

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА
ФАКУЛЬТЕТ УПРАВЛІННЯ ФІНАНСАМИ ТА БІЗНЕСУ

**Кафедра цифрової економіки та бізнес-
аналітики**

КУРСОВА РОБОТА
з навчальної дисципліни
“Проектування та адміністрування БД і СД”

Тема:

«Інформаційна система приватної поліклініки»

Науковий керівник:

к.ф.-м.н., доц. Депутат Б.Я.

(прізвище, ім'я, по-батькові)

_____ (підпис)

Виконавець:

Щадило Н. Я.

(прізвище, ім'я, по-батькові)

УФЕ-31с група

_____ (підпис)

“ ____ ” _____ 2020 р.

“ ____ ” _____ 2020 р.

Загальна кількість балів _____

(підписи, ППІ членів комісії)

Львів 2020

ЗМІСТ

ВСТУП	3
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ	5
1.1 Коротка характеристика об’єкту управління інформаційної системи приватної поліклініки	5
1.2 Опис предметної області.....	7
РОЗДІЛ 2. РОЗРОБКА БАЗИ ДАНИХ	11
2.1 Основні відомості про реляційні бази даних	11
2.2 Система управління базами даних MySQL.....	13
2.3 Розробка архітектури програмної системи	15
2.4 Проектування структури бази даних.....	18
2.4.1 Перелік таблиць бази даних.....	19
2.4.2 Перелік полів таблиць бази даних.....	20
2.5 Функціональні залежності між атрибутами.....	21
2.6 Даталогічне проектування бази даних	23
2.6.1 Склад таблиць бази даних.....	24
2.7 Запити до таблиць бази даних	26
РОЗДІЛ 3. ІНСТРУКЦІЯ З ВИКОРИСТАННЯ WEB-САЙТУ	31
3.1 Програмна реалізація проекту	31
3.2 Структура веб-сайту.....	32
3.3 Програмування клієнтської частини	33
3.4 Програмування серверної частини.....	34
3.5 Макети сторінок веб-сайту	35
ВИСНОВКИ	38
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	39
ДОДАТКИ	41

ВСТУП

Кожен з нас піклується про своє здоров'я. Хтось почуває себе добре, і не має потреби відвідувати лікаря протягом тривалого часу, а хтось частіше почуває себе гірше ніж зазвичай. Саме тоді виникає потреба в отриманні медичної допомоги. Процес запису пацієнта на прийом до лікаря завжди буде актуальною задачею, тому що в разі виникнення проблеми зі здоров'ям, кожен з нас звертається за допомогою в сферу медицини і не кожен готовий чекати своєї черги довгий час. 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 для того, щоб пацієнт записався на прийом, йому потрібно іти на реєстратуру медичного закладу, і саме там він може здійснити запис на прийом, а це завжди займає багато часу. Тому розробка веб-сайту, який надасть можливість користувачу записатись на прийом до потрібного йому лікаря в медичному закладі є досить актуальним завданням. Такий програмний продукт, який буде містити в собі різноманітні функції, набагато пришвидшить та спростить процес запису на прийом до лікаря. Метою цієї курсової роботи є: розробка веб-сайту, на якому можна подати заявку на прийом до лікаря в медичний заклад та дослідити цю інформаційну систему приватної поліклініки.

Для досягнення мети курсової роботи, необхідно вирішити такі завдання:

1. коротко охарактеризувати інформаційну систему для приватної поліклініки;
2. проаналізувати предметну область;
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-сайт «Приватна поліклініка» для запису на прийом в медичний заклад;

Об'єкт дослідження – особливість розробки інформаційної системи для приватної поліклініки за допомогою системи управління базами даних MySQL.

Предмет досліджень – застосування технологій створення веб-орієнтованого програмного забезпечення для розробки веб-сайту «Приватна поліклініка».

В процесі розробки використовуватимуться наступні мови програмування та технології:

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

HTML- мова тегів, якою пишуться гіпертекстові документи для мережі Інтернет.

CSS- спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду. Самі ж сторінки написані мовами розмітки даних.

MySQL- вільна система керування реляційними базами даних.

Курсова робота складається зі вступу, 3 розділів, 17 підрозділів, висновків та додатків.

Загальний обсяг роботи складає 40 сторінок, яка включає 13 таблиць, 14 рисунків, список використаних джерел з 20 позицій та додатки.

У Розділ 1 проаналізовано предметну область інформаційної системи.

У Розділ 2 розділі описано розробку навчальної бази даних.

У Розділ 3 продемонстровано підходи та реалізацію WEB-орієнтованої системи запису пацієнта на прийом до лікаря.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Коротка характеристика об'єкту управління інформаційної системи приватної поліклініки

У будь-якій поліклініці наявний відділ реєстратури. Там же починається «знайомство» з чергами: за талонами, за карточками, 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 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 :

- пояснювати місцезнаходження потрібних лікарів;
- повідомляти пацієнтам графік роботи того чи іншого спеціаліста;
- виписувати талон 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 . 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 некоректно записати особисті дані пацієнта	Пацієнт може записатися на прийом не виходячи з дому
Запис на прийом в реєстратурі поліклініки	Потрібно стояти в черзі для отримання талону	Працівник реєстратури Може надати інформацію, яка цікавить пацієнта
Запис на прийом у журналі само запису	Оскільки ведення журналу само запису відбувається в паперовому вигляді, то можливі дублювання та втрата даних	Пацієнт може записатися на прийом самостійно, без допомоги працівника реєстратури

На сьогоднішній день більшість медичних закладів мають власні веб-сайти, де пацієнти можуть знайти потрібну їм інформацію.

На рисунку 1.1 можемо побачити один з прикладів веб-сайту поліклініки.



Рис 1.1 Клініка гематології Doctor Smart

З більш детальною інформацією можна ознайомитись на сайті клініки(<https://smart-clinic.032.ua/>).

Крім того, деякі з них реалізують функцію реєстрації на прийом до лікаря. Даний веб-сайт приватної поліклініки забезпечить для широкого кола

користувачів можливість переглядати інформацію про 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:

- користувач має можливість ознайомитись з медичним закладом;
- відсутність черг в реєстратурі для того щоб отримати талон на прийом;
- відсутність черг біля кабінетів лікарів, оскільки в базі даних будуть

вказані дата та час прийому;

- електронний графік прийому лікарів;
- електронний запис на прийом відбувається набагато швидше ніж видача талона працівниками реєстратури.

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

1.2 Опис предметної області

Для представлення роботи веб-сайту “Приватна поліклініка” виділено наступні бізнес-процеси (рисунок 1.2):

- реєстрація користувача в системі;
- реєстрація на прийом до лікаря;
- управління даними медичного закладу.

За управління даними медичного закладу в системі відповідатиме адміністратор, а реєструватися в системі і на прийом до лікаря буде сам користувач(пацієнт).



Рис. 1.2. Діаграма бізнес-процесів розроблюваного програмного продукту

Розглянемо детальніше кожен бізнес-процес з представленого списку (рисунок 1.2). На рисунку 1.3 зображено діаграму функцій процесу реєстрації користувача(пацієнта) в системі.

