КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук та інформаційних технологій

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Розробка системи обробки електронних заяв»

Студента 4 курсу, 9 групи, Лаврика Іллі Володимировича

спеціальності
122 «Комп'ютерні науки»

_____
підпис студента

Науковий керівник
доктор технічних наук, професор _____ Краскевич Валерій Євгенович
підпис керівника

Гарант освітньої програми
кандидат технічних наук, професор _____ Демідов Павло Георгійович
підпис керівника

Київ 2020

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«20» грудня 2019р.

Завдання
на випускний кваліфікаційний проект студенту

Лаврику Іллі Володимировичу
(прізвище, ім'я, по батькові)

1.Тема випускного кваліфікаційного проекту
«Розробка системи обробки електронних заяв»
Затверджена наказом ректора від *«04» грудня 2019 р. № 4111*
2. Строк здачі студентом закінченої роботи *12 червня 2020 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: дослідження та розробка системи обробки електронних заяв на прикладі медичного закладу.
Об'єкт дослідження: процеси прийому та обробки електронних заяв.
Предмет дослідження: моделі, методи та інформаційні технології обробки електронних заяв.
4. Перелік графічного матеріалу_____
_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Краскевич В.Є.	15.12.2019 р.	15.12.2019 р.
2	Краскевич В.Є.	15.12.2019 р.	15.12.2019 р.
3	Краскевич В.Є.	15.12.2019 р.	15.12.2019 р.

6. Зміст випускного кваліфікаційного проекту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ ФУНКЦІОНУВАННЯ СИСТЕМИ ОБРОБКИ ЕЛЕКТРОННИХ ЗАЯВ

1.1. Теоретичний аналіз електронних заяв

1.2. Теоретичний аналіз методів побудови системи обробки електронних заяв

1.3. Технології розробки програмної реалізації системи обробки електронних заяв

РОЗДІЛ 2. ОГЛЯД МЕТОДИК АНАЛІЗУ ДАНИХ ТА ПОБУДОВИ СИСТЕМИ ОБРОБКИ ЕЛЕКТРОННИХ ЗАЯВ

2.1 Аналіз та розподіл даних

2.2. Методи побудови веб-сторінок

2.3. Середовище розробки програмної реалізації

РОЗДІЛ 3. СТВОРЕННЯ СИСТЕМИ ОБРОБКИ ЕЛЕКТРОННИХ ЗАЯВ

3.1. Розробка алгоритму побудови системи обробки електронних заяв

3.3. Моделювання процесів опрацювання електронних заяв

3.4. Програмна реалізація системи обробки електронних заяв

3.4. Технологія використання розробленої системи обробки електронних заяв

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускного кваліфікаційного проекту	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускного кваліфікаційного проекту*	*01.10.2019*	*01.10.2019*
2	*Розробка та затвердження завдання на випускний кваліфікаційний проект*	*15.12.2019*	*15.12.2019*
3	*Вступ*	*03.02.2020*	
4	*РОЗДІЛ 1. Теоретичні аспекти функціонування системи обробки електронних заяв*	*28.02.2020*	
5	*РОЗДІЛ 2. Огляд методик аналізу даних та побудови системи обробки електронних заяв*	*06.04.2020*	
6	*РОЗДІЛ 3. Створення системи обробки електронних заяв*	*12.05.2020*	
7	*Висновки*	*15.05.2020*	
8	*Здача випускного кваліфікаційного проекту на кафедру науковому керівнику*	*20.05.2020*	
9	*Попередній захист випускного кваліфікаційного проекту*	*03.06.2020*	
11	*Виправлення зауважень, зовнішнє рецензування випускного кваліфікаційного проекту*	*09.06.2020*	
12	*Представлення готового зшитого випускного кваліфікаційного проекту на кафедру*	*12.06.2020*	
13	*Публічний захист випускного кваліфікаційного проекту*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15» грудня 2019 р.

Керівник випускного кваліфікаційного проекту

Краскевич В.Є.
(прізвище, ініціали, підпис)

Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

Завдання прийняв студент-дипломник Лаврик І.В.
(прізвище, ініціали, підпис)

9. Відгук керівника випускного кваліфікаційного проекту

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Керівник випускного кваліфікаційного проекту

_____ _____ р.

(підпис, дата)

10. Висновок про випускний кваліфікаційний проект

Випускний кваліфікаційний проект студента _____

(прізвище, ініціали)

може бути допущено до захисту в екзаменаційній комісії.

Гарант освітньої програми_____Демідов П.Г.____

(підпис, прізвище, ініціали)

Завідувач кафедри _____Пурський О.І.____

(підпис, прізвище, ініціали)

«_____»_____2020 р.

Анотація

У випускному кваліфікаційному проекті проаналізовано поняття «електронна заява», її необхідність, функції та можливості в умовах сучасного розвитку суспільства. Проаналізовано методи та програмне забезпечення для побудови веб-додатку. Розглянуто види сучасних баз даних. Створено структуру збереження та відображення даних в інформаційній системі. Побудовано моделі веб-додатку та взаємодії користувачів з ним. Здійснено комплексну розробку програмного забезпечення – веб-додаток для подачі та обробки електронних заяв.

Ключові слова: веб-сайт, веб-додаток, електронна заява, обробка електронних заяв, інформаційна технологія.

Anotation

The final qualification project analyzes the concept of "electronic statement", its necessity, functions and capabilities in the modern development of society. Methods and software for creating a web application are analyzed. Types of modern databases are considered. The structure of data storage and display in the information system is created. Models of web applications and interactions with it are built. Comprehensive software development has been carried out - web application for submitting and processing electronic statement.

Keywords: website, web application, electronic statement, processing of electronic statements, information technology.

ЗМІСТ

ВСТУП

В умовах сучасності організація зв'язку з клієнтами є однією з головних задач будь-якої компанії, яка надає медичні послуги, вирішення якої дозволяє якісно покращити ефективність роботи компанії. Цей зв'язок часто встановлюється за допомогою різного виду заяв.

Заява – письмове чи усне звернення заявника, в якому викладається прохання, замовлення або пропозиція.

В наш час комп'ютерні технології отримали широку розповсюдженість. Це зумовило перехід багатьох послуг та сервісів в мережу Інтернет. Безліч послуг стали доступними для людей дистанційно. Сучасні організації, які займаються наданням медичних послуг, часто мають свій веб-сайт. Мережа Інтернет дозволяє легко та обширно встановити зв'язок з клієнтами. В багатьох галузях це використовується і дозволяє зберегти безліч часу як для клієнтів, так і для бізнесу.

Створення технології подачі та обробки електронних заяв на базі веб-сайту медичного центру підвищить ефективність роботи останнього. За допомогою неї клієнт в будь-який момент, знаходячись в будь-якому місці, може оформити заявку на відвідування лікаря, якусь медичну послугу, тощо. До переваг веб-сайту належить можливість залишити заявку з будь-якого пристрою, чи то ПК, чи ноутбук, планшет, смартфон, тощо. Сайт працює 24 години на добу, майже без перерв і вихідних, це є ще одною перевагою сайту над іншими способами зв’язку. Не має потреби чекати телефонної розмови. Заявка не може бути загубленою. Вся інформація буде надана на e-mail. Перевагами системи на базі веб-сайту є:

- Мобільність;
- Доступність;
- Простота;

Компанія, яка надає відповідні послуги отримує заявку та обробляє її в будь-який час, результат обробки автоматично надсилається клієнту. Все це зумовлює актуальність обраної теми, мету і завдання.

Мета роботи: дослідження та розробка системи обробки електронних заяв на прикладі медичного закладу.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- Проаналізувати структуру електронних заяв;
- Проаналізувати вимоги до веб-додатку;
- Дослідити методи побудови веб-додатків;
- Спроектувати структурну модель веб-додатку;
- Розробити інформаційно-логічну модель подачі та обробки заяв;
- Розробити веб-додаток системи обробки електронних заяв;

Об’єкт дослідження: процеси прийому та обробки електронних заяв.

Предмет дослідження: моделі, методи та інформаційні технології обробки електронних заяв.

Методи дослідження: теоретична основа дослідження базується на загальнонауковому аналітичному методі, та системному підході. Для вирішення поставлених задач практично застосовувалися такі методи:

- загальнонауковий аналітичний метод;
- методи побудови баз даних;
- метод модульного програмування;
- функціональне моделювання;
- моделювання даних;

Практичне значення. Програмна реалізація може бути використана для розгортання системи обробки електронних заяв на веб-сайті будь-якого медичного закладу.

РОЗДІЛ 1.
ТЕОРЕТИЧНІ АСПЕКТИ ФУНКЦІОНУВАННЯ СИСТЕМИ ОБРОБКИ ЕЛЕКТРОННИХ ЗАЯВ

1.1. Теоретичний аналіз електронних заяв

Електронна заява – це електронний документ, в якому заявник вказує необхідні, відповідно до її форми, дані, та подає компанії, до якої йде звернення, на сайті або надсилає на email.

Система обробки електронних заяв – інформаційна технологія, яка забезпечує механізм отримання та обробки електронних заяв. Аналізує отримані дані, автоматизує певні процеси та представляє комфортний інтерфейс для роботи оператора.

В медичній сфері заявки від клієнтів дозволяють оптимізувати організацію графіку роботи медичного персоналу, завчасно підготувати необхідне обладнання та підготуватись до роботи, зменшити черги та покращити роботу з клієнтами.

Основні види заяв клієнтів медичного центру:

- реєстрація на прийом до лікаря;
- реєстрація на медичну послугу;
- оформлення декларації з терапевтом;
- оформлення декларації з педіатром;
- оформлення декларації з сімейним лікарем.

Електронна заява до медичного центру має містити такі дані клієнта: прізвище, ім’я, ім'я по батькові, номер мобільного телефону, адресу електронної скриньки. Також необхідно надати клієнту можливості вказати обраний напрям, день та час прийому, та, у разі необхідності, додаткові дані або прохання.

1.2. Теоретичний аналіз технології подачі електронних заяв

Для подачі електронних заяв потрібно побудувати механізм та надати інтерфейс, який дозволить користувачам комфортно вказувати всю необхідну інформацію.

Інтерфейс для клієнтів має представляти форму для заповнення. Кожне поле повинно бути налаштованим на прийом лише відповідного типу даних. Форма не має бути навантаженої, це необхідно щоб клієнт міг швидко її заповнити, але в поданій заявці має бути достатньо інформації, задля коректної обробки заявки, та зменшення навантаження на оператора системи.

Перед побудовою системи обробки електронних заяв необхідно дослідити як різні медичні установи реалізували інтерфейс та способи подачі електронних заяв.

На рис 1.1 зображено приклад форми заявки для клієнтів одного медичного центру. Така форма заявки дозволяє вказати дуже мало інформації, з цього з’являється потреба роботи оператора додатково в телефонній розмові. Відповідно, таке рішення зменшує ефективність роботи оператора, та недостатньо використовуються можливості електронних заявок. Зі сторони клієнта, таке рішення теж не є комфортним, оскільки прибирає всі переваги електронної заявки, прирівнюючи її до звичайного телефонного дзвінка.

