KEY (`userID`) REFERENCES `wp_users` (`ID`);

DROP TABLE IF EXISTS `wp_users`;
CREATE TABLE IF NOT EXISTS `wp_users` (
  `ID` bigint(20) UNSIGNED NOT NULL AUTO_INCREMENT,
  `user_login` varchar(60) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT '',
  `user_pass` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT '',
  `user_email` varchar(100) COLLATE utf8mb4_unicode_ci NOT NULL DEFAULT '',
  PRIMARY KEY (`ID`),
  KEY `user_login_key` (`user_login`),
  KEY `user_email` (`user_email`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=UTF-8 ;
```

```
DROP TABLE IF EXISTS `workdays`;
CREATE TABLE IF NOT EXISTS `workdays` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `doctorID` int(11) NOT NULL,
  `day` tinyint(1) NOT NULL,
  `start_time` time NOT NULL,
  `end_time` time NOT NULL,
  PRIMARY KEY (`id`),
  KEY `doctorID` (`doctorID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

ALTER TABLE `workdays`
  ADD CONSTRAINT `workdays_ibfk_1` FOREIGN KEY (`doctorID`) REFERENCES `doctors`
(`id`);
```

ДОДАТОК Д ТЕСТОВІ ВИПАДКИ ФУНКЦІОНАЛЬНОГО ТЕСТУВАННЯ

Таблиця Д.1

Тестові випадки функціонального тестування

Id	Summary	Steps to reproduse	Pre - condition	Expected result	Passed/Failed
1	Verify ability of user to log in with valid data.	1. Enter valid Login. 2. Enter valid Password 3. Click on Sign in button	1. User is located on the Log in page. 2. User has an account.	1. User is succesfully logged into account. 2. System redirects user to the Home page.	Passed
2	Verify ability of user to log in with valid Login and invalid password.	1. Enter valid Login. 2. Enter invalid Password 3. Click on Sign in button	1. User is located on the Log in page. 2. User has an account.	Message "Please enter a valid password" will appear.	Passed
3	Verify ability of user to choose medical institution.	1. Choose Ternopil City polyclinic №1	1. User is located on the Medical institution page. 2. User is authorized	There is a shift to choose doctor page.	Passed
4	Verify ability of user to choose doctor.	1.Choose Gregory House.	1. User is located on the Choose doctor page. 2. User is authorized	There is a shift to Work schedule page.	Passed
5	Verify ability of user to choose the date of reception.	1. Click on the field Choose the date of reception 2. Choose valid date. 3. Click on button Choose	1. User is located on the Work schedule page. 2. User is authorized.	In the field Choose a date appeared selected date.	Passed
6	Verify ability of user to choose the time of reception.	1. Click on the field Choose the time of reception. 2. Click on time of reception.	1. User is located on the Work schedule page. 2. User is authorized.	In the field Choose the time of reception appeared selected time.	Passed

Продовження таблиці Д.1

7	Verify ability of user to confirm the order ticket (date and time).	1. Click on the field Choose the date of reception 2. Choose valid date. 3. Click on button Choose 4. Click on the field Choose the time of reception. 5. Click on time of reception. 6. Click on button Confirm.	1. User is located on the Work schedule page. 2. User is authorized.	There is a shift to Personal Data page.	Passed
8	Verify ability of user to confirm the order ticket(Personal information).	1. Enter valid full Name 2. Choose Date of the birth 3. Enter valid City 4. Enter valid Street 5. Enter valid House 6. Enter valid Apartment 7. Click on button Confirm.	1. User is located on the Introduction of Personal Data page. 2. User is authorized.	There is a shift to page with generated ticket.	Passed
9	Verify ability of user to print ticket for admission.	1. Click on button Print 2. Click on combination of buttons Ctrl+Shift+P 3. Select a Printer 4. Click on button Print	1. User is located on the Generated ticket for admission page. 2. User is authorized.	1.Ticket successfully printed. 2. System redirects user to page with generated ticket.	Passed
10	Verify ability of user to see previous tickets.	1. Click on Previous tickets page. 2. Choose a previous ticket.	1. User is located on the Home page. 2. User is authorized.	There is a shift to page with generated ticket.	Passed

Продовження таблиці Д.1

11	Verify ability of user to register on the site with valid data.	1. Click on Register page 2. Enter valid Name 3. Enter valid Last name 4. Enter valid Login 5. Enter valid Email 6. Enter valid Password 7. Enter the same password in the Confirm your password field 8. Enter valid Captcha 9. Click on button Register	1. User is located on the Home page.	Message "Check your Email for confirmation" will appear .	Passed
