Міністерство освіти і науки України
Одеський національний технологічний університет
Кафедра комп'ютерної інженерії

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему *Розробка веб-сайту для онлайн бронювання зустрічей*
(назва кваліфікаційної роботи згідно наказу ОНТУ)

Здобувача *Артюхова М.В.*
(прізвище, ініціали)

4 курсу *543б* групи

Керівник: *к.т.н., доц. Нєнов О.Л.*
(посада, прізвище та ініціали)

ст.викл. Слушна Н.В.
(посада, прізвище та ініціали)

Консультанти: _____
(посада, прізвище та ініціали)

Phd, ст.викл. Богданов О.О.
(посада, прізвище та ініціали)

Кваліфікаційна робота допускається до захисту
Рішення кафедри від *05.06.* 2024 р., протокол № *8*

Завідувач кафедри комп. інженерії _____ Сергій АРТЕМЕНКО
(назва кафедри) (підпис) (Ім'я ПРІЗВИЩЕ)

Одеса - 2024 рік

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерної інженерії, програмування та кіберзахисту*
Кафедра *комп'ютерної інженерії*
Ступінь вищої освіти *бакалавр*
Спеціальність *123 «Комп'ютерна інженерія»*
Освітня програма *Розробка ігор та інтерактивних медіа у віртуальній реальності*

ЗАТВЕРДЖУЮ

Зав. кафедри *комп'ютерної інженерії*
Сергій АРТЕМЕНКО
« *30* » *серпня* 2023 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Артюхова Максима Володимировича

1. Тема роботи *Розробка веб-сайту для онлайн бронювання зустрічей*

Затверджена наказом університету від « *30* » *серпня* 2023р., наказ № *442-03*

2 Термін здачі здобувачем закінченої роботи *28 травня 2024 р.*

3. Вихідні дані роботи
Стандарт взаємодії сервісів ASP.NET Core Framework 4.5. 2. Технологія доступу к даним ASP.NET Identity. 3. СУБД MongoDB. 4. Середовище розробки Rider. 5. Технології розробки Angular. 6. Сервер застосувань Node.Js.

4. Перелік питань, які потрібно розробити
1. Вступ. 2. Аналіз предметної області. 3. Проектування WEB-системи. 4. Програмна реалізація веб-сайту та бази даних 5. Економічні розрахунки. 6. Охорона праці. 7. Загальні висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайд 1. Тема роботи. Слайд 2. Актуальність роботи. Слайд 3-4. Завдання Слайд 5. Аналіз існуючих аналогів. Слайд 6. Технології. Слайд 7. Архітектура Слайд 8-10. Проектування системи Слайд 11. Проектування структури та інтерфейсу. Слайд 12.База даних. Слайд 13. Реалізація продукту. Слайд14. Економічні показники Слайд15. Демонстраційне відео

6. Консультанти по роботі, із зазначенням розділів роботи, що стосуються їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Економіка	*Phd, ст.викл. Богданов О.О.*		
Охорона праці	*ст. викладач Слушна Н.В.*		
Нормоконтроль	*ст. викладач Слушна Н.В.*		

7. Дата видачі завдання *30.08.2023*

Керівники _____ *Наталя СЛУШНА*

_____ *Олексій НЄНОВ*

Завдання прийняв до виконання _____ *Максим АРТЮХОВ*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	*Дослідженні предметної області*	*30.01.2024*	
2.	*Дослідження існуючих аналогів*	*10.03.2024*	
3.	*Проектування структури сайту.*	*20.03.2024*	
4.	*Програмна реалізація сайту.*	*01.04.2024*	
5.	*Наповнення контентом. Тестування.*	*15.05.2024*	
6.	*Підготовка техніко-економічної частини*	*21.05.2024*	
7.	*Підготовка розділу охорони праці*	*25.05.2024*	
8.	*Оформлення пояснювальної записки.*	*31.05.2024*	
9.	*Оформлення графічної частини та лістингу*	*08.06.2024*	

Здобувач-дипломник_____ *Максим АРТЮХОВ*

Керівники роботи _____ *Наталя СЛУШНА*

_____ *Олексій НЄНОВ*

Несу відповідальність за ідентичність електронного та друкованого варіантів кваліфікаційної роботи, даю згоду на обробку персональних даних та не заперечую проти розміщення кваліфікаційної роботи на офіційних web-ресурсах ОНТУ.

Підтверджую, що в кваліфікаційній роботі відсутні порушення норм академічної доброчесності.

Здобувач-дипломник *Максим АРТЮХОВ* _____

АНОТАЦІЯ

Дипломна робота виконувалась за темою "Розробка веб-сайту для онлайн бронювання зустрічей". Ця робота є актуальною, тому що сьогодні великим попитом користуються сайти бронювання зустічей та заходів. Технічно підковані люди використовують такі веб-сайти, щоб заздалегідь забронювати місця для заходів та з комфортом керувати сімейними, соціальними, політичними та професійними справами.