На рис 1.2 зображено форму заявки для клієнтів іншого медичного закладу. Потреба в інформації, яку може надати клієнт, тут більша. Є можливість повного вибору лікаря та певних послуг, поля для всіх необхідних контактних даних. Проте відсутнє поле для вказання побажань чи додаткової інформації від клієнта, та деякі поля не несуть в собі користі, лише ускладнюють заповнення форми клієнтом.

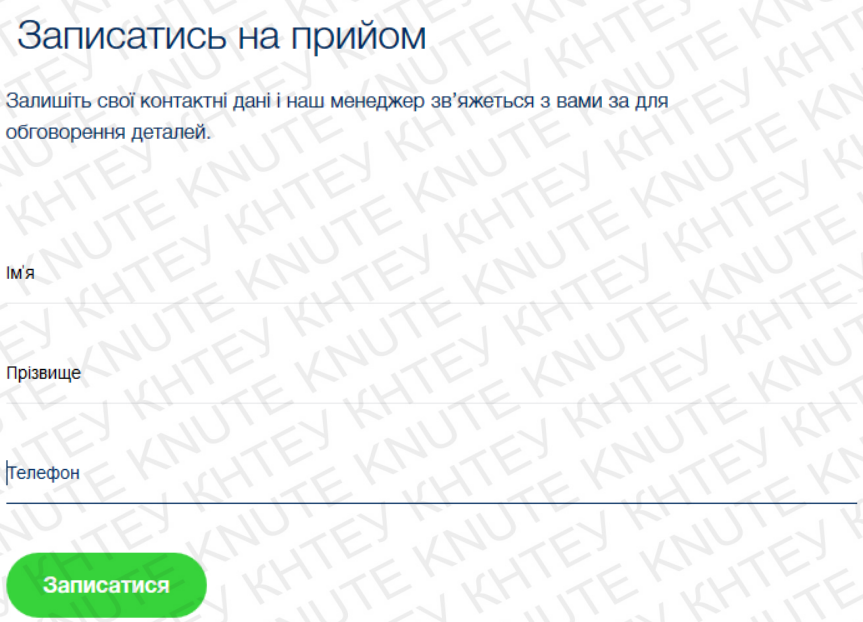

Рис. 1.1. Приклад форми заявки

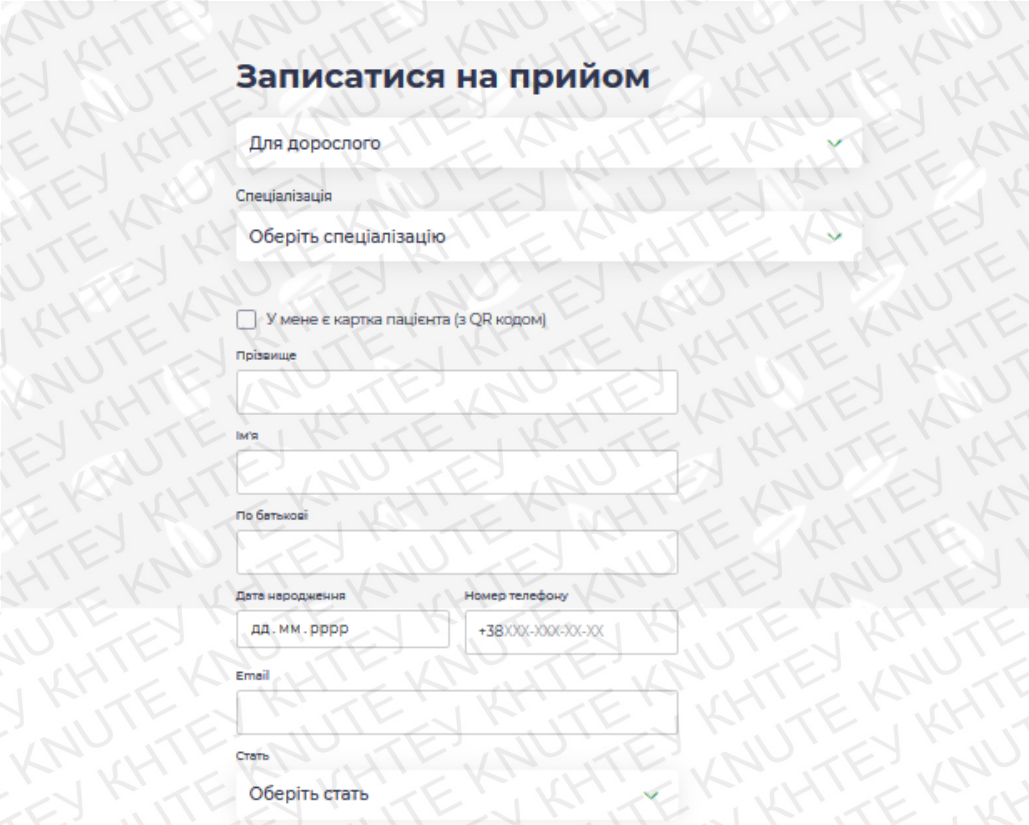

Рис. 1.2. Приклад форми заявки

1.3. Технології розробки програмної реалізації системи обробки електронних заяв

Для розробки системи обробки електронних заяв необхідно оглянути існуючі та можливі способи вирішення проблеми. Розглянуто найвідоміші сервіси, які надають власну систему роботи з заявками:

Formdesigner – це умовно безкоштовний онлайн сервіс, який надає конструктор форм для веб-сайтів (рис. 1.3). Також надає інтерфейс для роботи з формами в межах власної системи. Створені форми можна розповсюджувати за посиланням, або вбудовувати HTML-код у власний сайт. [1] Всі подані заявки та їх обробка зберігається на сервері системи Formdesigner.

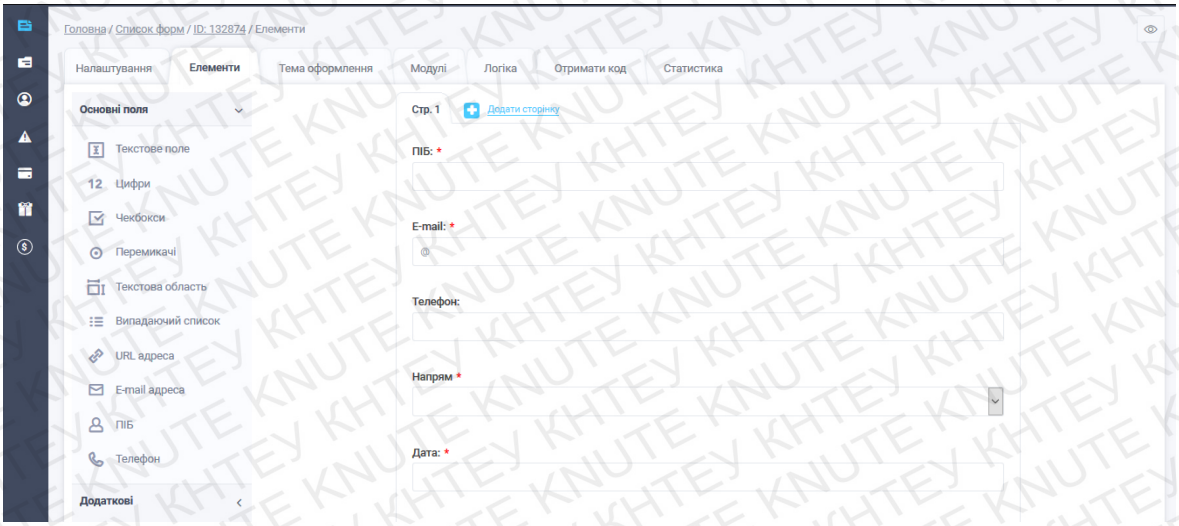

Рис. 1.3. Конструктор форм Formdesigner

Google Forms – безкоштовний онлайн сервіс від компанії Google. Основна функція сервісу – створення форм для різноманітних задач. В системі також є функціонал для обробки поданих заяв та перегляду статистики (рис. 1.4). Робота з заявками на веб-сайті Google Forms. Створену форму можна вбудувати у власний веб-сайт. [2]

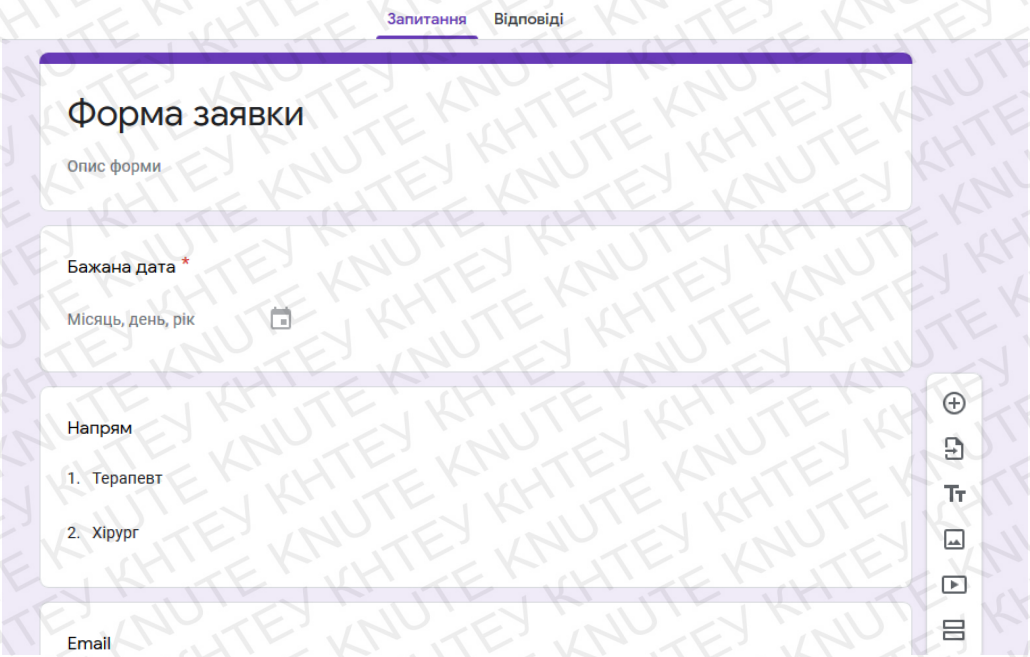

Рис. 1.4. Робота з Google Forms

Бітрікс24 – хмарний сервіс, розроблений на мові програмування PHP, який надає користувачам можливість побудувати веб-сайт за допомогою автоматизованого конструктору (рис. 1.5). Система містить в собі модуль для комунікації з клієнтами та обробки поданих заяв в інтерфейсі сервісу. [3]