12	Verify ability of user to register on the site with empy field Name.	1. Click on Register page 2. Leave empty Name 3. Enter valid Last name 4. Enter valid Login 5. Enter valid Email 6. Enter valid Password 7. Enter the same password in the Confirm your password field 8. Enter valid Captcha 9. Click on button Register	1. User is located on the Home page.	Message about required field	Passed

ДОДАТОК Е КОД АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base" href="http://127.0.0.1/" />
<title>BadRegistation</title>
</head>

<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">BadRegistation</td></tr>
</thead><tbody>
<tr>
	<td>open</td>
	<td>/clinic/login/</td>
	<td></td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>css=#menu-item-52 &gt; a &gt; span</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>id=wpcrl_email</td>
	<td>nazar</td>
</tr>
<tr>
	<td>type</td>
	<td>id=wpcrl_password</td>
	<td>nazar</td>
</tr>
<tr>
	<td>type</td>
	<td>id=wpcrl_fname</td>
	<td>Asfasf</td>
</tr>
<tr>
	<td>type</td>
	<td>id=wpcrl_lname</td>
	<td>Afasf</td>
</tr>
<tr>
	<td>click</td>
	<td>id=wpcrl_username</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>id=wpcrl_username</td>
	<td>afsafsa</td>
</tr>
<tr>
	<td>type</td>
	<td>id=wpcrl_email</td>
	<td>asfsafa</td>
</tr>
```

```
	<tr>
		<td>type</td>
		<td>id=wpcrl_password</td>
		<td>asfasf</td>
	</tr>
	<tr>
		<td>click</td>
		<td>id=wpcrl_password2</td>
		<td></td>
	</tr>
	<tr>
		<td>type</td>
		<td>id=wpcrl_password2</td>
		<td>agsgga</td>
	</tr>
	<tr>
		<td>type</td>
		<td>name=wpcrl_captcha</td>
		<td>sfgs</td>
	</tr>

	</tbody></table>
	</body>
	</html>

	<?xml version="1.0" encoding="UTF-8"?>
	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
	<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
	<head profile="http://selenium-ide.openqa.org/profiles/test-case">
	<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
	<link rel="selenium.base" href="http://127.0.0.1/" />
	<title>ViewSchedule</title>
	</head>
	<body>
	<table cellpadding="1" cellspacing="1" border="1">
	<thead>
	<tr><td rowspan="1" colspan="3">ViewSchedule</td></tr>
	</thead><tbody>
	<tr>
		<td>open</td>
		<td>/clinic/</td>
		<td></td>
	</tr>
	<tr>
		<td>clickAndWait</td>
		<td>id=87</td>
		<td></td>
	</tr>
	<tr>
		<td>clickAndWait</td>
		<td>//ul[@id='specialities-list']/a[5]/li</td>
		<td></td>
	</tr>

	</tbody></table>
	</body>
	</html>

	<?xml version="1.0" encoding="UTF-8"?>
	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
	<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
	<head profile="http://selenium-ide.openqa.org/profiles/test-case">
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base" href="http://127.0.0.1/" />
<title>New Test</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">New Test</td></tr>
</thead><tbody>
<tr>
	<td>open</td>
	<td>/clinic/login/</td>
	<td></td>
</tr>
<tr>
	<td>clickAndWait</td>
	<td>css=#menu-item-52 &amp;gt; a &amp;gt; span</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>id=wpcrl_email</td>
	<td>nazar</td>
</tr>
<tr>
	<td>type</td>
	<td>id=wpcrl_password</td>
	<td>nazar</td>
</tr>
<tr>
	<td>type</td>
	<td>id=wpcrl_fname</td>
	<td>Asfasf</td>
</tr>
<tr>
	<td>type</td>
	<td>id=wpcrl_lname</td>
	<td>Afasf</td>
</tr>
<tr>
	<td>click</td>
	<td>id=wpcrl_username</td>
	<td></td>
</tr>
<tr>
	<td>type</td>
	<td>id=wpcrl_username</td>
	<td>afsafsa</td>
</tr>
<tr>
	<td>type</td>
	<td>id=wpcrl_email</td>
	<td>asfsafa</td>
</tr>
<tr>
	<td>type</td>
	<td>id=wpcrl_password</td>
	<td>asfasf</td>
</tr>
<tr>
	<td>click</td>
	<td>id=wpcrl_password2</td>
	<td></td>
</tr>
```

```
	<tr>
		<td>type</td>
		<td>id=wpcrl_password2</td>
		<td>agsgga</td>
	</tr>
	<tr>
		<td>type</td>
		<td>name=wpcrl_captcha</td>
		<td>sfgs</td>
	</tr>

	</tbody></table>
	</body>
	</html>

	<?xml version="1.0" encoding="UTF-8"?>
	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
	<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
	<head profile="http://selenium-ide.openqa.org/profiles/test-case">
	<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
	<link rel="selenium.base" href="http://127.0.0.1/" />
	<title>New Test</title>
	</head>
	<body>
	<table cellpadding="1" cellspacing="1" border="1">
	<thead>
	<tr><td rowspan="1" colspan="3">New Test</td></tr>
	</thead><tbody>
	<tr>
		<td>open</td>
		<td>/clinic/doctors/gregory_house/?clinic=87&amp;speciality=%D0%A2%D0%B5%D1%80
%D0%B0%D0%BF%D0%B5%D0%B2%D1%82&amp;room=409</td>
		<td></td>
	</tr>
	<tr>
		<td>clickAndWait</td>
		<td>link=авторизуватися!</td>
		<td></td>
	</tr>
	<tr>
		<td>type</td>
		<td>id=wpcrl_username</td>
		<td>nazar</td>
	</tr>
	<tr>
		<td>type</td>
		<td>id=wpcrl_password</td>
		<td>nazar</td>
	</tr>
	<tr>
		<td>click</td>
		<td>css=button.btn.btn-primary</td>
		<td></td>
	</tr>
	<tr>
		<td>clickAndWait</td>
		<td>css=#menu-item-263 &gt; a &gt; span</td>
		<td></td>
	</tr>

	</tbody></table>
	</body>
	</html>
```