В рамках роботи в першому розділі проведений аналіз предметної області застосування, описана актуальність задачі, визначені наявні проблеми, підготовлено план роботи для реалізації проєкту. В другому розділі описані основні етапи проектування: визначення необхідних деталей виконання бізнес-процесів, визначення об'єктів і сервісів системи, проектування бази даних. У третьому розділі розглянуто процес розробки сайту на основі сформованого технічного завдання та проведеного проектування, надаються результати тестування. Четвертий розділ обґрунтовує економічну вигоду від застосування одержаної системи. У п'ятому розділі розглянуто питання охорони праці, де описані основні принципи та правила поводження з комп’ютерною технікою.

В результаті було розроблено універсальний сайт для онлайн бронювання зустрічей з простим і зрозумілим у використанні інтерфейсом, призначенням якого може бути, наприклад, пошук людей для спільного проводження тренингів. Також було проведено тестування роботи веб-сервісу.

Результати цієї роботи можуть бути корисними для малого бізнеса. Веб рішення дозволяють клієнтам бронювати та планувати зустрічі онлайн, не вимагаючи інсталяції. Такі системи є гнучкими та зручними в управлінні, забезпечуючи доступ та управління в будь-який момент.

Ключові слова: бронювання, реєстрація, *MongoDB, C#, TypeScript, ASP.NET Core, Angular.*

ANNOTATION

The thesis was focused on the topic "Development of a website for online booking of students." This work is relevant, because today there is a great demand for armoring sites and entries. Tech-savvy people use such websites in order to book a place to visit in advance and comfortably handle family, social, political and professional rights.

As part of the work, in the first section of the analysis of the subject area of the problem, the relevance of the task is described, obvious problems are identified, and a work plan is prepared for the implementation of the project. Another section describes the main stages of design: identification of necessary parts, identification of business processes, identification of objects and system services, design of a database. The third section examines the process of developing the site on the basis of the formed technical specifications and the design carried out, and provides the results of testing. The fourth section is about the economic benefit from the stagnation of the acquired system. The fifth section examines nutritional defense and describes the basic principles and rules of using computer technology.

As a result, a universal website for online booking of travel agents was created with a simple and intuitive interface, suitable for people who could, for example, search for a successful appointment. The robotic web service was also tested.

The results of this work may be detrimental to small businesses. Web solutions allow clients to book and schedule appointments online without requiring installation. Such systems are flexible and easy to manage, ensuring access and control at any time.

Key words: *reservation, registration, MongoDB, C#, TypeScript, ASP.NET Core, Angular.*

ЗМІСТ

ВСТУП

Життя людини настільки комфортне, наскільки ефективно вона може керувати своїм часом. Нерідко час, який людина могла би витратити на навчання, роботу чи сім'ю, доводиться витрачати на рішення нецільових задач.

Онлайн бронювання зустрічей – одна із нових сучасних можливостей сайтів. Люди використовують веб-сайти, щоб бронювати готелі, ресторани, громадські зали тощо, щоб легко організувати різні типи заходів.

Кожна людина хоч раз призначала зустріч або записувалася на прийом до якогось фахівця. Раніше це можна було зробити по телефону або через месенджер і, щоб не забути про зустріч, поставити нагадування. Планування зустрічей вручну може стати завданням, яке не тільки вимагає значних зусиль, але і супроводжується ризиком помилок, таких як подвійні бронювання та, як наслідок, незадоволені клієнти. Вся справа в засобах планування, які звикли використовувати люди. Класичні календарі, дзвінки, повідомлення та листування електронною поштою застаріли, і їх давно настав час замінити на онлайн-планувальники зустрічей.

Сервіси бронювання зустрічей дозволяють уникнути нераціонального використання часу. Достатньо скинути своєму співрозмовнику посилання, де будуть вказані доступні часові рамки. Також ці сервіси нагадують про майбутню зустріч, надсилають листи всім учасникам.

Використання сервісу онлайн-запису може дозволити ефективніше планувати навантаження, уникати помилок у розкладі й підвищить точність та ефективність роботи, що буде сприяти поліпшенню якості обслуговування і рівня задоволеності клієнтів. Відповідно з розумінням ефективності використання систем онлайн бронювання зустрічей зростає попит на подібні системи в останні часи.

Саме тому обрана тема дипломного проекта – розробка веб-сайту для онлайн бронювання зустрічей – буде актуальною та значущою.

Найвідомішими представниками систем онлайн-бронювання зустрічей є *Calendly*, *Bookafy*, *ScheduleOnce*, *SimplyBook* та інші.

Зазначимо мету, об’єкт, предмет та задачі, які необхідно вирішити в ході даної роботи.