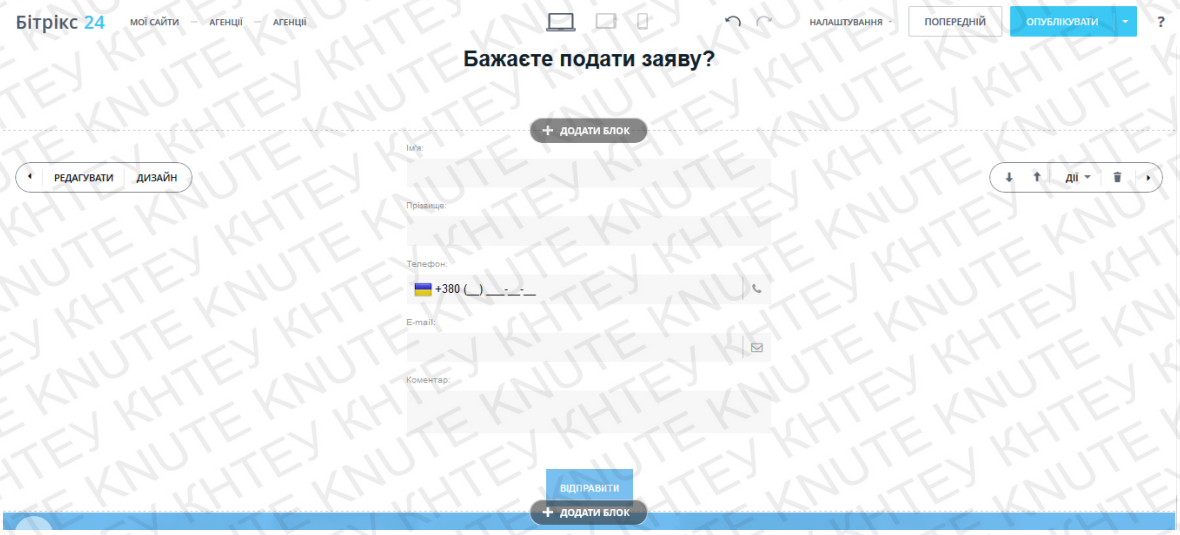

Рис. 1.5. Конструктор веб-сайтів Бітрікс24

Stepform – це універсальний конструктор для створення форм. Надає можливість розповсюджувати форми за посиланням, або вбудувати їх у власний веб-сайт. Обробка поданих заяв може відбуватися на веб-сайті сервісу (рис. 1.6) або перенаправлятися на необхідний сервер. [4].

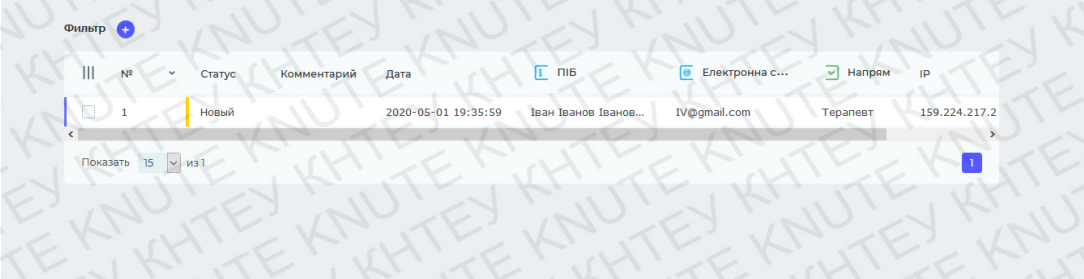

Рис. 1.6. Робота з заявами в Stepform

Головним недоліком існуючих рішень є те, що вони часто виступають посередником між клієнтом та організацією. Також важливо вказати що програмний код таких реалізацій є закритим, що ускладнює подальші модернізації чи зміни. Рівень безпеки багатьох готових рішень теж не є відомим. Всі ці недоліки суперечать так званій «тріаді кібербезпеки». Оскільки інформація вказана в заявах клієнтів зазвичай є лікарською таємницею, то дуже важливо забезпечити її захист відповідно до основ законодавства України про охорону здоров'я [5]. Саме тому обрано будувати веб-додаток власноруч, використовуючи ресурси з відкритим кодом, для забезпечення якості розробки.

РОЗДІЛ 2.
ОГЛЯД МЕТОДИК АНАЛІЗУ ДАНИХ ТА ПОБУДОВИ СИСТЕМИ ОБРОБКИ ЕЛЕКТРОННИХ ЗАЯВ

2.1. Аналіз та розподіл даних

Задля забезпечення якісного функціонування системи обробки електронних заяв, необхідно побудувати правильну структуру бази даних. Основними видами баз даних є реляційні та нереляційні.

Реляційна база даних – база даних, заснована на реляційній моделі даних. Дані за такою моделлю прийнято зображувати у вигляді таблиць (рис.2.1). Таблиці можуть мати між собою зв'язки які називають відношеннями, це дозволяє забезпечити поєднання даних з багатьох таблиць при виконанні запиту. Рядки таблиці представляють екземпляри цього типу сутності, їх також називають кортежами. Стовпці у свою чергу представляють значення, приписані цьому екземпляру, тому їх називають атрибутами. [6]

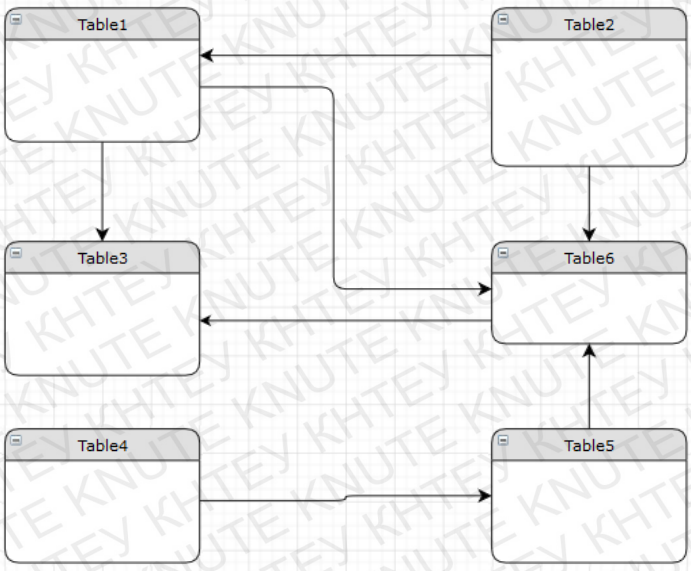

Рис. 2.1. Приклад структури реляційної бази даних

Переваги реляційних баз даних:

1. Відповідність ACID (атомарність, узгодженість, ізольованість, довговічність), що захищає цілісність бази даних, визначаючи точно, як транзакції взаємодіють з базою даних.

2. Кожна таблиця містить одну або декілька категорій даних у стовпцях.

3. Кожен рядок містить унікальний екземпляр даних для категорій, визначених стовпцями.

4. Користувач може отримати доступ до даних з бази даних, не знаючи структури таблиці бази даних.

Недоліки реляційних баз даних:

1. Масштабованість: користувачі повинні масштабувати реляційні бази даних на потужних серверах, які є дорогими та складними для обробки. Щоб масштабувати реляційну базу даних, вона повинна бути розподілена на декілька серверів.

2. Складність: дані сервера БД повинні бути вписані у таблиці. Якщо дані не вкладаються у таблиці, необхідно проектувати структуру бази даних, яка буде складною і важкою для обробки.

3. Вартість: реляційні системи керування базами даних (СКБД) вимагають дорогих систем зберігання та запатентованих серверів.[7-8]

Нереляційна база даних (NoSQL) не має чіткого визначення, так називають всі БД, які за механізмом зберігання та видобування даних відмінні від таблиць-відношень в реляційних базах даних (рис. 2.2).[9] Слід зазначити, що поява нереляційних баз даних була викликана декількома проблемами у використанні реляційних баз даних в межах роботи веб-додатків, а саме:

- значна кількість високонавантажених веб-додатків обробляють такі об’єми даних, які на порядок більше тих, що можуть бути оброблені реляційними базами даних;
- реляційні системи управління базами даних не завжди здатні забезпечити одночасний доступ до даних мільйонам користувачів Internet;
- на сьогодні існує потреба в обробці нетабличних даних.

Зазвичай дані нереляційних БД зберігаються у форматі ключ-значення, в документах або іншим методом без використання таблиць.Принципи формування системних показників соціально-економічного розвитку регіонів є першоосновою для формування теоретичної бази дослідження. Чітке визначення принципів дозволить побудувати ефективну систему показників та уникнути додаткових проблем під час аналізу [7-8].

1. Достовірність. Передбачає забезпечення дослідника інформацією, яка б відповідала часовим та просторовим рамкам об’єкту дослідження.

2. Об’єктивність. Показники повинні відображати реальну дійсність.

3. Однозначність трактування. Усі показники та способи їх розрахунку повинні базуватися на єдиній методиці, що давало б можливість здійснювати їх однозначну позитивну або негативну оцінку.

4. Порівнянність. Тобто забезпеченість їх співставності з аналогічними показниками інших регіонів.

5. Повнота охоплення. Показники повинні відображати широкий спектр соціально-економічних процесів, що відбуваються в регіоні.

6. Лаконічність. Система повинна містити таку кількість показників, яка б дозволяла охарактеризувати усі особливості розвитку регіону, що відображають якісні процеси в економіці та соціальній сфері.

7. Структурованість. Система показників повинна поділятися на окремі елементи (підсистеми) в залежності від процесів, що характеризуються тими чи іншими показниками. При цьому між усіма елементами системи мають існувати взаємозв’язки (між первинними показниками, індикаторами та індексами, між показниками окремих сфер функціонування та інші).