ДОДАТОК Ж ТЕСТ ПЛАН

1. Вступ

Тестування займає важливий етапом розробки програмного забезпечення. Тестування дозволяє з’ясувати чи справді розроблений продукт відповідає поставленим вимогам, а також та дозволяє визначити помилки та дефекти, які є в продукті для їх подальшого виправлення.

2. Розробка тестів.

Для розроблюваного програмного продукту проведено наступні види тестування: функціональне та GUI тестування.

Для перелічених видів тестування розроблені та виконані тестові випадки. Результати тестів задокументовано.

2.1. Функціональне тестування;

Для реалізації функціонального тестування було розроблено 12 тестових випадків. Тестові випадки, щоб максимально якісно протестувати програмний продукт на відповідність функціональним вимогам зазначеним в специфікації. В результаті проведення функціонального тестування 12 із 12 тестових випадків пройшли успішно.

2.2. GUI тестування

В результаті проведення тестування користувацького інтерфейсу можна зробити висновок, що даний вид тестування є успішним так як всі розроблені тести пройшли успішно, що означає, що користувацький інтерфейс організовано правильно.

2.3. Тестування безпеки

В системі реалізовані механізми організації безпеки даних, а саме рівні доступу до інформації, перевірка та фільтрація отриманих даних ззовні.

3. Метрики.

Всі заплановані тестові випадки розроблено. Покриття тестами варіантів використання становить 100%.

4. Критерії прийняття відхилення релізу.

Всі розроблені тестові випадки виконано.

Функціональне тестування не виявило багів в роботі розробленого продукту.

Результати GUI тестування показали, що інтерфейс користувача розроблений правильно.

Тестування безпеки відповідає поставленим вимогам.

ДОДАТОК З КОД ПРОГРАМНОГО ПРОДУКТУ

template-clinics.php

```
<?php

get_header();
$clinics = new WP_Query( array( 'post_type' => 'clinic', 'posts_per_page' => -1 ) );
$clinic_categories = get_terms( 'category' );
?>

    <div id="Content">
        <div class="content_wrapper clearfix">
            <div class="sections_group">
                <div class="entry-content" itemprop="mainContentOfPage">
                    <div  class="container-fluid"  style="padding-right:  20px;  padding-
left: 30px;">
                        <div class="row">
                        <?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; ?>
                        </div>
                    </div>
                    <div class="section " style="padding-top:60px; padding-bottom:0px;
background-color:"><div   class="section_wrapper   clearfix"><div   class="items_group
clearfix"><div  class="column  one  column_column  "><div  class="column_attr    animate
bounceIn"        data-anim-type="bounceIn"><img        src="http://127.0.0.1/clinic/wp-
content/uploads/2014/11/doctor_heart_rate.png"                  class="scale-with-grid"
alt=""></div></div></div></div></div>
                </div>
            </div>
        </div>
```

```
    </div>

<?php get_footer(); ?>
```

single-clinic.php

```
<?php
/**
 * The Template for displaying all single posts.
 *
 * @package Betheme
 * @author Muffin group
 * @link http://muffingroup.com
 */

get_header();
?>

<!-- #Content -->
<div id="Content">
          <div class="content_wrapper clearfix">
            <!-- .sections_group -->
            <div class="sections_group">
                  <div  class="container-fluid"  style="padding-right:  20px;  padding-
left: 30px;">
                <div class="row">
                    <?php
                    $doctors = array();
                    // Find connected pages (for all posts)
                    p2p_type( 'doctors_to_clinics' )->each_connected( $wp_query );
                    ?>

                    <?php while ( $wp_query->have_posts() ) : $wp_query->the_post(); ?>
                        <?php
                            $clinic_name = $post->ID;
                        ?>
                        <h1 style="text-align: center;"><?php the_title(); ?> </h1>
                        <h4 style="text-align: center;">Оберіть лікаря</h4>

                        <?php
                            foreach  (  $post->connected  as  $post  )  :  setup_postdata(
$post );
                                $name = get_post_meta( $post->ID, 'doctor_name', true);
                                $room = get_post_meta( $post->ID, 'doctor_room', true);
                                $speciality     =     wp_get_post_terms(     $post->ID,
'speciality', array("fields" => "names"));
                                $slug  =  wp_get_post_terms(  $post->ID,  'speciality',
array("fields" => "slugs"));
                                $link = esc_url(get_permalink());
                                $doctors[] = array("slug" => $slug[0], "name" => $name,
"speciality" => $speciality[0], "room" => $room, "link" => $link);
                            endforeach;
                            $specialities = array();
                            foreach ($doctors as $key => $row)
                            {
                                $specialities[$key] = $row['speciality'];
                            }
                            array_multisort($specialities, SORT_ASC, $doctors);
                        ?>
                    <ul id="specialities-list">
```

```
                        <?php
                        $previousSpeciality;
                        foreach ($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"]."&r
oom=".$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;
                        }
                        ?>
                    </ul>

                    <?php
                        wp_reset_postdata(); // set $post back to original post
                        endwhile;
                    ?>
                </div>
            </div>
            <div   class="section   "   style="padding-top:60px;   padding-bottom:0px;
background-color:"><div   class="section_wrapper   clearfix"><div   class="items_group
clearfix"><div  class="column  one  column_column  "><div  class="column_attr   animate
bounceIn"        data-anim-type="bounceIn"><img        src="http://127.0.0.1/clinic/wp-
content/uploads/2014/11/doctor_heart_rate.png"                  class="scale-with-grid"
alt=""></div></div></div></div></div>
            </div>

            <!-- .four-columns - sidebar -->
            <?php get_sidebar(); ?>

          </div>
</div>

<?php get_footer(); ?>
```