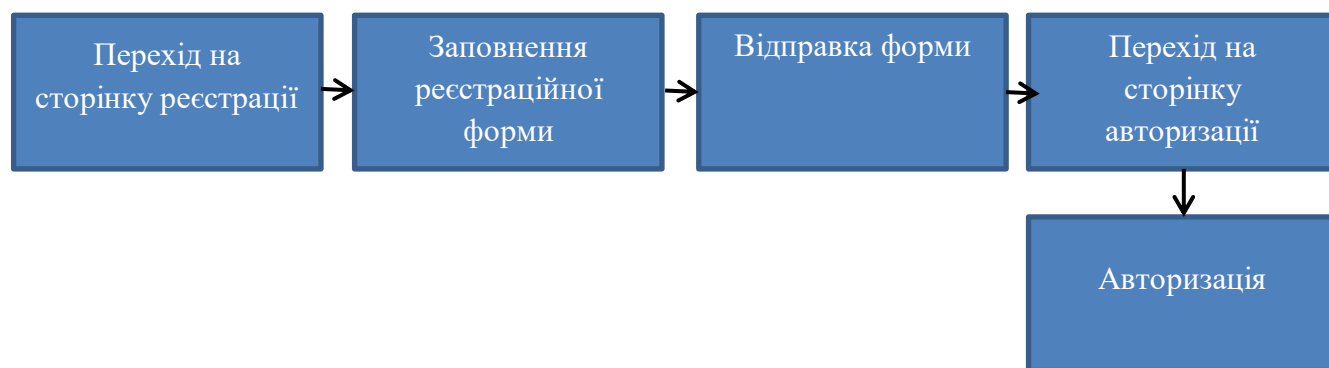


Рис. 1.3. Діаграма функцій процесу реєстрації користувача

Характеристику 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 процесу	Реєстрація користувача в системі

Продовження табл. 1.2

Основні учасники	Клієнт
Вхідна подія	Перехід на сторінку реєстрації
Вхідні документи	Дані з реєстраційної форми
Вихідна подія	Авторизація на сайті
Вихідні документи	Перехід на особистий профіль
Клієнт процесу	Користувач

Для того, щоб користувач(пацієнт) 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 та заповнити поля реєстраційної форми (електронна пошта, ім'я користувача, прізвище та пароль) та авторизуватися, використовуючи при цьому електронну пошту та пароль, вказані при реєстрації.

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, вводить особисті дані, перед ним випадає список лікарів, і він має змогу вибрати потрібного

йому лікаря. Після обрання лікаря, користувачу буде показано список графіку роботи лікарів. Після цього користувач обирає зручну йому день та годину прийому. Після обрання всіх даних, користувач подає заявку на прийом, де буде сформовано талон на прийом до лікаря.

Характеристику процесу реєстрації користувача в системі наведено в таблиці 1.3

Таблиця 1.3

Характеристика процесу реєстрації на прийом до лікаря

Назва характеристики	Значення характеристики
Ім'я 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 сформований і надісланий на пошту
Клієнт процесу	Користувач

Отже, підбиваючи підсумок за першим розділом, було коротко охарактеризовано об'єкт управління інформаційної системи для приватної поліклініки, подано характеристику процесу реєстрації користувача в системі та характеристику процесу реєстрації на прийом до лікаря.

РОЗДІЛ 2. РОЗРОБКА БАЗИ ДАНИХ

2.1 Основні відомості про реляційні бази даних

Будь-яка професійна діяльність так чи інакше пов'язана з інформацією, з організацією її збору, зберігання, вибірки. Можна сказати, що невід'ємною частиною повсякденного життя стали бази даних, для підтримки яких потрібний певний організаційний метод, або механізм. Такий механізм називають системою управління базами даних (СУБД).

База даних (БД) - спільно використовуваний набір логічно зв'язаних даних (і їх опис), призначений для задоволення інформаційних потреб організації.

СУБД (система управління базами даних) - програмне забезпечення, за допомогою якого користувачі можуть визначати, створювати і підтримувати базу даних, а також отримувати до неї контрольований доступ.

Системи управління базами даних існують вже багато років, багато хто з них зобов'язані своїм походженням системам з неструктурованими файлами на великих ЕОМ. Разом із загальноприйнятими сучасними технологіями в області систем управління базами даних починають з'являтися нові напрями, що обумовлено вимогами бізнесу, що росте, об'ємами корпоративних даних, що постійно збільшуються, і, звичайно ж, впливом технологій Internet.

Управління основними потоками інформації здійснюється за допомогою так званих систем управління реляційними базами даних, які беруть свій початок в традиційних системах управління базами даних. Реляційна база даних — база даних, заснована на реляційній моделі даних. Слово «реляційний» походить від англ. relation. Для роботи з реляційними БД застосовують реляційні СКБД. Інакше кажучи, реляційна база даних — це база даних, яка сприймається користувачем як набір нормалізованих відношень різного ступеня.

Реляційна база даних є сукупністю елементів даних, організованих у вигляді набору формально описаних таблиць, з яких дані можуть бути доступними або повторно зібрані багатьма різними способами без необхідності реорганізації таблиць бази даних.

Кожна таблиця БД представляється як сукупність рядків і стовпців, де рядки (записи) відповідають екземпляру об'єкту, конкретній події або явищу, а стовпці (поля) - атрибутам (ознакам, характеристикам, параметрам) об'єкту, події, явища.

У кожній таблиці БД необхідна наявність первинного ключа - так іменують поле або набір полів, що однозначно ідентифікує кожен екземпляр об'єкту або запис. Значення первинного ключа в таблиці БД повинне бути унікальним, тобто в таблиці не допускається наявність два і більш за записи з однаковими значеннями первинного ключа. Він повинен бути мінімально достатнім, а значить, не містити полів, видалення яких не відіб'ється на його унікальності. Зв'язки між об'єктами реального миру можуть знаходити своє віддзеркалення в структурі даних, а можуть і матися на увазі, тобто бути присутнім на неформальному рівні.

Між двома або більш таблицями бази даних можуть існувати відносини підлеглості, які визначають, що для кожного запису головної таблиці (батьківській) можлива наявність одного або декількох записів в підлеглий таблиці (дочірній).

Виділяють три різновиди зв'язку між таблицями бази даних:

- «один-до-багатьох»;
- «один-до-одного»;
- «багато-до-багатьох».

Зв'язок «один-до-багатьох»

Зв'язок «один-до-багатьох» має місце, коли одному запису батьківської таблиці може відповідати декілька записів дочірньої. Зв'язок «один-до-багатьох» іноді називають зв'язком «багато-до-одного». І у тому, і в іншому випадку суть зв'язку між таблицями залишається незмінною. Зв'язок «один-до-багатьох» є найпоширенішим для реляційних баз даних. Вона дозволяє моделювати також ієрархічні структури даних.

Зв'язок «один-до-одного»

Зв'язок «один-до-одного» має місце, коли одному запису в батьківській таблиці відповідає один запис в дочірній. Це відношення зустрічається набагато рідше, ніж відношення «один-до-багатьох». Його використовують, якщо не хочуть, щоб таблиця БД «розпухала» від другорядної інформації, проте для читання зв'язаної інформації в декількох таблицях доводиться проводити ряд операцій читання замість однієї, коли дані зберігаються в одній таблиці.

Зв'язок «багато-до-багатьох» застосовується в наступних випадках:

- одному запису в батьківській таблиці відповідає більш за один запис в дочірній;
- одному запису в дочірній таблиці відповідає більш за один запис в батьківській.

Будь-який зв'язок «багато-до-багатьох» в реляційній базі даних необхідно замінити на зв'язок «один-до-багатьох» (одну або більш) за допомогою введення додаткових таблиць.

2.2 Система управління базами даних MySQL

Структура програмного забезпечення MySQL є багаторівневою з незалежними модулями[19].

Дана система управління базами даних (СУБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL - одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL - компактний багатопоточний сервер баз даних. Характеризується великою швидкістю, стійкістю і простотою використання.

MySQL був розроблений компанією «ТсХ» для підвищення швидкодії обробки великих баз даних. MySQL вважається гарним рішенням для малих і середніх додатків. Вихідні коди сервера компілюються на безлічі платформ. Найбільш повно можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому.

Для некомерційного використання MySQL є безкоштовним. Можливості сервера MySQL:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн.;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

Недоліки сервера MySQL:

- не реалізована підтримка транзакцій. Натомість пропонується використовувати LOCK/UNLOCK TABLE;
- відсутня підтримка зовнішніх (foreign) ключів;
- відсутня підтримка тригерів і збережених процедур;

- відсутня підтримка представлень (VIEW).

Зазначені недоліки не є критичними при розробці малих і середніх ІС-інформаційних систем для робочих груп.

2.3 Розробка архітектури програмної системи

Для розробки архітектури веб-сайту «Приватна 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 запитом 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.