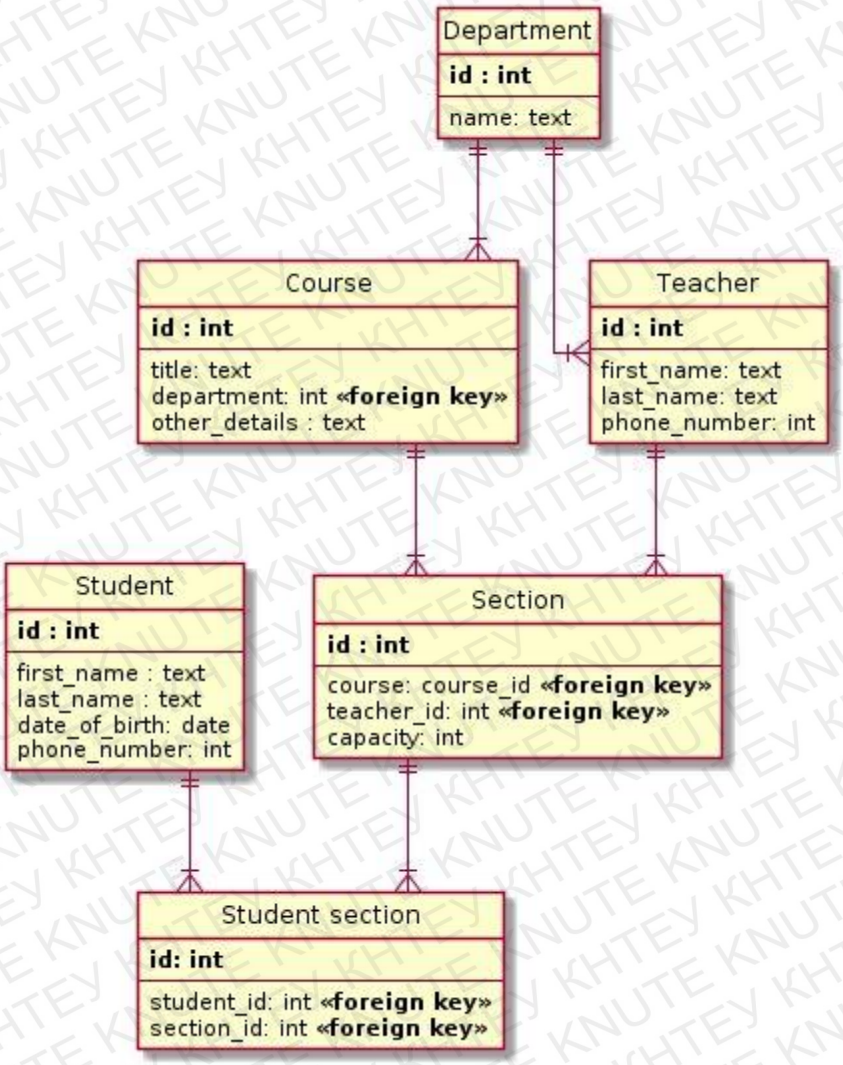

Рис. 2.2. Приклад структури нереляційної бази даних

Переваги нереляційних баз даних:

1. Зберігання великих обсягів даних, які погано структуровані. БД не обмежує типи даних, які можна зберігати разом, і дозволяє додавати нові типи. За допомогою баз даних на базі документів можна зберігати дані в одному місці, не визначаючи заздалегідь, які «типи» цих даних.

2. Швидша обробка даних, ніж у реляційних базах даних.

3. Дешевше: не реляційні бази даних використовують дешеві кластери товарних серверів для управління операціями та даними.

4. Краща масштабованість у порівнянні з реляційними базами даних. Проте нереляційні бази даних не повністю масштабовані у всіх ситуаціях.

Недоліки нереляційних баз даних:

1. Узгодженість даних: більшість нереляційних баз даних не виконують транзакції ACID. Натомість використовується принцип «кінцевої послідовності». Це забезпечує деякі переваги продуктивності, але це створює ризик того, що дані на одному вузлі бази даних можуть не синхронізуватися з даними іншого вузла;

2. Відсутність стандартизації: в NoSQL немає специфічного типу інтерфейсу бази даних або способу програмування. Мова дизайну та запитів баз даних NoSQL різко відрізняється між різними продуктами NoSQL - набагато ширше, ніж серед традиційних баз даних SQL.

3. Деякі не реляційні бази даних погано розповсюджуються на декілька вузлів. Якщо база даних не може відокремитись автоматично, вона не може автоматично збільшуватись або зменшуватись у відповідь на коливальний попит. [7-8]

Сучасні системи управління реляційними базами даних зазвичай створені на основі мови структурованих запитів – SQL. Найпопулярнішими представниками таких СУБД є:

- MySQL
- SQLite
- PostgreSQL
- Microsoft SQL Server
- Firebird

MySQL – одна з систем управління реляційними базами даних. Початковий код цього ПЗ є загальнодоступним, тому вивчити і модифікувати його може кожен користувач.

MySQL поділяється на дві частини: серверну і клієнтську.

Сервер MySQL постійно підтримується апаратним забезпеченням. Програми-клієнти посилають серверу SQL-запити за допомогою мережі, сервер їх обробляє і зберігає результат. Система працює так: клієнт вказує, яку інформацію він хоче отримати від сервера баз даних, а відповідно сервер баз даних посилає відповідь програмі-клієнту.

MySQL має трирівневу структуру: бази даних - таблиці - записи. Бази даних і таблиці MySQL являються файлами з розширеннями frm, MYD, MYI. На одному MySQL сервері може працювати одразу декілька баз даних, доступ до них можна розмежовувати логіном і паролем. [10]

SQLite - це внутрішня бібліотека, яка реалізує автономний, безсерверний, не потребуючий конфігурації, транзакційний двигун бази даних SQL. Код для SQLite знаходиться у відкритому доступі і тому є вільним для використання з будь-якою метою, комерційною чи приватною. SQLite - це найбільш широко розгорнута база даних у світі з великою

кількістю додатків. На відміну від більшості інших баз даних SQL, у SQLite немає окремого серверного процесу. SQLite читає і записує безпосередньо в звичайні файли диска. Повна база даних SQL з кількома таблицями, індексами, тригерами та переглядами міститься в одному файлі на диску. Формат файлу бази даних є кросплатформенним, тобто можна вільно копіювати базу даних між 32-бітовою та 64-бітовою системами або між архітектурами big-endian та little-endian. Ці особливості роблять SQLite популярним форматом файлів додатків. Файли баз даних SQLite - це рекомендований формат зберігання Бібліотекою Конгресу США. Бібліотека є дуже швидкою і може працювати при невеликій кількості пам’яті. [11]

PostgreSQL - це потужна об'єктно-реляційна база даних із відкритим кодом, яка використовує та розширює мову SQL у поєднанні з багатьма функціями, які безпечно зберігають та масштабують найскладніші навантаження даних. PostgreSQL оснащений багатьма функціями, спрямованими на допомогу розробникам у створенні програм, адміністраторам для захисту цілісності даних та побудови середовищ, стійких до відмов, та допомагають керувати вашими даними незалежно від того, наскільки великий чи малий набір даних. PostgreSQL намагається відповідати стандарту SQL, коли така відповідність не суперечить традиційним особливостям або може призвести до поганих архітектурних рішень. Багато функцій, необхідних стандарту SQL, підтримуються, хоча іноді мають дещо різний синтаксис або функції. [12]

Microsoft SQL Server — система управління базами даних, розроблена та підтримується корпорацією Microsoft. Сервер надає функції збереження та

надання даних при отриманні запиту від інших застосунків, при тому що останні можуть виконуватися на самому сервері, або в мережі.

Transact-SQL – спеціально створена мова запитів. Розроблена спільно Microsoft з Sybase. Transact-SQL є розширеною реалізацією стандарту ANSI/ISO щодо структурованої мови запитів SQL. Система використовується для баз даних різних розмірів, від невеликих, до колосальних, як на підприємствах. [13]

Firebird - це компактна, крос-платформна, вільна реляційна база даних, що пропонує безліч стандартних функцій ANSI SQL, працює на Linux, Windows та різноманітних платформах Unix. Firebird пропонує високу продуктивність та потужну підтримку мови для збережених процедур та тригерів. Він використовується у виробничих системах, під різними назвами, з 1981 року. [14]

Найпопулярнішими представниками систем управління нереляційними базами даних є:

- MongoDB
- CouchDB
- ArangoDB

MongoDB — документо-орієнтована система керування базами даних (СКБД). Код системи відкритий. Опис схеми таблиць не потребується. Систему MongoDB є швидкою та її легко маштабувати Вона оперують даними у форматі ключ/значення, та реляційними СКБД, функціональними і зручними у формуванні запитів. Система MongoDB написана мовою програмування C++, а поширюється в рамках ліцензії AGPLv3.

MongoDB має можливість зберігати документи в JSON-подібному форматі, мова формування запитів досить гнучка, має можливість створювати індекси для різних збережених атрибутів, забезпечує ефективне зберігання великих бінарних об'єктів, веде журнал операцій зі зміни і додавання даних в БД, має забезпечення аби працювати відповідно до парадигми Map/Reduce, Конфігурація відмовостійка та є підтримка реплікації. Розподіл даних, на основі певного ключа, комбінація якого з реплікацією даних, дозволяє побудувати маштабований кластер даних, з відсутньою точкою відмови (збій вузла не позначиться на роботі БД), по серверам забезпечується вбудованими засобами. Є автоматичне відновлення після збою і перерозподіл навантаження з вузла, якщо він вийшов з ладу. Зміна розмірів кластера чи перетворення сервера на кластер не зупиняє роботу БД просто додаючи нові машини. [15-16]

Apache CouchDB — розподілена документо-орієнтована система управління базами даних класу NoSQL-систем, що не вимагає опису схеми даних. CouchDB використовує декілька форматів та протоколів для зберігання, передачі та обробки своїх даних, він використовує JSON для зберігання даних, JavaScript як мову запиту за допомогою MapReduce та HTTP для API.

Кожна база даних в CouchDB - це сукупність незалежних документів. Кожен документ підтримує власні дані та автономну схему. Додаток може отримати доступ до декількох баз даних, так що одна зберігається на мобільному телефоні користувача, а інша на сервері. Метадані документа містять інформацію про перегляд, що дає змогу об'єднати будь-які відмінності, які могли виникнути під час відключення баз даних. CouchDB реалізує форму управління мультиверсійною паралельністю (MVCC), щоб не

блокувати файл бази даних під час запису. Конфлікти залишаються програмі для вирішення. Вирішення конфлікту зазвичай передбачає спочатку об'єднання даних в один із документів, а потім видалення старих документів. [17]

ArangoDB — система керування базами даних, являється відкритою, може застосовуватися в різних цілях, що надає гнучкі моделі зберігання документів, графів і даних у форматі ключ-значення. AQL мова запитів, яка нагадує SQL, необхідна для роботи з БД, також можна керувати за допомогою спеціального розширення написаного мовою JavaScript. Відповідає вимогам ACID (атомарність, узгодженість, ізольованість, надійність), підтримують транзакції, маштабованість горизонтальна й вертикальна. Управління проводиться за допомогою веб-інтерфейсу або консольного клієнту ArangoSH.

Головні особливості ArangoDB:

Дані структуруються у формі документів, в яких метадані та інформація про структуру відокремлені від користувацьких даних, та не потребують визначення схеми даних

ArangoDB може використовуватися як сервер для веб-додатків мовою JavaScript. REST/Web API забезпечує підтримку доступ до бази.

JavaScript можна використовувати для звернення до БД веб-додатків, та обробників за сторони СКБД

Модель зберігання даних досить гнучка, в ній можна комбінувати пари ключ-значення, документи і параметри, які визначають зв'язки між записами (надано засоби для обходу вершин графів).

Можна обирати тип індексу, який буде відповідати розв'язуваним задачам (індекс можна використовувати для пошуку повного тексту)

Надійність автоматично налаштовується: додатки самі визначать, в якому режимі працювати (вище надійність, чи швидкість).