single-doctor.php

```
<?php
/**
 * The Template for displaying all single doctor posts.
 *
 * @package Betheme
 * @author Muffin group
 * @link http://muffingroup.com
 */

get_header();
?>
<link                 rel="stylesheet"                href="http://127.0.0.1/clinic/wp-
content/themes/betheme/lib/themes/default.css">
```

```
<link                rel="stylesheet"                href="http://127.0.0.1/clinic/wp-
content/themes/betheme/lib/themes/default.date.css">
<link                rel="stylesheet"                href="http://127.0.0.1/clinic/wp-
content/themes/betheme/lib/themes/default.time.css">
<script src="http://127.0.0.1/clinic/wp-content/themes/betheme/lib/picker.js"></script>
<script                                                src="http://127.0.0.1/clinic/wp-
content/themes/betheme/lib/picker.date.js"></script>
<script                                                src="http://127.0.0.1/clinic/wp-
content/themes/betheme/lib/picker.time.js"></script>
<script src="http://127.0.0.1/clinic/wp-content/themes/betheme/lib/legacy.js"></script>

<!-- #Content -->
<div id="Content">
          <div class="content_wrapper clearfix">
             <!-- .sections_group -->
             <div class="sections_group">
                   <div  class="container-fluid"  style="padding-right:  20px;  padding-
left: 30px;">
                    <?php while ( $wp_query->have_posts() ) : $wp_query->the_post(); ?>
                    <?php
                        echo                                                    "<span
class='breadcrumbs'>".get_the_title($_GET["clinic"])."</span>";
                        echo         "<input         id='doctorID'         type='hidden'
value='".get_the_id()."'>";
                    ?>
                    <i class="icon-right-bold"></i>
                    <span class='breadcrumbs'><?php echo $_GET["speciality"] ?></span>
                    <i class="icon-right-bold"></i>
                    <span class='breadcrumbs'><?php the_title(); ?></span><br/>
                        <section class="section">
                            <div  class="container  schedule_table"  style="text-align:
center;">
                                <h3   style='text-align:   center;   margin-top:   20px;
margin-bottom: 25px;'>Графік роботи</h3>
                                <i   class="fa   fa-spinner   fa-pulse   fa-3x   fa-fw"
style="margin: 10px 0;"></i>
                            </div>

                            <?php
                            if (!is_user_logged_in()) {
                                echo  "<div  id='not_authorized'><h3  style='text-align:
center;  margin-top:  20px;'>Для  того,  щоб  записатися  на  прийом  Ви  повинні  <a
style='text-decoration:          underline;'          href="          .get_site_url().
"/login/>авторизуватися!</a></h3></div>";
                            }
                            else { ?>

                            <div class="container pickers" style="display: none;">
                                <form       method="post"       action="<?php       echo
get_site_url().'/personal-data/' ?>" style="text-align: center;">
                                    <input  type="hidden"  name="clinicID"  value="<?php
echo $_GET['clinic']; ?>">
                                    <input  type="hidden"  name="doctorID"  value="<?php
echo get_the_id(); ?>">
                                    <input type="hidden" name="speciality" value="<?php
echo $_GET['speciality']; ?>">
                                    <input  type="hidden"  name="room"  value="<?php  echo
$_GET['room']; ?>">
                                    <h3    style="text-align:    center;    margin-top:
20px;">Оберіть дату прийому</h3><br/>
                                    <fieldset id="datepicker_fieldset">
```

```
                                        <input       placeholder="Оберіть       дату..."
id="input_01"  class="datepicker"  name="chosenDate"  type="text"  autofocuss  value=""
data-valuee="">
                                    </fieldset>
                                    <div id="timeInputWrapper" style="display:none;">
                                        <h3   style="text-align:   center;   margin-top:
20px;">Оберіть час прийому</h3><br/>
                                        <fieldset id="timepicker_fieldset">
                                            <input     readonly     placeholder="Оберіть
час..."  id="input_02"  class="timepicker"  name="chosenTime"  type="time"  autofocuss
value="">
                                        </fieldset>
                                    </div>
                                    <button     name="submitDate"     id="submitDateBtn"
style="display: none;">Підтвердити!</button>
                                </form>
                            </div>

                            <?php } ?>

                        </section>
                            <div id="container-for-date"></div>
                            <div id="container-for-time"></div>
                    <?php
                        wp_reset_postdata(); // set $post back to original post
                        endwhile;
                    ?>
              </div>
              <div   class="section   "   style="padding-top:30px;   padding-bottom:0px;
background-color:"><div   class="section_wrapper   clearfix"><div   class="items_group
clearfix"><div  class="column  one  column_column  "><div  class="column_attr    animate
bounceIn"        data-anim-type="bounceIn"><img        src="http://127.0.0.1/clinic/wp-
content/uploads/2014/11/doctor_heart_rate.png"                  class="scale-with-grid"
alt=""></div></div></div></div></div>
              </div>

              <!-- .four-columns - sidebar -->
              <?php get_sidebar(); ?>

            </div>
</div>
<?php get_footer(); ?>
```