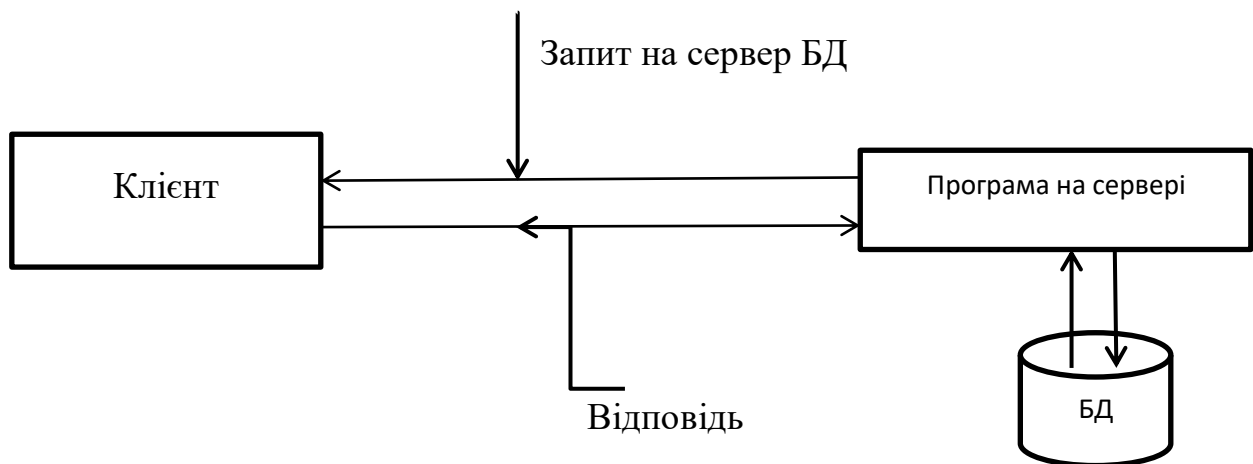


Рис. 2.1. Схема клієнт-серверної архітектури

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.

Діаграма станів показує те, що система може бути описана як скінченне число станів. 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 2.2



Рис. 2.2. Діаграма станів процесу реєстрації в системі

В процесі реєстрації система отримує реєстраційні дані з форми на сторінці сайту, перевіряє чи не існує користувача з вказаною електронною адресою.

Після успішного проходження перевірки дані користувача зберігаються в систему про успішну реєстрацію на сайті. На рисунку 2.3 зображено діаграму станів, що описує процес реєстрації пацієнта на прийом



Рис. 2.3. 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 лікарів та їх графік роботи в закладі. Користувач обирає лікаря і обирає графік роботи лікаря і надсилає запит на прийом до лікаря. Запит оформляється адміністратором сайту, увійшовши в систему за допомогою авторизації в системі.

2.4 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[21]:

- 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 концептуальної моделі даних. На основі визначених елементів і зв'язків створити ER – діаграму.

Діаграма концептуальної моделі даних представлена на рисунку 2.4

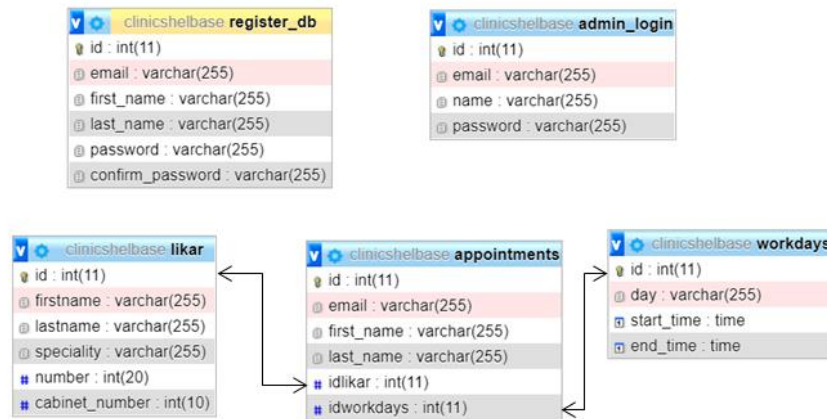


Рис. 2.4. Концептуальна модель даних

На основі концептуальної моделі даних будується реляційна модель даних, в якій кожному об'єкту відповідає таблиця, що містить всі його атрибути. Унікальний атрибут буде вважатися первинним ключем. Для утворення зв'язків між таблицями використовуються поля, котрі є зовнішніми ключами. Як результат побудови зв'язків між первинними та зовнішніми ключами, отримуємо реляційну модель даних.

2.4.1 Перелік таблиць бази даних

Таблицями бази даних є:

- а) *likar* - таблиця, яка містить інформацію про лікарів;
- б) *workdays* – таблиця, яка містить інформацію про графік роботи лікарів;
- в) *register_db* – таблиця, яка містить інформацію про користувача, який реєструється в системі;

2.4.2 Перелік полів таблиць бази даних

Поля, які ідентифікують властивості таблиці «*likar*»:

- а) *id* - ідентифікаційний код лікаря;
- б) *firstname* – ім'я лікаря;
- в) *lastname* – прізвище лікаря;
- г) *speciality* – спеціальність, яка закріплена за лікарем;
- д) *number* – номер телефону лікаря;
- е) *cabinet_number* – номер кабінету лікаря;

Поля, які ідентифікують властивості таблиці «*workdays*»:

- а) *id* - ідентифікаційний код графіку;
- б) *day* - день тижня за графіком;
- в) *start_time* - початок прийому;
- г) *end_time* – закінчення прийому;

Поля, які ідентифікують властивості таблиці «*register_db*»:

- а) *id* - ідентифікаційний код користувача;
- б) *email* - особистий email користувача;
- в) *first_name* – ім'я користувача;
- г) *last_name* – прізвище користувача;
- д) *password* – пароль введений користувачем;
- е) *confirm_password* – повторний пароль введений користувачем;

Поля, які ідентифікують властивості таблиці «*admin_login*»:

а) id - ідентифікаційний код адміністратора;

б)email – email адміністратора;

в)name – ім'я адміністратора;

д)password – пароль введений адміністратором;

Поля, які ідентифікують властивості таблиці «appointments»:

а) id - ідентифікаційний код користувача;

б)email - особистий email користувача;

в)first_name – ім'я користувача;

г)last_name – прізвище користувача;

д)idlikar – id лікаря, до якого записався користувач;

е)idworkdays – id прийому за графіком;

2.5 Функціональні залежності між атрибутами

Функціональна залежність - це зв'язок між атрибутами[20]. Припустімо, якщо нам відоме значення одного атрибута, тоді можемо знайти значення іншого атрибута. Функціональна залежність пов'язує атрибути в одному відношенні з єдиним значенням в іншому.

Функціональні залежності між атрибутами в таблиці «likar», зображено, що ключовим полем є поле ідентифікаційний код категорії, тобто поле «id», із значенням «Primary». Поля «firstname», «lastname», «speciality», «number», «cabinet_number» підпорядковуються полю «id».

Таблиця 2.1

Функціональні залежності між атрибутами сутності «лікар»

Найменування атрибутів	Функціональні залежності
id	_____
firstname	←_____
lastname	←_____
speciality	←_____
number	←_____
cabinet_number	←_____

Далі ми розглянемо функціональні залежності між атрибутами в таблиці «workdays», у якій міститься інформація про графік роботи лікарів. Графічно вона зображена у Таблиці 2.4. У ній відображено, що всі поля підпорядковуються ключовому полю «id»

Таблиця 2.2

Функціональні залежності між атрибутами сутності «workdays»

Найменування атрибутів	Функціональні залежності
id	_____
day	←_____
start_time	←_____
end_time	←_____

Функціональна залежність між атрибутами в таблиці «register_db», у якій міститься інформація про користувача сайту. У ній зображується, що ключовим полем є ідентифікаційний код користувача, тобто поле «id», із значенням «Primary». Усі інші поля підпорядковуються полю «id».