Транзакції: запити можуть бути запущені водночас до кількох документів чи колекцій з опціональною узгодженістю транзакцій та ізоляцією

Підримується реплікація та шардинг: можна створити master-slave конфігурацій і рознести набір даних на різні сервери залежно від вказаної ознаки. [18]

2.2. Методи побудови веб-сторінки

Мова гіпертекстової розмітки (HyperText Murkup Language — HTML) використовується для створення і візуального відображення веб-сторінок. Вона становить основу кожної веб-сторінки в мережі Інтернет. Термін «гіпертекст» означає текст, сформований за допомогою мови розмітки і який містить гіперпосилання, якими зв'язані веб-сторінки одна з одною. HTML підтримує як візуальні зображення, так і інший медіаконтент. HTML є мовою, що описує структуру і семантику вмісту веб-документу. Контент веб-сторінки розмічений за допомогою тегів, що представляють HTML-елементи. Прикладами таких елементів є <head>, <title>, <body>, <article>, <section>, <p>, , , <img>, <picture>, і т.д. Ці елементи формують будівельні блоки для будь-якого веб-сайту. [19]

Каскадні таблиці стилів (CSS) – мова стилю сторінок, яка використовується для опису подання документа, написаного мовою розмітки. Вона дозволяє задавати стилі в обраних HTML документах. [20]

CSS розроблений таким чином, що дозволяє розділити презентацію та контент, включаючи макет, кольори та шрифти. Цей поділ може покращити доступність контенту, забезпечити більшу гнучкість та контроль у конкретизації характеристик презентації, дасть змогу декільком веб-

сторінкам обмінюватися форматуванням, вказавши відповідний CSS в окремому файлі .css, а також зменшити складність та повторення структурного вмісту.

Поділ форматування та контенту також робить можливим подання однієї і тієї ж сторінки розмітки в різних стилях для різних методів візуалізації, таких як екран, друк, голос (через мовленнєвий браузер чи зчитувач екрана) та на основі шрифту Брайля тактильні пристрої. CSS також має правила альтернативного форматування, якщо доступ до вмісту з мобільного пристрою. [21]

За допомогою CSS будується модульна система верстки. Кожен блок - це незалежний елемент, в який можна вкладати необмежену кількість елементів. Блок можна позиціонувати, міняти йому розміри, стилізувати. Блокові елементи можна накладати один на одного. З огляду на підтримку прозорості, можна домогтися безлічі цікавих ефектів, виділити якусь ділянку або реалізувати за допомогою скрипта завантаження різного вмісту в одному і тому ж місці. [22]

Табличний метод дозволяє побудувати каркас сайту за допомогою таблиць. Метод основується на тезі «table» та його елементах. Такий підхід дозволяє достатньо легко створити багато колонок та стовбців, проте робота з вкладеними елементами стає некомфортною і з’являються проблеми з адаптивністю відображення сайту на мобільних платформах.

2.3. Середовище програмування програмної реалізації

Веб-фреймворк - програмний каркас, який призначено для побудови веб-додатків, служб або ресурсів. Він спрощує розробку, додає автоматизацію і позбавляє від необхідності частих повторень коду. Багато фреймворків покращують роботу з базами даних.

Для розробки логіки веб-додатку необхідно підібрати мову програмування та фреймворк, який має потрібне функціональне забезпечення.

Найпопулярніші сучасні веб-фреймворки:

- Django
- Node.js
- Laravel
- jQuery
- Ruby on Rails
- Spring

Django – це відкритий безкоштовний високорівневий Python веб-фреймворк, що дозволяє швидко будувати безпечні та легко підтримувані сайти. [23]

Веб-сайт на Django будується з одного або декількох додатків, які бажано робити відчужуваними, що дозволяє їх легко підключити до інших проектів. Це одна з основних архітектурних відмінностей цього фреймворка від інших. [24]

Спочатку Django розроблявся, як засіб для роботи новинних ресурсів, це зумовило його архітектуру: в ньому є ряд засобів, які допомагають у швидкій та якісній розробці інформаційних веб-сайтів. Адміністративна частина веб-сайтів не потребує розробки, оскільки все вбудовано в Django, також є встановлений модуль для керування вмістом, який можна додати в будь-який веб-сайт, зроблений на Django, та який може контролювати водночас декількома сайтами на одному сервері. В адміністративному модулі реалізовано забезпечення, яке дозволяє створювати, змінювати і вилучати будь-які об'єкти наповнення сайту, кожна дія записується до журналу, а

також інтерфейс дозволяє налаштувати управління користувачами і групами з призначенням та зміною прав. [25]

Node.js — крос-платформний JavaScript фреймворк з відкритим кодом, який виконує код JavaScript за межами вебпереглядача. Node.js дозволяє розробникам використовувати JavaScript для написання інструментів командного рядка та для сценаріїв на стороні сервера - запуску скриптів на стороні сервера для створення динамічного вмісту веб-сторінки до відправки сторінки у вебпереглядач користувача. Node.js об'єднує розробку веб-додатків навколо однієї мови програмування, а не різних мов для скриптів на сервері та клієнтах. [26]

Laravel — безкоштовний, відкритий PHP-фреймворк, створений Taylor Otwell. Основне призначення – розробка веб-додатків відповідно до шаблону model–view–controller (MVC). Особливим його робить виділена модульна система упакування з вбудованим менеджером залежностей Composer, забезпечуються різні способи роботи з реляційними базами даних, утиліти, які спрощують розгортання додатків і технічне обслуговування. В Laravel є реалізована власна потужна екосистема, яка містить в собі різноманітні інструменти розробки, які зазвичай є безкоштовними. [27-28]

jQuery — це швидка, невелика та багатофункціональна бібліотека JavaScript. Завдяки простому у користуванні API, який працює в безлічі вебпереглядачів, такі речі, як обробка та маніпулювання документами HTML, обробка подій, анімація та Ajax, набагато простіші. [29]

Ruby on Rails — об'єктно-орієнтований фреймворк для побудови веб-додатків мовою програмування Ruby. Основується на шаблоні MVC для веб-додатків, та забезпечує їхню інтеграції з веб-сервером та базою даних. [30-31]

Розробка веб-додатків в Ruby on Rails базується на таких принципах:

- максимальне використання механізмів повторного використання, що дозволяють мінімізувати дублювання коду в додатках;
- за замовчуванням використовуються угоди по конфігурації, типові для більшості додатків - явна специфікація конфігурації потрібно тільки в нестандартних випадках.

Також фреймворк має власну екосистему плагінів RubyGems, наприклад:

- Devise — автентифікація;
- CanCanCan — авторизація;
- Kaminari, Will paginate, Pagy — розподіл записів БД;
- Active Admin — стоверення панелей адміністрування;
- CommunityEngine — створення соціальних мереж.

Spring Framework — це фреймворк з відкритим кодом та плагінами, які підтримують інверсію управління платформою Java. Основними особливостями Spring Framework можна скористатися за допомогою будь-якого Java-додатку, проте є розширення для створення веб-додатків на платформі Java EE. [32]

Spring Framework складається з кількох модулів, які надають широкий спектр послуг:

- Spring Core Container: конфігурація компонентів додатків і управління життєвим циклом об'єктів Java;

- Aspect-oriented programming: надає можливість реалізувати наскрізні процедури;
- Data access: забезпечує роботу з реляційною системою управління базами даних на Java з використанням JDBC та доступні об'єктно-реляційні відображення та інструменти з NoSQL баз даних;
- MVC: програмний каркас, що дозволяє створення веб-додатків і веб-служб RESTful.
- Spring Security: налаштовувана безпека, яка підтримують цілий ряд стандартів, протоколів та інструментів.
- JMX: конфігурація і управління Java-об'єктами для місцевої (локальної) або віддаленої конфігурації;
- Testing: підтримка класів, які дозволяють писати юніт-тести та інтеграційні тести;

РОЗДІЛ 3.
СТВОРЕННЯ СИСТЕМИ ОБРОБКИ ЕЛЕКТРОННИХ ЗАЯВ

3.1. Розробка алгоритму побудови системи обробки електронних заяв

Спочатку необхідно розподілити порядок розробки системи обробки електронних заяв:

- Визначення задач веб-додатку;
- Вибір архітектури бази даних та програмного забезпечення
- Побудова сайту для веб-додатку;
- Побудова бази даних;
- Створення функціонального забезпечення для подачі заяв;
- Створення функціонального забезпечення для обробки заяв;
- Верстка сторінок веб-додатку;
- Аналіз роботи веб-додатку.

Основними задачами веб-додатку буде створення, надсилання, прийом та обробка електронних заяв.

В якості архітектури програмного забезпечення було обрано фреймворк Django. Цей вибір зумовлений такими властивостями цього фреймворку як:

- Модульна структура, яка відповідає принципам об’єктно-орієнтованого програмування.
- Функціональне забезпечення;
- Активний розвиток розробниками та спільнотою;
- Комфорт у використанні.

На етапі «побудова сайту для веб-додатку» в фреймворку Django було створено веб-сторінку каркас, на основі якої відтворюються функціональні сторінки подачі та обробки електронних заяв.

Для побудови бази даних, для зберігання поданих клієнтами заяв, було обрано реляційну БД SQLite, оскільки всі необхідні дані клієнтів легко помістити в табличку, а робота з даними швидка, проста і ефективна.

На етапі «Створення функціонального забезпечення для подачі заяв» було зроблено форму, яка в собі містить поля для вказання: ПІБ, номеру мобільного телефону, адреси електронної скриньки, обраного напрямку, бажаної дати та часу прийому і місце для написання додаткової інформації (коментар).

На етапі «Створення функціонального забезпечення для обробки заяв» побудовано функцію для відображення отриманих заяв та обробки їх зі зворотнім повідомленням про результат обробки на електронну скриньку клієнта.

В результаті верстки сторінок веб-додатку було:

1. створено html-файли для відображення необхідних матеріалів веб-додатку;
2. налаштовано розміщення відповідних матеріалів;
3. налаштовано зовнішній вигляд веб-сторінок;

Етап «аналіз роботи веб-додатку» необхідний аби перевірити адекватність функціонування та відображення створеної системи та усунути знайдені несправності.

3.2. Моделювання процесів опрацювання електронних заяв

Перед розробкою системи обробки електронних заяв необхідно промоделювати її діяльність. За використанням уніфікованої мови моделювання (UML) було побудовано діаграми: послідовності та прецедентів.

Діаграма прецедентів або Use Case являє собою представлення взаємодії користувача з системою, яке показує взаємозв'язок між користувачем та різними випадками використання, в яких користувач бере участь.

На діаграмі представлено три актори (рис. 3.1). Перший з них – користувач, який має бажання подати заявку до медичного центру, є основним актором та виконує такі функції:

1. Вхід на сайт;
2. Заповнення форми заявки;
   a. Перезапис заяви
3. Надсилання заявки
4. Перегляд відповіді

До вторинних акторів належать: компанія та сервер. Сервер тримає сайт, виконує автоматизовані функції з фіксації та обробки заяв. Компанія адмініструє сайт, фіксує та обробляє заявки клієнтів.