personalData.php

```
<?php
/**
 * Template Name: Personal data form
 *
 * @author Shershen Nazar
 */

get_header();
?>

<div id="Content">
            <div class="content_wrapper clearfix">
              <!-- .sections_group -->
              <div class="sections_group">
                    <div class="container">
                 <?php
```

```
                    echo                                                    "<span
class='breadcrumbs'>".get_the_title($_POST["clinicID"])."</span>";
                ?>
                <i class="icon-right-bold"></i>
                <span class='breadcrumbs'><?php echo $_POST["speciality"] ?></span>
                <i class="icon-right-bold"></i>
                <span class='breadcrumbs'><?php echo get_the_title($_POST['doctorID']);
?></span>
                <i class="icon-right-bold"></i>
                <span class='breadcrumbs'><?php echo $_POST['chosenDate']; ?></span>
                <i class="icon-right-bold"></i>
                <span     class='breadcrumbs'><?php     echo     $_POST['chosenTime'];
?></span><br/><br/><br/>
                    <section class="section" id="ticketFormSection">
                        <form  id="ticketForm"  method="post"  action="<?php  echo
get_site_url().'/ticket/' ?>">
                            <?php    wp_nonce_field(    basename(    __FILE__    ),
'ticket_nonce' ); ?>
                            <input type="hidden" name="clinicID" value="<?php echo
$_POST["clinicID"]; ?>">
                            <input type="hidden" name="doctorID" value="<?php echo
$_POST['doctorID']; ?>">
                            <input  type="hidden"  name="speciality"  value="<?php
echo $_POST['speciality']; ?>">
                            <input  type="hidden"  name="chosenDate"  value="<?php
echo $_POST['chosenDate']; ?>">
                            <input  type="hidden"  name="chosenTime"  value="<?php
echo $_POST['chosenTime']; ?>">
                            <input  type="hidden"  name="room"  value="<?php  echo
$_POST['room']; ?>">
                            <div class="data-row float">
                                <label   for="patient_name">Прізвище,   Ім'я,   По-
батькові</label>
                                <input   required   type="text"   name="patientName"
id="patient_name">
                            </div>
                            <div class="data-row">
                                <label                  for="patient_birthDay">Дата
народження</label>
                                <input  required  type="date"  name="patientBirthday"
id="patient_birthDay">
                            </div>
                            <div class="data-row float">
                                <label for="patient_city">Населений пункт</label>
                                <input   required   type="text"   name="patientCity"
id="patient_city">
                            </div>
                            <div class="data-row float">
                                <label for="patient_street">Вулиця</label>
                                <input  required  type="text"  name="patientStreet"
id="patient_street">
                            </div>
                            <div class="data-row float">
                                <label for="patient_building">Будинок</label>
                                <input  required  type="text"  name="patientBuilding"
id="patient_building">
                            </div>
                            <div class="data-row" >
                                <label for="patient_appartment">Квартира</label>
                                <input     type="text"     name="patientAppartment"
id="patient_appartment">
```

```
                                </div>
                                <div class="submit_personal_data_wrapper">
                                    <button                    name="submitPersonalData"
id="submit_personal_data">Підтвердити!</button>
                                </div>
                            </form>
                    </section>
            </div>
        </div>
        <div class="section " style="padding-top:30px; padding-bottom:0px; background-
color:"><div  class="section_wrapper  clearfix"><div  class="items_group  clearfix"><div
class="column one column_column "><div class="column_attr  animate bounceIn" data-anim-
type="bounceIn"><img                                    src="http://127.0.0.1/clinic/wp-
content/uploads/2014/11/doctor_heart_rate.png"                  class="scale-with-grid"
alt=""></div></div></div></div></div>
    </div>
</div>

<?php get_footer(); ?>
```