Таблиця 2.3

Функціональні залежності між атрибутами сутності «register_db»

Найменування атрибутів	Функціональні залежності
id	_____
email	←_____
first_name	←_____
last_name	←_____
password	←_____
confirm_password	←_____

Функціональна залежність між атрибутами в таблиці «admin_login», у якій міститься інформація про адміністратора сайту. У ній зображується, що ключовим полем є ідентифікаційний код адміністратора, тобто поле «id», із значенням «Primary». Усі інші поля підпорядковуються полю «id».

Таблиця 2.4

Функціональні залежності між атрибутами сутності «admin_login»

Найменування атрибутів	Функціональні залежності
id email name password	<div> <div></div> <div>←</div> <div>←</div> <div>←</div> </div>

Функціональна залежність між атрибутами в таблиці «appointments», у якій міститься інформація про пацієнта сайту. У ній зображується, що ключовим полем є ідентифікаційний код пацієнта, тобто поле «id», із значенням «Primary». Усі інші поля підпорядковуються полю «id».

Таблиця 2.5

Функціональні залежності між атрибутами сутності «appointments»

Найменування атрибутів	Функціональні залежності
id email first_name last_name idlikar idworkdays	<div> <div></div> <div>←</div> <div>←</div> <div>←</div> <div>←</div> <div>←</div> </div>

2.6 Даталогічне проектування бази даних

Даталогічний аналіз і проектування – це остаточне проектування реляційної схеми даних з врахуванням середовища обраної СУБД[20]. Проектування виконується на основі інфологічної схеми і полягає у виконанні таких пунктів:

- виділення таблиць (R-відношень);
- визначення фізичних форматів атрибутів на основі типів СУБД;
- визначення складу індексованих полів;
- визначення зв'язків навігації і логічної цілісності;
- проектування представлень;
- проведення нормалізації реляційних схем (усунення надлишковості, багатозначності).

Результат даталогічного проектування – реляційна схема бази даних.

2.6.1 Склад таблиць бази даних

Структура MySQL трирівнева: бази даних – таблиці – записи. Бази даних і таблиці MySQL фізично представляються файлами з розширеннями frm, MYD, MYI. Ім'я бази даних MySQL унікальна в межах системи, а таблиці - в межах бази даних, поля - в межах таблиці. Таблиці складаються із записів, а записи, у свою чергу, складаються з полів. Поле має два атрибути - ім'я і тип даних.

Основні типи даних, які використовуються в базі даних MySQL:

- VARCHAR – може зберігати не більше 255 символів.
- TIME – час в форматі ГГ: ХХ: СС.
- INT – діапазон від -2 147 483 648 до 2 147 483 647;

Для початку, в кореневому каталозі phpMyAdmin створимо нову базу даних, яка міститиме таблиці даних поліклініки. У рядку «Створити нову БД» введемо назву бази даних «clinicshelbase» і обираємо кодування utf8_general_ci. Таке кодування дозволяє зберігати в таблицях даних символи кирилиці (при використанні іншого типу кодування, текстова інформація веб-сторінки може відображатись некоректно). Отже, після того, як створена база даних «clinicshelbase», наповнимо її таблицями, які будуть містити інформацію зі сторінок веб-сайту.

Таблиця 2.6

Склад таблиці «register_db»

№п/п	Найменування атрибутів	Тип полів	Розмір полів
1	id	INT	11
2	email	VARCHAR	255
3	first_name	VARCHAR	255
4	last_name	VARCHAR	255
5	password	VARCHAR	255
6	confirm_password	VARCHAR	255

Далі розглянемо з яких атрибутів, типів, полів та їх розмірів складається таблиці «likar» і «workdays».

Таблиця 2.7

Склад таблиці «likar»

№п/п	Найменування атрибутів	Тип полів	Розмір полів
1	id	INT	11
2	firstname	VARCHAR	255
3	lastname	VARCHAR	255
4	speciality	VARCHAR	255
5	number	INT	20
6	cabinet_number	INT	10

Таблиця 2.8

Склад таблиці «workdays»

№п/п	Найменування атрибутів	Тип полів	Розмір полів
1	id	INT	11
2	day	VARCHAR	255
3	start_time	TIME	-
4	end_time	TIME	-

Таблиця 2.9

Склад таблиці «admin_login»

№п/п	Найменування атрибутів	Тип полів	Розмір полів
1	id	INT	11
2	email	VARCHAR	255
3	name	VARCHAR	255
4	password	VARCHAR	255

Таблиця 2.10

Склад таблиці «appointments»

№п/п	Найменування атрибутів	Тип полів	Розмір полів
1	id	INT	11
2	email	VARCHAR	255
3	first_name	VARCHAR	255
4	last_name	VARCHAR	255
5	idlikar	INT	11
6	idworkdays	INT	11

2.7 Запити до таблиць бази даних

Web-сайт реалізує такі запити до бази даних:

1. Запити типу «SELECT» :

-(SELECT * FROM likar) – використовується для вибору всієї інформації з таблиці likar;

-(SELECT * FROM workdays) – використовується для вибору всієї інформації з таблиці workdays;

-(SELECT * FROM appointments) – використовується для вибору всієї інформації з таблиці appointments;

Прикладом використання запиту SELECT у коді є:

```
$sql= "SELECT email, password FROM register_db WHERE email = ".$email."
AND password =
".$password." limit 1";
$result = $conn->query($sql);
if($result->num_rows==1){
    $_SESSION['is_login']=true;
    $_SESSION['email'] = $email;
```

```

echo "<script> location.href='profile.php';</script>";
}else{
    $remsg= '<div class="alert">Enter valid email and password</div>';
}
}
}else{
    exit;
    echo '<div class="alert alert-warning mt-2">Enter valid email and password</div>';
}

```

- ```

1. $sql = "SELECT email, first_name, last_name FROM register_db WHERE
 email = '$email'";
 $result = $conn->query($sql);
 if($result->num_rows ==1){
 $row = $result->fetch_assoc();
 $first_name = $row['first_name'];
 $last_name = $row['last_name'];
 }

```
- ```

2. <select name="likar" class="form-control" id="likar">
    <option value="">select doctor</option>
    <?php
        include('dbConnection.php');
        $query=mysqli_query($conn,"SELECT * FROM likar");
        while ($row=mysqli_fetch_array($query)) {
            ?>
            <option value="<?php echo $row['6'];?>"><?php echo $row['firstname'].
            '$row['lastname']. ' / '.$row['speciality']. ' / '.$row['number']. '
            / '.$row['cabinet_number'];?></option>
            <?php
        }

```

```
?>
```

```
</select>
```

3. <select name="workdays" class="form-control" id="workdays">

```
<option value="">select day</option>
```

```
<?php
```

```
include('dbConnection.php');
```

```
$query=mysqli_query($conn,"SELECT * FROM workdays");
```

```
while ($row=mysqli_fetch_array($query)) {
```

```
?>
```

```
<option value="<?php echo $row['id'];?>"><?php echo $row['day'].' -
```

```
',$row['start_time'].' -',$row['end_time'];?></option>
```

```
<?php
```

```
}
```

```
?>
```

```
</select>
```

4. <?php

```
$sql = "SELECT * FROM appointments";
```

```
$result = $conn-> query($sql);
```

```
if ($result->num_rows > 0) {
```

```
while ($row = $result-> fetch_assoc()) {
```

```
?>
```

```
<table>
```

```
<tr>
```

```
<th>Email</th>
```

```
<th>First name</th>
```

```
<th>Last name</th>
```

```
<th>Id likar</th>
```

```
<th>Id workdays</th>
```

```

</tr>
<?php
    echo "<tr><td>" . $row["email"] . "</td><td>" . $row["first_name"] .
    "</td><td>" . $row["last_name"] . "</td><td>" . $row["likar"] . "</td><td>" .
    $row["workdays"] . "</td></tr>";
}
}
else { ?><div class="no-records"><h>Not found
appointments<h/></div><?php
}
$conn-> close(); ?>

```

Запити типу «INSERT» :