Генеральний сценарій успіху виглядає так:

1. Користувач заповнює форму заяви;
2. Користувач надсилає заяву;
3. Компанія фіксує ім'я покупця, номер телефону, заявку тощо;
4. Компанія обробляє заявку
5. Компанія надсилає користувачу повідомлення з підтвердженням його заявки.

Розширення

1a. Перезапис заяви

Рис. 3.1. Use Case

Діаграма послідовності показує взаємодії об'єктів, розташованих у часовій послідовності. Вона зображує об'єкти та класи, що беруть участь у сценарії, та послідовність повідомлень, що обмінюються між об'єктами, необхідні для виконання функціональності сценарію. Діаграми послідовності зазвичай пов'язані з реалізацією випадку використання в логічному представленні системи, що розробляється.

На діаграмі (рис. 3.2) зображено такі об’єкти: користувач, сайт, форма, обробка заявки, база даних, результат обробки та компанія. Акторами є користувач і компанія.

Актор «користувач» представляє собою клієнта компанії, який має надіслати заявку аби отримати необхідні йому послуги.

Сайт — об’єкт, який надає необхідну користувачу інформацію, форму заяви, та спосіб її подачі.

Форма — об’єкт, який вказує користувачу, як правильно розподілити інформацію для подачі заяви.

Обробка заявки — об’єкт, в якому відбувається відповідна назві функція, отримуються та передаються дані до бази даних.

База даних — зберігає дані про заявки клієнтів, в тому числі їх статус оброблення.

Актор «компанія» контролює обробку заяв та адмініструє сайт.

Користувач заходить на сайт, заповнює форму, в разі потреби може змінити вказані дані, та надсилає її. На сервері записуються до бази даних та опрацьовуються дані заявки. Компанія обробляє заявку та за допомогою функцій серверу надсилає користувачу відповідь.

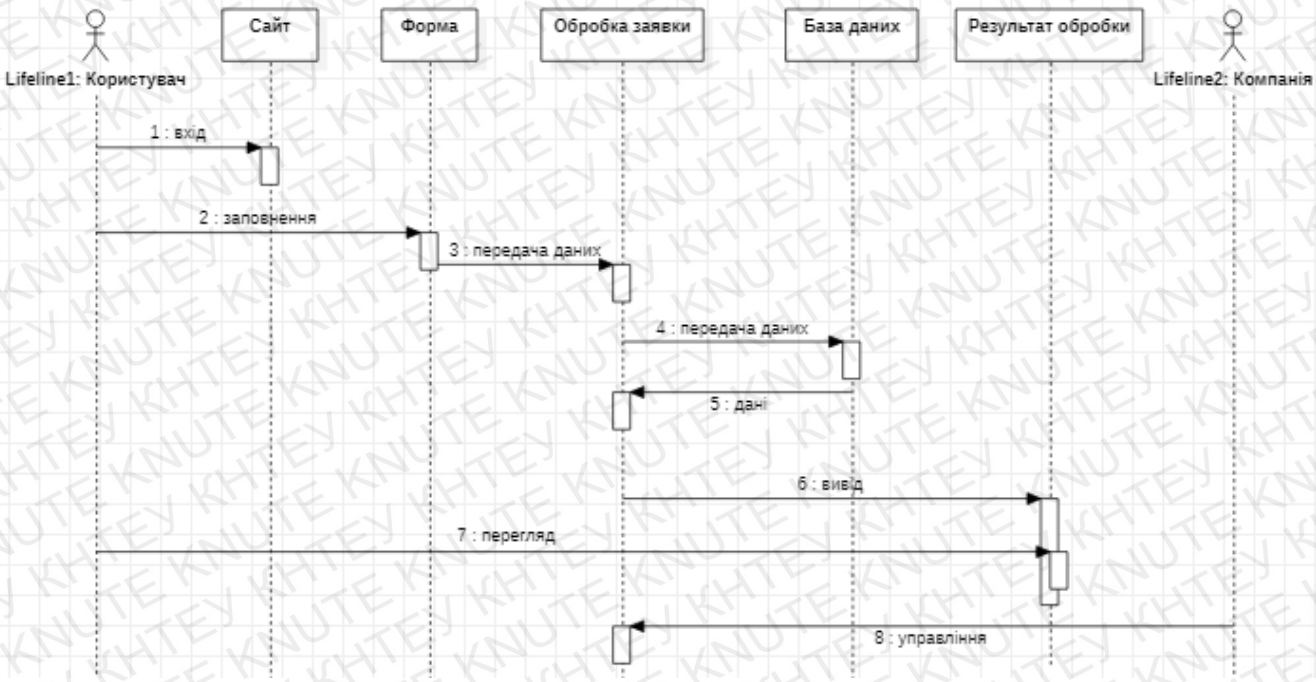

Рис. 3.2. Діаграма послідовності роботи користувача з веб-додатком

3.3. Програмна реалізація системи обробки електронних заяв

В створеному проекті Django необхідно підключити потрібні стандартні та створені модулі:

- Модуль адміністрування
- Модуль аутентифікації
- Модуль статичних файлів
- Модуль типів даних
- Модуль повідомлень
- Модуль подачі заяв
- Модуль обробки заяв

Модуль адміністрування – містить в собі налаштування для управління веб-додатком, керування приорітетами розподілу прав, доступом до бази даних, тощо.

Модуль аутентифікації – веб-додаток забезпечує доступ до різних елементів залежно від рівня доступу конкретного користувача. Звичайні користувачі можуть лише заповнювати заявки та відповідно їх надсилати. Адміністратори мають можливості обробляти заявки, а головний адміністратор може здійснювати розподіл прав доступу між користувачами зі сторони компанії.

Модуль статичних файлів – відповідає за роботу з підключеними статичними файлами, в тому числі з html-файлами, необхідними для відображення вмісту сторінок..

Модуль типів даних – відслідковує всі підключені бази даних, та налаштовує роботу з ними.

Модуль повідомлень – необхідний для коректної роботи з формами.

Модуль обробки заяв – надає функції для відображення заяв клієнтів та роботи з ними. Для відображення сторінки з поданими заявками побудована база даних яка міститиме їх. Структура БД наступна:

- Ім’я (name)
- Прізвище (prizv)
- Ім’я по-батькові (pobatkovy)
- Номер мобільного телефону (mobnomer)
- Електронна скринька (email)
- Дата й час прийому (datepr)
- Обраний напрям (likar)
- Коментар (comment)
- Дата й час подачі заявки (timepodachi)
- Результат обробки (rezobrobky).

```
name = models.CharField(max_length=15, help_text="ім'я")
prizv = models.CharField(max_length=50, help_text="прізвище")
pobatkovy = models.CharField(max_length=50, help_text="ім'я по батькові")
mobnomer = models.CharField(max_length=12, help_text="номер мобільного телефону")
email = models.EmailField(max_length=80, help_text="адреса електронної поштової скриньки")
datepr = models.DateTimeField(auto_now=False, auto_now_add=False)
likar = models.CharField(max_length=50, choices=LIKARI, default=SIML)
comment = models.TextField(default="")
timepodachi = models.DateTimeField()
rezobrobky= models.CharField(max_length=50,choices=OBROBKA, default=NEOBROB)
```

Рис.3.3. Налаштування бази даних

Також в модулі містяться файли для відображення списку поданих заяв та сторінки кожної заявки, реалізовано в файлах zs.html і z.html. За допомогою бібліотеки smtplib реалізовано зворотній зв’язок з клієнтом після обробки заявки.

Модуль подачі заяв – відображає форму для клієнтів, та відповідає за запис поданих заяв до бази даних. Сторінка форми зберігається в файлі prz.html, логіка в forms.py.

3.4. Технологія використання програмної реалізації системи обробки електронних заяв

Користувачі створеного веб-додатку поділяються на дві групи:

- Клієнти
- Адміністратори

Для кожної групи створено власний інтерфейс аби забезпечити всіх необхідними функціями.

Почнемо розгляд з точки зору клієнта. Спочатку необхідно зайти на сайт компанії та натиснути кнопку «Подати заявку» (рис. 3.4).

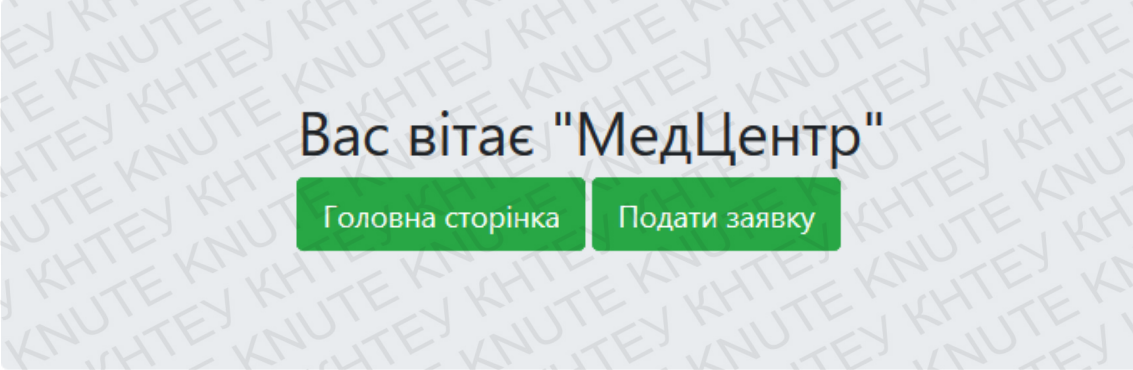

Рис.3.4. Головна сторінка

Після чого клієнт потрапляє на сторінку з формою (рис.3.5). Аби надіслати заявку форму потрібно заповнити і натиснути кнопку «Надіслати». У разі пропуску необхідних полів та при введенні некоректних даних, заявка не буде надіслана, а на веб-сторінці буде відображено помилки біля відповідних полів форми (рис.3.6).