ticketGenerate.php

```
<?php
/**
 * Template Name: Ticket Generate
 *
 * @author Shershen Nazar
 */

get_header();
?>

<?php

$is_valid_nonce             =             (isset($_POST['ticket_nonce'])             &&
wp_verify_nonce($_POST['ticket_nonce'], basename(__FILE__))) ? 'true' : 'false';

$clinicName = get_the_title($_POST['clinicID']);
$doctorName = get_the_title($_POST['doctorID']);
$chosenDate = date_create($_POST['chosenDate']);
$chosenTime = $_POST['chosenTime'];
$patientBirthday = date_create($_POST['patientBirthday']);



function     saveTicket($clinicName,     $doctorName,     $chosenDate,     $chosenTime,
$patientBirthday) {

    $ticketHtml = "<div id='ticket-wrapper'>";
    $ticketHtml          .=          "<h3          style='text-align:          center;'>"
.get_the_title($_POST['clinicID']). "</h3>";
    $ticketHtml .= "<hr><ul class='ticket-details'>";
    $ticketHtml .= "<li>
                        <span class='detail-title'>Дата та час<br/>прийому</span>
                        <span     class='detail-content'>"     .date_format($chosenDate,
'd.m.Y'). " " .$chosenTime. "</span>
                    </li>";
    $ticketHtml .= "<li>
                        <span class='detail-title'>Спеціальність<br/>лікаря</span>
                        <span class='detail-content'>" .$_POST['speciality']. "</span>
                    </li>";
```

```
    $ticketHtml .= "<li>
                        <span class='detail-title'>Ім'я лікаря</span>
                        <span class='detail-content'>"  .$doctorName. "</span>
                    </li>";
    $ticketHtml .= "<li>
                        <span class='detail-title'>Кабінет</span>
                        <span class='detail-content'>" .$_POST['room']. "</span>
                    </li>";
    $ticketHtml .= "<li>
                        <span class='detail-title'>Ім'я пацієнта</span>
                        <span class='detail-content'>" .$_POST['patientName']. "</span>
                    </li>";
    $ticketHtml .= "<li>
                        <span class='detail-title'>Дата<br/>народження</span>
                        <span  class='detail-content'>"  .date_format($patientBirthday,
'd.m.Y'). "</span>
                    </li>";
    $ticketHtml .= "<li>
                        <span class='detail-title'>Адреса<br/>проживання</span>
                        <span class='detail-content'>" .$_POST['patientCity']. " "
.$_POST['patientStreet'].      " "      .$_POST['patientBuilding'].      " "
.$_POST['patientAppartment']. "</span>
                    </li>";
    $ticketHtml .= "<li>
                <i                                    class='icon-location'></i><span>"
.esc_html(get_post_meta($_POST['clinicID'], 'clinic_address', true)). "</span><br>
                <i                 class='icon-phone'></i><span>Тел:                "
.esc_html(get_post_meta($_POST['clinicID'],    'clinic_phone1',    true)).    "; "
.esc_html(get_post_meta($_POST['clinicID'], 'clinic_phone2', true)). "</span>
            </li></ul></div>";

    echo $ticketHtml;

    $post_data = array(
        'post_title'    => wp_strip_all_tags($_POST['patientName']),
        'post_status'   => 'publish',
        'post_author'   => get_current_user_id(),
        'post_type'     => 'ticket'
    );

    $post_id = wp_insert_post($post_data, true);

    if(is_int($post_id)) {

        if ( isset($clinicName) ) {
            update_post_meta( $post_id, 'clinic_name', sanitize_text_field($clinicName)
);
        }

        if ( isset( $_POST[ 'clinicID' ] ) ) {
            update_post_meta(   $post_id,   'clinic_id',   sanitize_text_field(   $_POST[
'clinicID' ] ) );
        }

        if ( isset( $_POST[ 'room' ] ) ) {
            update_post_meta(  $post_id,  'doctor_room',  sanitize_text_field(  $_POST[
'room' ] ) );
        }

        if ( isset ($doctorName) ) {
```

```
            update_post_meta( $post_id, 'doctor_name', sanitize_text_field($doctorName)
);
        }

        if ( isset( $_POST[ 'doctorID' ] ) ) {
            update_post_meta(   $post_id,   'doctor_id',   sanitize_text_field(   $_POST[
'doctorID' ] ) );
        }

        if ( isset( $_POST[ 'patientName' ] ) ) {
            update_post_meta(  $post_id,  'patient_name',  sanitize_text_field(  $_POST[
'patientName' ] ) );
        }

        if ( isset( $_POST[ 'patientCity' ] ) ) {
            update_post_meta(  $post_id,  'patient_city',  sanitize_text_field(  $_POST[
'patientCity' ] ) );
        }

        if ( isset( $_POST[ 'patientBuilding' ] ) ) {
            update_post_meta(    $post_id,    'patient_building',    sanitize_text_field(
$_POST[ 'patientBuilding' ] ) );
        }

        if ( isset( $_POST[ 'patientStreet' ] ) ) {
            update_post_meta( $post_id, 'patient_street', sanitize_text_field( $_POST[
'patientStreet' ] ) );
        }

        if ( isset( $_POST[ 'patientAppartment' ] ) ) {
            update_post_meta(    $post_id,    'patient_apartment',    sanitize_text_field(
$_POST[ 'patientAppartment' ] ) );
        }

        if ( isset($patientBirthday ) ) {
            update_post_meta(               $post_id,               'patient_birthdate',
sanitize_text_field(date_format($patientBirthday,  "Y-m-d")));
        }

        if ( isset($chosenDate)) {
            update_post_meta(                $post_id,                'visiting_date',
sanitize_text_field(date_format($chosenDate,  "Y-m-d")));
        }

        if ( isset($chosenTime) ) {
            update_post_meta(                $post_id,                'visiting_time',
sanitize_text_field($chosenTime));
        }
    }

    $current_user = wp_get_current_user();
    $email = $current_user->user_email;
    $to = $email;
    $subject = "Талон на прийом до лікаря";
    $headers = "MIME-Version: 1.0" . "\r\n";
    $headers .= "Content-type:text/html;charset=UTF-8" . "\r\n";
    mail($email, $subject, $ticketHtml, $headers);
}

?>
```

```
 <script>
function print(ele) {
    var openWindow = window.open("", "Талон", "width=1000, height=600");
    openWindow.document.write("<link rel='stylesheet' href='http://127.0.0.1/clinic/wp-
content/themes/betheme/css/custom.css?ver=7.0' type='text/css'>");
    openWindow.document.write("<div id='ticket-wrapper' class='without-shadow'>");
    openWindow.document.write(ele.previousSibling.innerHTML);
    openWindow.document.write("</div>");
    openWindow.document.close();
    openWindow.focus();
    setTimeout(function(){ openWindow.print(); openWindow.close(); }, 1);
}
</script>

<div id="Content">
          <div class="content_wrapper clearfix">
             <div class="sections_group">
                   <div class="container">
                <section class="section" id="ticketGenerated">
                    <?php if(!$is_valid_nonce){

                        echo  "<br/><h1  style='text-align:  center;'>Перевірочні  дані
невірні!</h1>";
                    }
                    else {

                        echo "<br/><h1>Ваш талон</h1><br/>";
                        saveTicket($clinicName,  $doctorName,  $chosenDate,  $chosenTime,
$patientBirthday);
                        echo        "<button        href='#'        class='printButton'
onclick='print(this)'>Друкувати</button>";
                    }
                    ?>
                </section>
            </div>
        </div>
        <div class="section " style="padding-top:30px; padding-bottom:0px; background-
color:"><div  class="section_wrapper  clearfix"><div  class="items_group  clearfix"><div
class="column one column_column "><div class="column_attr  animate bounceIn" data-anim-
type="bounceIn"><img                                    src="http://127.0.0.1/clinic/wp-
content/uploads/2014/11/doctor_heart_rate.png"                  class="scale-with-grid"
alt=""></div></div></div></div></div>
    </div>
</div>
<?php get_footer(); ?>
```