Заповнення таблиці “likar”:

```

-(INSERT INTO `likar`(`id`, `firstname`, `lastname`, `speciality`, `number`,
`cabinet_number`) VALUES (1,' Mary Yaroslavivna', 'Ponomarenko', 'terapevt',
987654321', '6'));
-(INSERT INTO `likar`(`id`, `firstname`, `lastname`, `speciality`, `number`,
`cabinet_number`) VALUES (2,' Lilya Ivanivna', 'Matkiv', 'urolog', '675423123',
'3'));
-(INSERT INTO `likar`(`id`, `firstname`, `lastname`, `speciality`, `number`,
`cabinet_number`) VALUES (3,' Tanya Petrivna', 'Zired', 'dentist', '965645321',
'2'));
-(INSERT INTO `likar`(`id`, `firstname`, `lastname`, `speciality`, `number`,
`cabinet_number`) VALUES (4,' Albina Igorivna', 'Monter', 'neurologist',
954567899', '1'));
-(INSERT INTO `likar`(`id`, `firstname`, `lastname`, `speciality`, `number`,
`cabinet_number`) VALUES (5,' Mila Oleksandrivna', 'Merly', 'oculist',
987898765', '5'));

```

```
-(INSERT INTO `likar`(`id`, `firstname`, `lastname`, `speciality`, `number`,
`cabinet_number`) VALUES (6, 'Yulia Andriivna', 'Venger', 'gynecologist',
987655678', '4'));
```

Заповнення таблиці “appointments”:

```
INSERT INTO `admin_login`(`id`, `email`, `name`, `password`) VALUES
('1', 'admin@gmail.com', 'Admin', 'admin');
```

Прикладом використання запиту INSERT у коді є:

```
$email = $_REQUEST['email'];
$first_name = $_REQUEST['first_name'];
$last_name = $_REQUEST['last_name'];
$password = $_REQUEST['password'];
$confirm_password = $_REQUEST['confirm_password'];
$sql = "INSERT INTO register_db(email, first_name, last_name, password,
confirm_password)
VALUES('$email', '$first_name', '$last_name', '$password', '$confirm_password')";
if ($conn->query($sql) === FALSE)
    echo "Error: " . $sql . "<br>" . $conn->error;
$conn->close();
}
```

Заповнення таблиці “workdays”:

```
-(INSERT INTO `workdays`(`id`, `day`, `start_time`, `end_time`) VALUES
(1, 'Monday', '09:00:00', '09:30:00'));
```

```
-(INSERT INTO `workdays`(`id`, `day`, `start_time`, `end_time`) VALUES
(1, 'Monday', '09:30:00', '10:00:00'));
```

І відповідно до кожного дня тижня з понеділка по п’ятницю з інтервалом в 30 хв.

РОЗДІЛ 3. ІНСТРУКЦІЯ З ВИКОРИСТАННЯ WEB-САЙТУ

3.1 Програмна реалізація проекту

В якості середовища для написання коду було обрано програмний продукт ХАМРР. ХАМРР — 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 HTML. Для надання розмітці стилів використовувалися каскадні таблиці стилів CSS. Було обрано варіант блочної верстки з використанням фреймворку Bootstrap.

Дана програма реалізована у вигляді web-сайту, який працює через будь-який доступний браузер.

Для адміністрування системи управління базами даних MySQL використовується веб-застосунок з графічним веб-інтерфейсом, розроблений на мові програмування PHP – phpMyAdmin.

Після завершення верстки сайту, потрібно підключити форми(реєстрації, авторизації, запису на прийом, форму авторизації для адміністратора) до бази даних, в нашому випадку до phpMyAdmin.

Приклад підключення до бази даних:

```
<?php
$db_host = "localhost";
$db_user = "root";
$db_password = "";
$db_name = "clinicshelbase";
$conn = new mysqli($db_host, $db_user, $db_password, $db_name);
if($conn->connect_error){
    die('Connection failed');
}
```

```
//else {
//echo "Connect";
//}?>
```

3.2 Структура веб-сайту

Доступ до сайту відбувається через адресу: <http://clinicshel/index.html>

Вигляд діаграми варіантів використання для користувачів продукту (клієнт) на рисунку 3.1



Рис. 3.1 – Діаграма варіантів використання для користувача

Вигляд діаграми варіантів використання для адміністратора продукту на рисунку 3.2



Рис. 3.2 – Діаграма варіантів використання для адміністратора

Клієнтська частина має такі головні сторінки:

- Clinic_Shell – домашня сторінка, дозволяє повернутись на головну сторінку сайту з будь-якої іншої сторінки;
- About us – сторінка, яка містить в собі інформацію про поліклініку;
- Contacts – сторінка, яка містить в собі контактну інформацію поліклініки;
- Log in – сторінка, яка містить в собі форму авторизації на особистий профіль користувача;
- Sign up – сторінка, яка містить форму реєстрації, за допомогою якої користувач може зареєструватись в базу даних поліклініки;

Після того, коли користувач зареєструвався в системі, він може зайти в свій особистий профіль, в якому міститься інформація про користувача, можна перейти на іншу сторінку, на якій є форма запису на прийом до лікаря. Відповідно є функція виходу з особистого профілю.

3.3 Програмування клієнтської частини

Вигляд сторінок подано у додатках. Опис наведено нижче:

1. Web-сторінка: <http://clinicshel/index.html> знаходиться в клієнтській частині сайту. Являє собою головну сторінку (Додаток А);
2. Web-сторінка: http://clinicshel/about_us.html пропонує ознайомитись користувачу з інформацією про поліклініку (Додаток Б);
3. Web-сторінка: <http://clinicshel/contacts.html> пропонує ознайомитись користувачу з контактною інформацією поліклініки (Додаток В);
4. Web-сторінка: <http://clinicshel/register.php> дозволяє користувачу зареєструватись в базі поліклініки (Додаток Д) ;
5. Web-сторінка: <http://clinicshel/login.php> дозволяє користувачу залогуватись в системі (Додаток Є);

6. Web-сторінка: <http://clinicshel/submit.php> дозволяє користувачу записатись на прийом до лікаря (Додаток Ж);
7. Web-сторінка: <http://clinicshel/profile.php> дозволяє користувачу бачити свої дані (Додаток З);
8. Web-сторінка: <http://clinicshel/admin.php> дозволяє адміністратору авторизуватися в системі(Додаток И);
9. Web-сторінка: <http://clinicshel/adminprofile.php> дозволяє адміністратору бачити свої дані в профілі(Додаток Й);
10. Web-сторінка: <http://clinicshel/applicationssubmitted.php> дозволяє адміністратору бачити подані заявки на прийом(Додаток К);

3.4 Програмування серверної частини

PHP- скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера. PHP інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. Для програмування серверної частини веб-сайту і було використано PHP.

За допомогою скриптів PHP організовано такі операції для функціонування сайту:

1. Зберігання даних користувача, а саме: email, first name, last name, password.
2. Випадаючий список лікарів, які працюють в поліклініці.
3. Графік роботи лікарів.

Приклад PHP коду для зберігання даних користувача:

```
<?php
if(isset($_REQUEST['Submit'])) {
    include('dbConnection.php');
    date_default_timezone_set ("Europe/Kiev");
    $email = $_REQUEST['email'];
```

```

$first_name = $_REQUEST['first_name'];
$last_name = $_REQUEST['last_name'];
$password = $_REQUEST['password'];
$confirm_password = $_REQUEST['confirm_password'];
$sql = "INSERT INTO register_db(email, first_name, last_name, password,
confirm_password)
VALUES('$email', '$first_name', '$last_name', '$password',
'$confirm_password')";
if ($conn->query($sql) === FALSE)
    echo "Error: " . $sql . "<br>" . $conn->error;
$conn->close();} ?>

```

3.5 Макети сторінок веб-сайту

Сторінки веб-сайту мають меню, яке розміщене праворуч і вверху сторінки, основними елемента якого є логотип та перелік сторінок.

Вигляд меню показано на рисунку 3.3



Рисунок 3.3 Вигляд меню

Сторінка ClInIc_ShElL, як показано на Рисунок 3.4 складається з меню, перехід між якими можна здійснювати за допомогою вибору іншої сторінки, надтексту і підтексту головного гасла поліклініки, логотип.