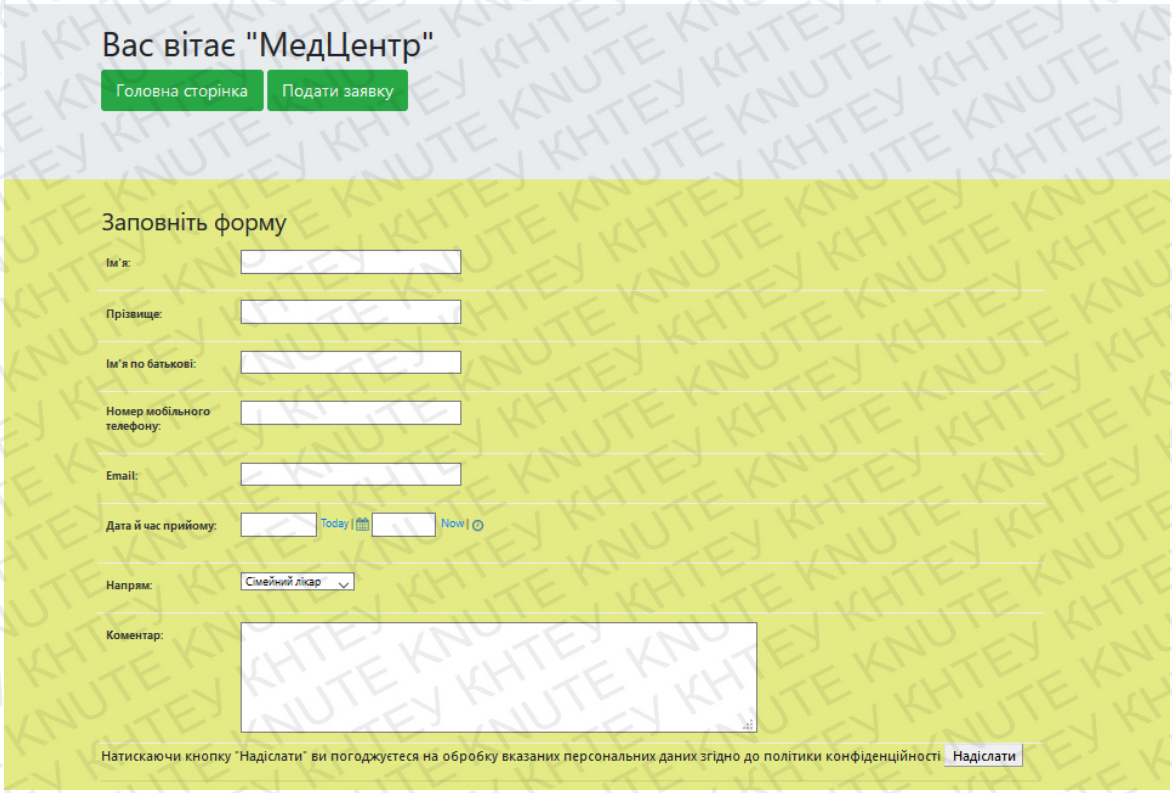

Рис.3.5. Форма подачі заявки

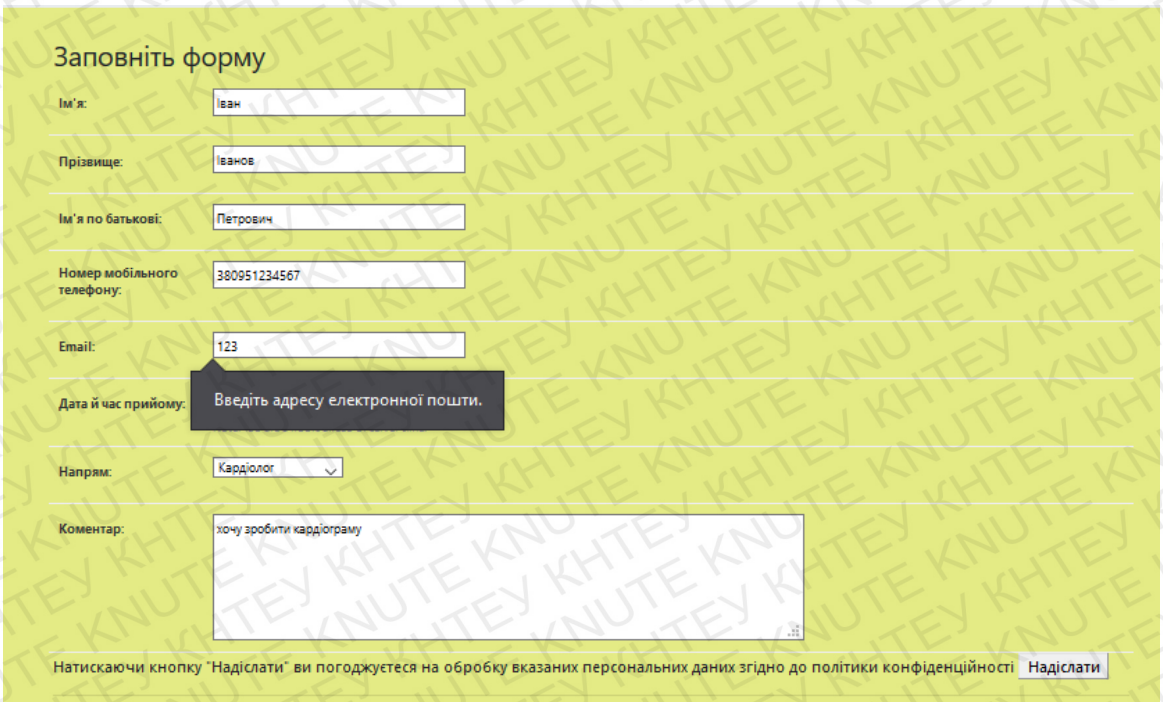

Рис.3.6. Форма з некоректно поданими даними

Після коректного заповнення і надсилання електронної заявки клієнт має зачекати її розгляду. Також клієнт, у разі необхідності, має можливість надіслати ще заявки аби зареєструватись на інші послуги.

Тепер розглянемо інтерфейс для роботи адміністратора в системі обробки електронних заяв.

Спочатку адміністратор має перейти за спеціальним посиланням де відразу з'явиться сторінка форми для входу в систему управління (рис.3.7).

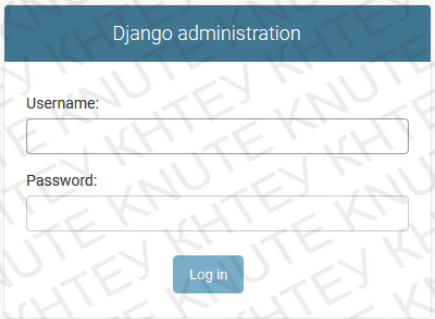

Рис.3.7. Форма входу в систему управління

Після успішної аутентифікації адміністратор потрапить на сторінку з усіма поданими заявами, також на цій сторінці є кнопки для швидкого доступу на головну сторінку сайту та панель адміністратора (рис.3.8).

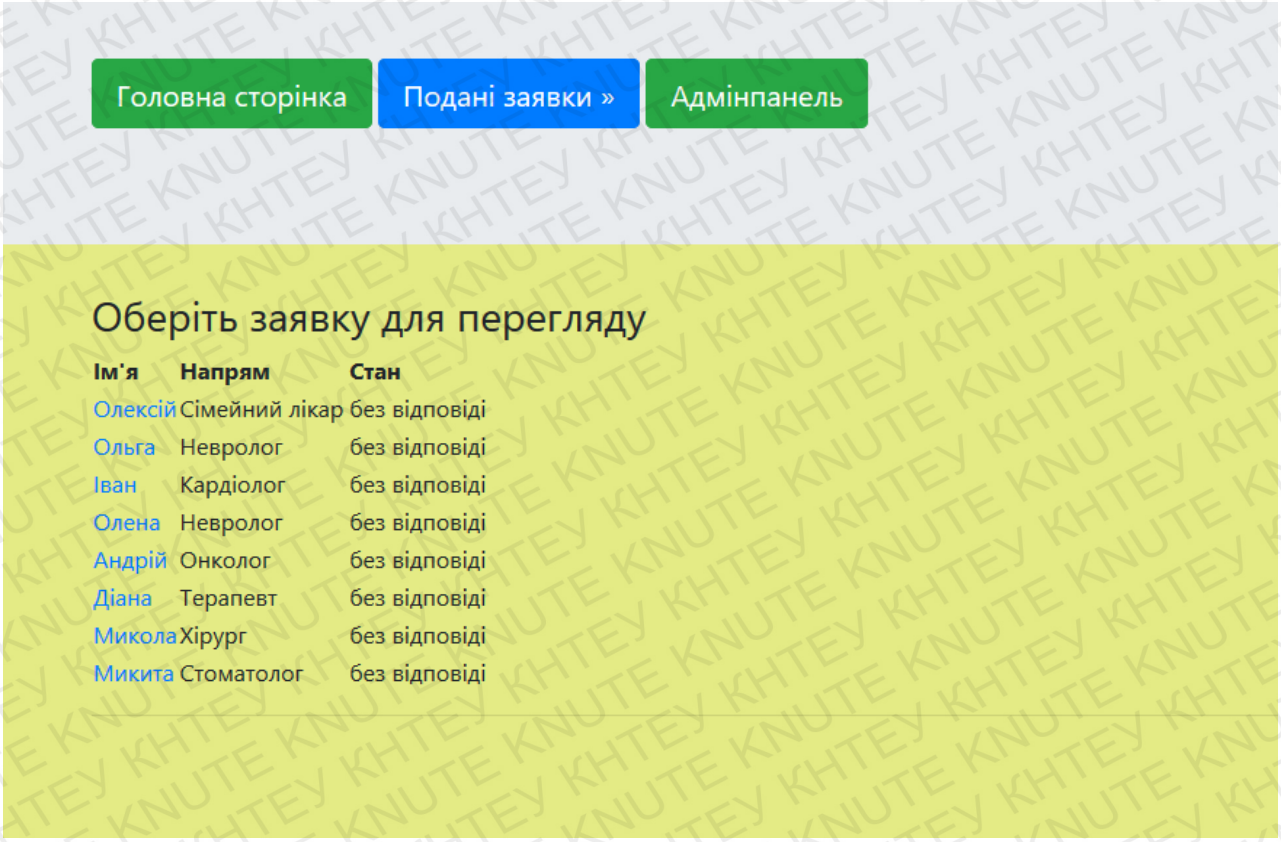

Рис.3.8. Сторінка поданих заяв

Щоб переглянути будь-яку заявку необхідно натиснути на ім’я клієнта, який подав її. Відкриється сторінка вибраної заявки, на якій відображено всю інформацію поданої заявки та є можливість прийняти чи відхили заявку (рис. 3.9). Після збереження заявки клієнту на електронну пошту надсилається лист з результатом обробки.