wp-clinic-cpt.php

```
<?php

function register_post_type_clinic() {
    $singular = __('Clinic');
    $plural = __('Clinics');

    //Used for the rewrite slug below.
    $plural_slug = str_replace( ' ', '_', $plural );

    //Setup all the labels to accurately reflect this post type.
    $labels = array(
             'name'                                  => $plural,
```

```
            'singular_name'          => $singular,
            'add_new'                       => 'Add New',
            'add_new_item'                  => 'Add New ' . $singular,
            'edit'                   => 'Edit',
            'edit_item'              => 'Edit ' . $singular,
            'new_item'               => 'New ' . $singular,
            'view'                              => 'View ' . $singular,
            'view_item'              => 'View ' . $singular,
            'search_term'            => 'Search ' . $plural,
            'parent'                        => 'Parent ' . $singular,
            'not_found'              => 'No ' . $plural .' found',
            'not_found_in_trash'     => 'No ' . $plural .' in Trash'
        );

    //Define all the arguments for this post type.
          $args = array(
            'labels'                    => $labels,
            'public'              => true,
        'publicly_queryable'  => true,
        'exclude_from_search' => false,
        'show_in_nav_menus'   => true,
        'show_ui'             => true,
        'show_in_menu'        => true,
        'show_in_admin_bar'   => true,
        'menu_position'       => 6,
        'menu_icon'           => 'dashicons-building',
        'can_export'          => true,
        'delete_with_user'    => false,
        'hierarchical'        => false,
        'has_archive'         => true,
        'query_var'           => true,
        'capability_type'     => 'page',
        'map_meta_cap'        => true,
        // 'capabilities' => array(),
        'rewrite'             => array(
          'slug' => strtolower( $plural_slug ),
          'with_front' => true,
          'pages' => true,
          'feeds' => false,

        ),
        'supports'            => array(
          'title'
        )
          );

    //Create the post type using the above two varaiables.
          register_post_type( 'clinic', $args);
}
add_action( 'init', 'register_post_type_clinic' );

function clinic_register_taxonomy() {

    $plural = __( 'Categories' );
          $singular = __( 'Category' );

    $labels = array(
             'name'                       => $plural,
        'singular_name'             => $singular,
        'search_items'              => 'Search ' . $plural,
        'popular_items'             => 'Popular ' . $plural,
```

```
        'all_items'                  => 'All ' . $plural,
        'parent_item'                => null,
        'parent_item_colon'          => null,
        'edit_item'                  => 'Edit ' . $singular,
        'update_item'                => 'Update ' . $singular,
        'add_new_item'               => 'Add New ' . $singular,
        'new_item_name'              => 'New ' . $singular . ' Name',
        'separate_items_with_commas' => 'Separate ' . $plural . ' with commas',
        'add_or_remove_items'        => 'Add or remove ' . $plural,
        'choose_from_most_used'      => 'Choose from the most used ' . $plural,
        'not_found'                  => 'No ' . $plural . ' found.',
        'menu_name'                  => $plural,
          );

    $args = array(
             'hierarchical'          => false,
        'labels'                => $labels,
        'show_ui'               => true,
        'show_admin_column'     => true,
        'update_count_callback' => '_update_post_term_count',
        'query_var'             => true,
        'rewrite'               => array( 'slug' => strtolower( $singular ) ),
          );

    register_taxonomy( 'category', 'clinic', $args );
}
add_action( 'init', 'clinic_register_taxonomy' );
```