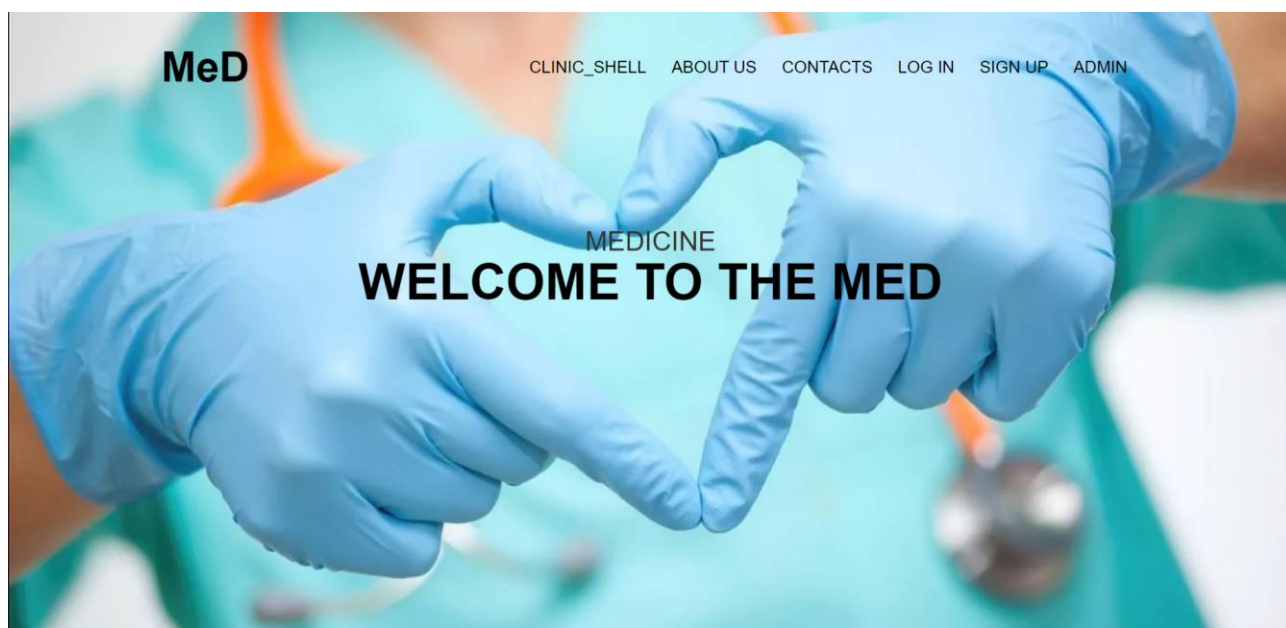


Рисунок 3.4 Вигляд сторінки CLInIc_ShElL

Сторінка About us, як показано на Рисунок 3.5 складається з блоків інформації про сам медичний заклад, меню та логотипу.

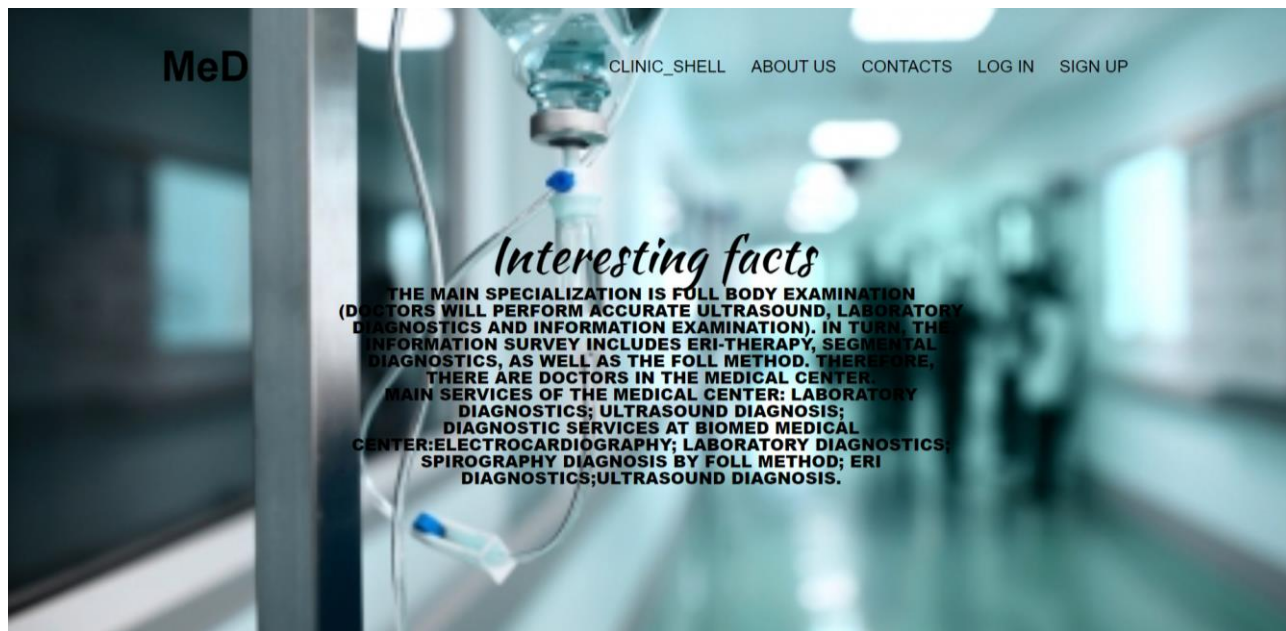


Рисунок 3.5 Вигляд сторінки About us

Сторінка Contacts, яку показано на Рисунок 3.6 складається з блоку контактної інформації закладу, меню та логотипу.



Рисунок 3.6 Вигляд сторінки About us

ВИСНОВКИ

В Розділі 1 було опрацьовано предметну область системи, визначено переваги та недоліки розборки веб-сайту, визначено характеристики бізнес-процесу.

У даному розділі було проаналізовано предметну область для розроблюваного продукту. Досліджено структуру та напрями діяльності об'єкту управління. Також були проаналізовані усі бізнес-процеси, які відбуваються у даному об'єкті управління. Подано характеристику процесу реєстрації користувача в системі та характеристику процесу реєстрації на прийом до лікаря.

В Розділі 2 було спроектовано та описано структуру БД, її таблиці та записи. Визначено типи даних та функціональні залежності. Розроблено запити та визначено атрибути полів. Для кращого розуміння архітектури продукту було розроблено діаграми, які показують взаємодію користувачів та їх функцій між собою та у системі.

Оскільки дана система оперує даними, було спроектовано архітектуру бази даних. База даних є реляційною, була створена діаграма корпоративної моделі даних, яка показує об'єкти системи та зв'язки між ними (ER-діаграма). Вона показує взаємодію між відношеннями і дозволяє приступити до програмування бази даних.

Розділ 3. В ході практичного завдання було розроблено веб-сайт приватної поліклініки “CIIInIc_ShEIL” використовуючи мову програмування PHP.

Програмна реалізація веб-сайту почалася з верстки шаблону сайту.