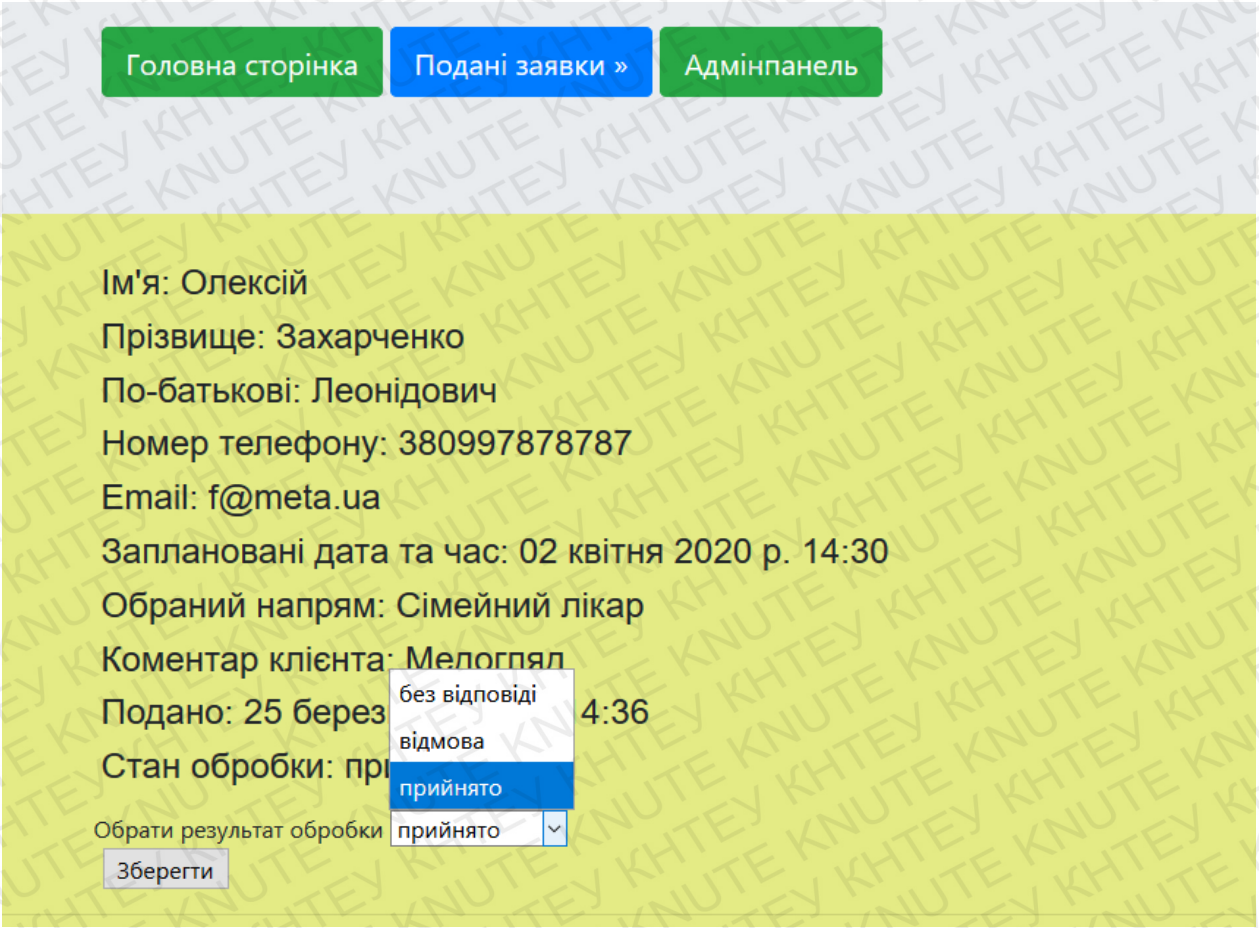

Рис.3.9. Сторінка заявки

Після обробки заявки автоматично сортуються так, що всі необроблені розміщені зверху, що покращує комфорт роботи з системою (рис. 3.10).

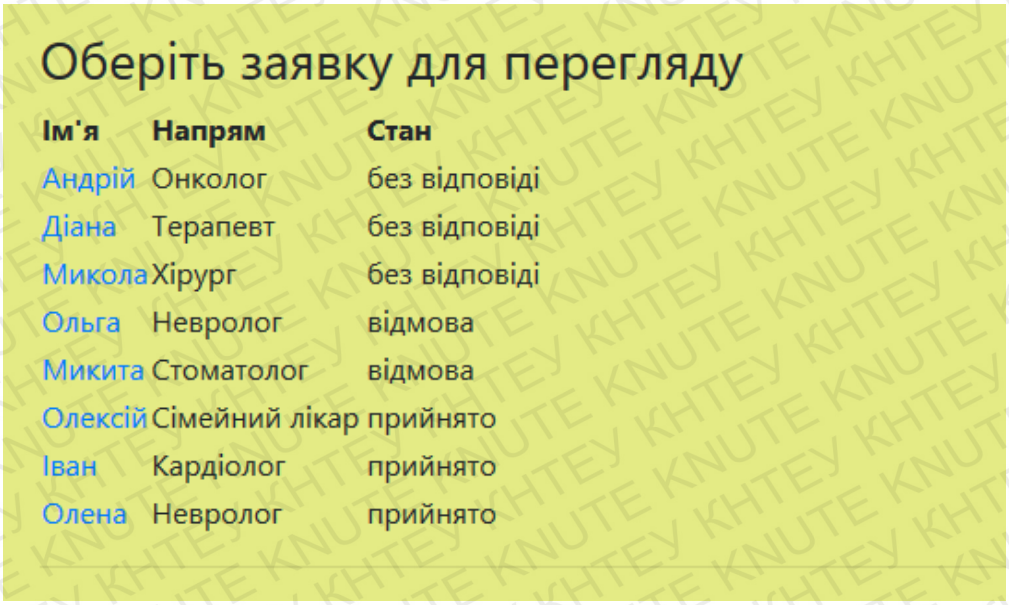

Рис.3.10. Сторінка поданих заяв після обробки

Система обробки електронних заяв розроблена і реалізована за допомогою сучасних програмних засобів система обробки електронних заяв, проста у використанні, має зрозумілий інтерфейс, здійснює розподіл рівнів доступу користувачів, не потребує спеціалізованої підготовки користувачів та значних витрат на її підключення до веб-сайту, може легко модифікуватися в залежності від напрямку і завдань обробки заяв.

ВИСНОВКИ

У випускному кваліфікаційному проекті представлено результати теоретичних і прикладних досліджень, що полягають у розробці веб-додатку − системи обробки електронних заяв, з метою підвищення ефективності роботи з клієнтами медичного центру. В результаті проведених досліджень були отримані такі висновки:

1. Електронні заявки є ефективним способом комунікації з клієнтами в сучасному суспільстві. Інформація, яку вони надають, цілісна та конструктивна, що позитивно впливає на роботу медичного закладу.

2. При розробці програмного забезпечення дуже важливо використовувати сучасні інформаційні технології, які є швидкими в роботі та захищеними.

3. Проаналізовано різні види баз даних та систем управління ними. Визначено, що реляційна БД SQLite повністю задовольняє потребам розробленої системи.

4. Проаналізовано програмне забезпечення для побудови веб-додатків. Визначено, що фреймворк Django надає все необхідне технічне забезпечення.

5. Проведено моделювання роботи системи обробки електронних заяв, її взаємодія з клієнтами. Побудовано діаграми послідовності та прецедентів при роботі з клієнтами.

6. При розробці веб-додатку використано сучасні технології, це дозволило побудувати оптимізовану веб-технологію. Це дозволило створити інтерфейс, який адаптується до пристрою, на якому відкритий, та є комфортним у використанні як для клієнтів, так і для адміністраторів.

7. Розроблено і реалізовано за допомогою сучасних програмних засобів веб-додаток систему обробки електронних заяв, який можна вбудувати до

веб-сайту. Він дозволяє приймати електронні заявки, переглядати та надсилати відповідь після обробки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Онлайн конструктор веб-форм FormDesigner. [Електронний ресурс]. – Режим доступу: https://formdesigner.ru/uk

2. Google Forms. [Електронний ресурс]. – Режим доступу: https://google.com/forms

3. Бітрікс24. [Електронний ресурс]. – Режим доступу: https://www.bitrix24.ua/

4. Form and Survey Builder StepForm. [Електронний ресурс]. – Режим доступу: https://stepform.io

5. Основи законодавства України про охорону здоров'я. [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/2801-12

6. Relational database [Електронний ресурс]. – Режим доступу: https://www.oracle.com/database/what-is-a-relational-database/

7. Швець М.Ю., Заруба Д.С., Хохлов Ю.В. Порівняння SQL та NoSQL баз даних, 2018 [Електронний ресурс]. – Режим доступу: http://www.tech.vernadskyjournals.in.ua/journals/2018/6_2018/part_2/7.pdf

8. Pros and Cons of Non-Relation Databases, 2016 [Електронний ресурс]. – Режим доступу: https://veesp.com/en/blog/sql-or-nosql

9. Шаров С.В., Петровський В.В. Огляд нереляційних баз даних [Електронний ресурс]. – Режим доступу: http://eprints.mdpu.org.ua/id/eprint/602/1/шаров_петровский.pdf

10. Сучасні методи веб-програмування [Електронний ресурс]. – Режим доступу: http://sites.znu.edu.ua/webprog/lect/1222.ukr.html

11. About SQLite [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/about.html

12. Postgresql [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/

13. Microsoft SQL Server [Електронний ресурс]. – Режим доступу: https://www.microsoft.com/en-us/sql-server/

14. About Firebird [Електронний ресурс]. – Режим доступу: https://firebirdsql.org/en/about-firebird/

15. MongoDB [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/

16. Семенюк Р.А. Переваги та недоліки використання СУБД MongoDB [Електронний ресурс]. – Режим доступу: http://eprints.zu.edu.ua/24821/1/Semenyuk_APSI2017.pdf

17. Apache CouchDB Documentation. [Електронний ресурс]. – Режим доступу: https://docs.couchdb.org/en/stable/

18. Multi-model hightly available NoSQL database – ArangoDB. [Електронний ресурс]. – Режим доступу: https://www.arangodb.com/

19. HTML: Hypertext Markup Language. [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/uk/docs/Web/HTML

20. CSS-CSS3. [Електронний ресурс]. – Режим доступу: https://html5book.ru/css-css3/

21. CSS Tutorial [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/css/

22. Методы верстки сайтов [Електронний ресурс]. – Режим доступу: https://webformyself.com/metody-verstki-sajtov/

23. Django веб-фреймворк (Python) [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/uk/docs/Learn/Server-side/Django/Introduction

24. Django documentation and libraries for Django [Електронний ресурс]. – Режим доступу: https://django.fun/

25. Django documentation [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com/en/3.0/

26. Про Node.js [Електронний ресурс]. – Режим доступу: https://nodejs.org/uk/about/

27. Laravel The PHP Framework for Web Artisans [Електронний ресурс]. – Режим доступу: https://laravel.com/

28. Laravel - Overview [Електронний ресурс]. – Режим доступу: https://www.tutorialspoint.com/laravel/laravel_overview.htm

29. jQuery API [Електронний ресурс]. – Режим доступу: https://api.jquery.com/

30. Ruby on Rails Guides [Електронний ресурс]. – Режим доступу: https://guides.rubyonrails.org/

31. Фреймворк Ruby on Rails [Електронний ресурс]. – Режим доступу: https://web-creator.ru/articles/about_ruby_on_rails

32. Spring [Електронний ресурс]. – Режим доступу: https://spring.io/