wp-clinic-fields.php

```
<?php

function clinic_add_custom_metabox() {

          add_meta_box(
             'clinic_meta',
             __( 'Clinics' ),
             'clinic_meta_callback',
             'clinic',
             'normal',
             'core'
          );
}

add_action( 'add_meta_boxes', 'clinic_add_custom_metabox' );

function clinic_meta_callback( $post ) {
          wp_nonce_field( basename( __FILE__ ), 'clinics_nonce' );
          $clinic_stored_meta = get_post_meta( $post->ID ); ?>

    <div>
        <div class="meta-row">
            <div class="meta-th">
                <label   for="clinic-name"   class="clinic-row-title"><?php   _e('Clinic
Name', 'polyclinic'); ?></label>
            </div>
            <div class="meta-td">
                          <input         type="text"         class="clinic-row-content"
name="clinic_name" id="clinic-name"
                          value="<?php        if        (        !        empty        (
$clinic_stored_meta['clinic_name'] ) ) {
```

```
                                echo esc_attr( $clinic_stored_meta['clinic_name'][0] );
                            } ?>"/>
                    </div>
            </div>
            <div class="meta-row">
                <div class="meta-th">
                    <label  for="clinic-address"  class="clinic-row-title"><?php  _e('Clinic
Address', 'polyclinic'); ?></label>
                </div>
                <div class="meta-td">
                            <input         type="text"         class="clinic-row-content"
name="clinic_address" id="clinic-address"
                            value="<?php        if        (        !        empty        (
$clinic_stored_meta['clinic_address'] ) ) {
                                echo esc_attr( $clinic_stored_meta['clinic_address'][0]
);
                            } ?>"/>
                    </div>
            </div>
            <div class="meta-row">
                <div class="meta-th">
                    <label  for="clinic-phone1"  class="clinic-row-title"><?php  _e('Clinic
Phone 1', 'polyclinic'); ?></label>
                </div>
                <div class="meta-td">
                            <input         type="text"         class="clinic-row-content"
name="clinic_phone1" id="clinic-phone1"
                            value="<?php        if        (        !        empty        (
$clinic_stored_meta['clinic_phone1'] ) ) {
                                echo  esc_attr(  $clinic_stored_meta['clinic_phone1'][0]
);
                            } ?>"/>
                    </div>
            </div>
            <div class="meta-row">
                <div class="meta-th">
                    <label  for="clinic-phone2"  class="clinic-row-title"><?php  _e('Clinic
Phone 2', 'polyclinic'); ?></label>
                </div>
                <div class="meta-td">
                            <input         type="text"         class="clinic-row-content"
name="clinic_phone2" id="clinic-phone2"
                            value="<?php        if        (        !        empty        (
$clinic_stored_meta['clinic_phone2'] ) ) {
                                echo  esc_attr(  $clinic_stored_meta['clinic_phone2'][0]
);
                            } ?>"/>
                    </div>
            </div>
            <div class="meta-row">
                <div class="meta-th">
                    <label   for="clinic-email"   class="clinic-row-title"><?php  _e('Clinic
Email', 'polyclinic'); ?></label>
                </div>
                <div class="meta-td">
                            <input         type="text"         class="clinic-row-content"
name="clinic_email" id="clinic-email"
                            value="<?php        if        (        !        empty        (
$clinic_stored_meta['clinic_email'] ) ) {
                                echo  esc_attr(  $clinic_stored_meta['clinic_email'][0]
);
```

```
                        } ?>"/>
                    </div>
            </div>
        </div>

<?php
}

function clinic_meta_save ($post_id) {
    // Checks save status
    $is_autosave = wp_is_post_autosave($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';

    // Exits script depending on save status
    if ( $is_autosave || $is_revision || !$is_valid_nonce ) {
        return;
    }

    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' ] ) );
    }

    if ( isset( $_POST[ 'clinic_phone2' ] ) ) {
          update_post_meta(  $post_id,  'clinic_phone2',  sanitize_text_field(  $_POST[
'clinic_phone2' ] ) );
    }

}
add_action('save_post', 'clinic_meta_save');
```