Верстка виконувалася мовою розмітки гіпертексту HTML. Для надання розмітці стилів використовувалися каскадні таблиці стилів CSS. Було обрано варіант блочної верстки з використанням фреймворку Bootstrap.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

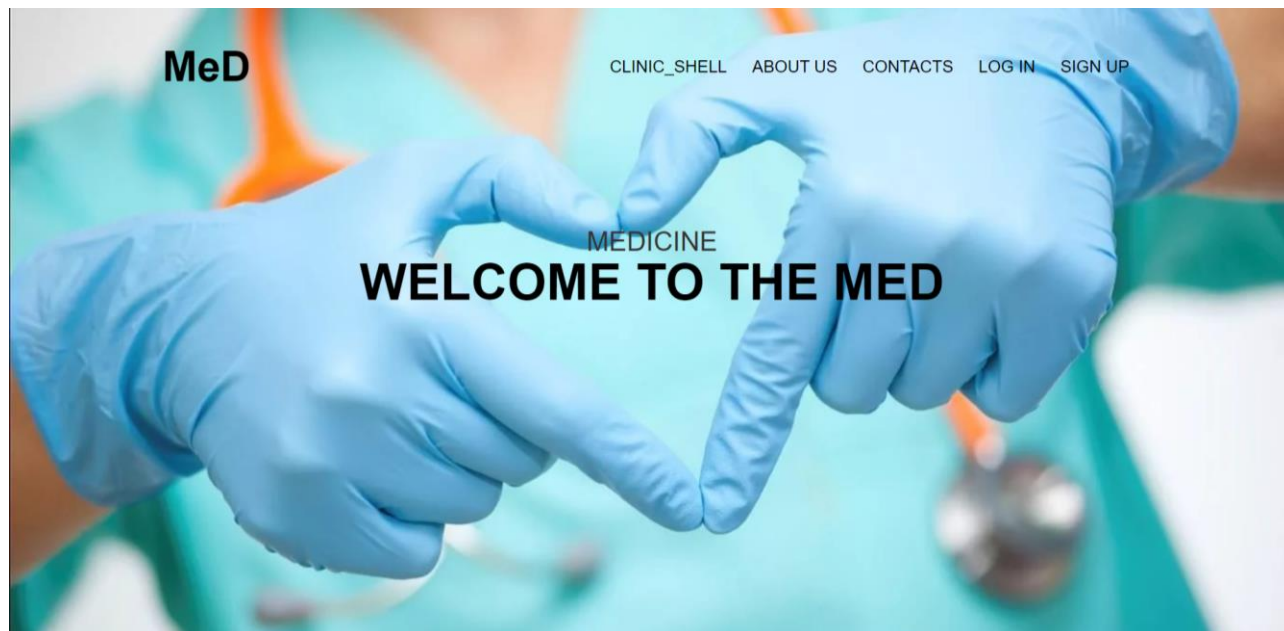
1. Буч Г. Язык UML: руководство пользователя / Г. Буч, Д. Рамбо, И. Якобсон // [пер. с англ. Н. Мухина]. – Москва. – 2019. – 493 с.
2. Дивак М.П. Системний аналіз та проектування КІС /М.П.Дивак// Навчальний посібник – Т.: Економічна думка. – 2018.
3. Калбертсон Роберт, Браун Крис, Кобб Гэри Быстрое тестирование. — М.: «Вильямс», 2017. — 374 с.
4. Костарев А. Ф. PHP 5. — Спб.: «БХВ-Петербург», 2016. — С. 1104.
5. Кузнецов Максим, Симдянов Игорь. PHP. Практика создания Webсайтов. — 2-е изд. перераб. и доп. — Спб.: «БХВ-Петербург», 2018. — С. 1264.
6. 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», 2016. — 464 с
7. Мэтт Зандстра. PHP: объекты, шаблоны и методики программирования, 3-е издание = PHP Objects, Patterns and Practice, Third Edition. — М.: «Вильямс», 2015. — С. 560
8. Пасічник В. В. Організація баз даних та знань / В.В. Пасічник, В.А. Резніченко. – К.: Видавнича група BVH, 2017. – 384 с.
9. Сеницын С. В., Налютин Н. Ю. Верификация программного обеспечения. — М.: БИНОМ, 2016. — 368 с.
10. Том 1. Основы, 7-е изд.» : підручник / К. С. Хорстманн, Г. Корнелл : Издательский дом "Вильямс", 2007. – 896 с.
11. Юр'єв В.К., Куценко Г.І. Громадське здоров'я та охорона здоров'я. СП, 2016. - С. 240-283.
12. CSS styles [Електронний ресурс]. - Режим доступу: <http://htmlbook.ru/css>
13. Dash, Python [Електронний ресурс]. – Режим доступу: <https://dash.plot.ly>.
14. Gosling The Java Language Specification Java SE 8 Edition [Електронний ресурс]. – Режим доступу: <http://docs.oracle.com/javase/specs/jls/se8/jls8.pdf>
15. PHP [Електронний ресурс]. <https://en.wikipedia.org/wiki/PHP>

- 16.PHP Developer [Електронний ресурс]. - Режим доступу:
https://dev.to/full_stackgeek/5-articles-being-a-php-developer-you-should-read-2lj9
- 17.PHP documentation [Електронний ресурс]. - Режим доступу:
<http://php.net/docs.php/>
- 18.Web developer information website [Електронний ресурс]. - Режим доступу:
<http://www.w3schools.com/>
- 19.Web developer information website [Електронний ресурс]. - Режим доступу:
<https://uk.wikipedia.org/wiki/PHP>
- 20.Вікіпедія. [Електронний ресурс]. - Режим доступу: <https://uk.wikipedia.org>
- 21.Cpp-reference. [Електронний ресурс]. – Режим доступу : <http://cpp-reference.ru/patterns/behavioral-patterns/command/>
- 22.Статті PHP[Електронний ресурс]. - Режим доступу:
<https://tutorialzine.com/tag/php>

ДОДАТКИ

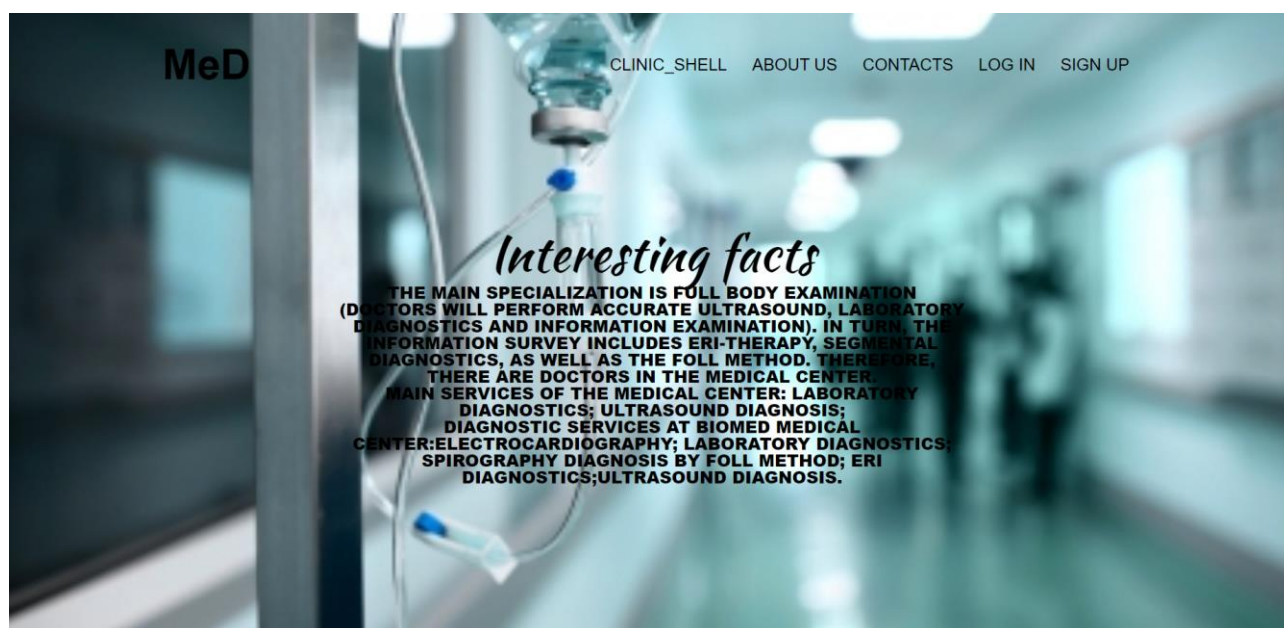
Додаток А

Головна сторінка

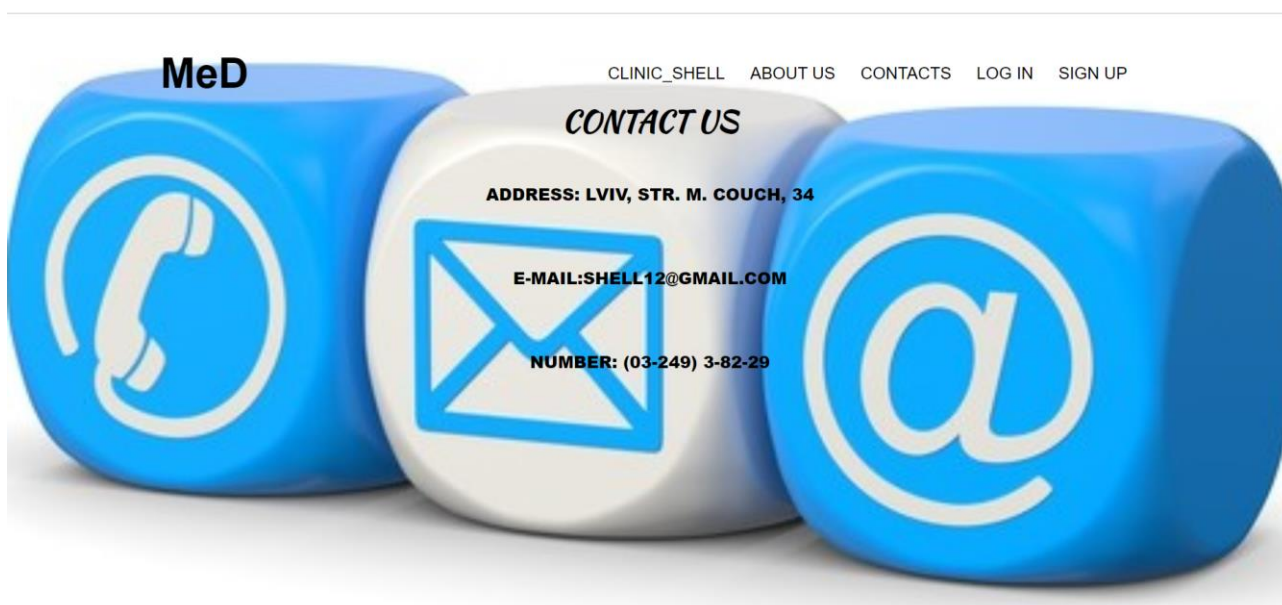


Додаток Б

Сторінка про поліклініку



Контактна інформація поліклініки



Сторінка форми реєстрації

The image shows a website header for 'MeD' with a navigation menu: CLINIC_SHELL, ABOUT US, CONTACTS, LOG IN, and SIGN UP. Below the menu is a 'Sign Up' form. The form has the following fields: Email, First name, Last name, Password, and Confirm Password. There are 'Submit' and 'Reset' buttons at the bottom of the form. A link 'Already have an account? Login here.' is located below the buttons. The background of the form is a blurred image of a person in a blue lab coat and gloves.

Сторінка форми авторизації

MeD

CLINIC_SHELL ABOUT US CONTACTS LOG IN SIGN UP

Log in

Email

Password

[Login](#) [Reset](#)

No personal profile? [Sign up!](#)

Сторінка форми запису на прийом

Make an appointment

Email

First name

Last name

Select a doctor:

Day/appointments

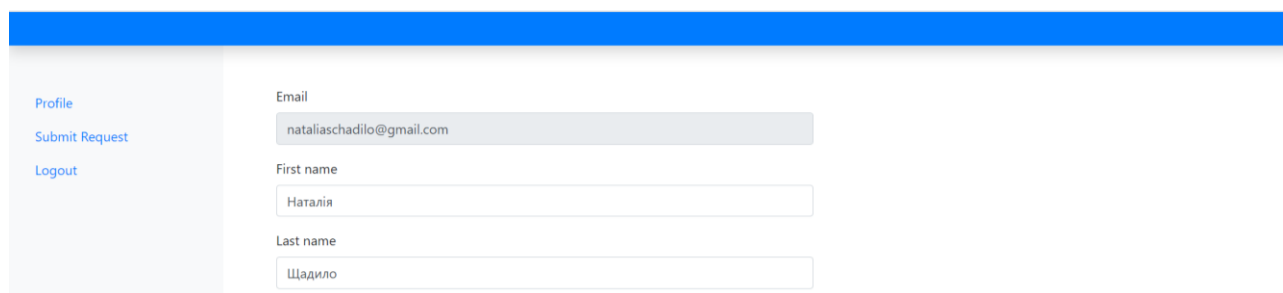
[Submit](#) [Reset](#)

Profile

Submit Request

Logout

Сторінка особистого профілю користувача



Profile

Submit Request

Logout

Email

nataliaschadilo@gmail.com

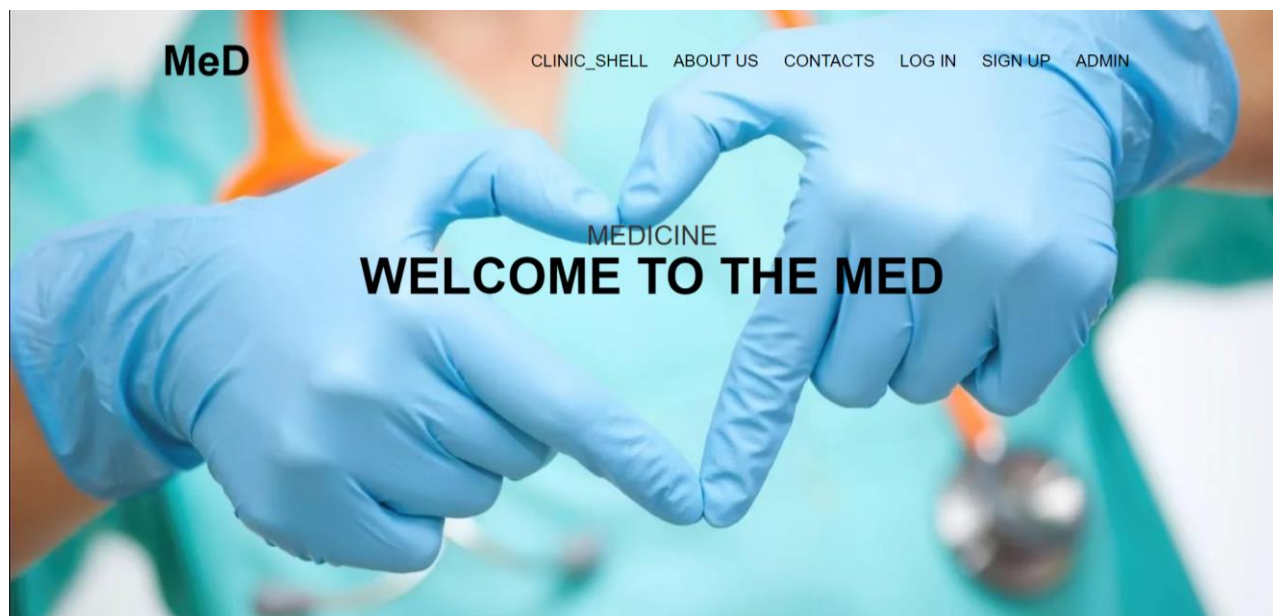
First name

Наталія

Last name

Щадило

Сторінка форми авторизації адміністратора



Сторінка особистого профілю адміністратора

[Profile](#)
[Applications submitted](#)
[Logout](#)

Email

admin@gmail.com

Name

Admin

Сторінка поданих заявок на прийом, які переглядає адміністратор

[Profile](#)
[Applications submitted](#)
[Logout](#)

Id patients	Email	First name	Last name	Id likar	Id workdays
1	nataliaschadilo@gmail.com	Наталія	Щадило	3	3
Id patients	Email	First name	Last name	Id likar	Id workdays
2	bodya@gmail.com	Богдан	Марк	4	16
Id patients	Email	First name	Last name	Id likar	Id workdays
3	nataliaschadilo@gmail.com	Наталія	Щадило	4	1
Id patients	Email	First name	Last name	Id likar	Id workdays
4	marlya@gmail.com	Марія	Щадило	2	21