Метою роботи є проектування і розробка веб-сайту для онлайн бронювання зустрічей для організації, яка займається соціалізацією молоді.

Об’єктом дослідження є процеси проектування і розробки веб-сайту для онлайн бронювання різного плану зустрічей (тренингів, мастер-класів, учбових занять).

Предметом дослідження є методи проектування і розробки веб-сайту для онлайн бронювання.

Методи дослідження. В ході виконання роботи можуть бути використані такі методи як аналіз, абстрагування, узагальнення, аналогія, класифікація.

Основними задачами, які необхідно вирішити в ході роботи, є:

1. Аналіз предметної області, дослідження існуючих аналогів та постановка задачі.

2. Проектування бази даних та системи бронювання.

3. Розробка демонстраційної версії сайту онлайн бронювання зустрічей.

Докладно для реалізації такого веб-сайту необхідно вирішити наступні завдання:

- вивчити предметну область та існуючі аналоги;
- вивчити архітектуру веб-сайтів та методи їх розробки;
- обрати технології для проектування та розробки;
- розробити вимоги до веб-сайту онлайн бронювання;
- розробити *UML*-моделі предметної галузі;
- реалізувати базу даних та веб-сайт;
- протестувати демо-версію сайту на помилки.

Веб-сайт повинен виконувати такі функції:

- забезпечувати реєстрацію клієнтів – співробітників організації та потенційних учасників зустрічей;

– надавати інформацію для потенційних учасників зустрічей про зустрічі, спектр послуг, що надаються;

– містити форму для онлайн запису на зустріч.

Новизна дипломної роботи: використання нереляційної бази даних *MongoDB* та сучасної системи *Angular framework*, яка надає можливість зручного кодування та спрощеного тестування.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ АНАЛОГІВ

1.1 Інформаційний огляд предметної області

Системи бронювання — це програмне забезпечення для планування зустрічей, які дозволяють підприємствам та приватним особам управляти зустрічами та бронюваннями. Часто такі сервіси мають можливість синхронізації з різними календарями, що дозволяє не дублювати записи про зустріч у кілька різних програм, а все зібрати в одному місці.

Умовно можна розділити ці сервіси на два види - бронювання зустрічей та бронювання послуг. Такий розподіл умовний і буде запропонований для простоти розуміння.

Система онлайн бронювання послуг — це цифровий інструмент, який дозволяє клієнтам бронювати послуги або продукти онлайн. Під бронюванням послуг мається на увазі процес, коли клієнт на сайті або в додатку за потрібною йому послугою бачить вільні слоти та бронює їх самостійно.

Система онлайн бронювання зустрічей — це цифровий інструмент, який дозволяє клієнтам системи бронювати зустрічі онлайн. Умовно кажучи, бронювання зустрічі — це коли представник компанії позначає свій доступний час і надає клієнту можливість вибрати з цих варіантів те, що йому підходить.

Основні переваги використання програмного забезпечення для онлайн бронювання включають цілодобову доступність для клієнтів, спрощене керування бронюванням, автоматичне планування та нагадування, а також покращений досвід роботи з клієнтами.

З урахуванням зростання популярності різноманітних допоміжних систем, важливо бути в курсі нових технологій та використовувати їх для покращення власного бізнесу й задоволення потреб сучасного клієнта.

На що слід звернути увагу при виборі програми онлайн-бронювання? Потрібно враховати сферу бізнесу та цільову галузь, так вдасться вибрати програмне забезпечення з необхідними функціями.

Як переконатися, що система планування зустрічей відповідає поставленим цілям? Необхідно почати з використання тестових версій програм для бронювання, щоб вивчити їхні функції та визначити, чи відповідають вони необхідним потребам.

Чи потрібен бізнесу функціонал онлайн-бронювання? Якщо компанія надає послуги й регулярно планує зустрічі з клієнтами, впровадження системи бронювання принесе вигоду і підвищить ефективність.

З урахуванням цих переваг використання програмного забезпечення для планування зустрічей, може стати важливим інструментом для оптимізації роботи компанії, яка надає послуги.

Клієнти можуть активно використовувати онлайн-ресурси для бронювання зустрічей, і ось основні шляхи, якими вони знаходять сторінки онлайн-запису[1]:

1. Соціальні мережі (37%). Більшість користувачів знаходять сервіси для планування через соціальні мережі, де вони можуть відгукнутися на рекомендації та рецензії.
2. Веб реферали (31%). Клієнти також звертають увагу на рекомендації та посилання в Інтернеті, які вказують на сервіси для бронювання.
3. Прямий ввід *URL* (28%). Деякі відвідувачі знаходять сторінки онлайн-бронювання напряму, можливо, через рекламу або прямий ввід *URL*.
4. Органічний пошук (3.5%). Гості використовують пошукові системи для знаходження відповідних сервісів.
5. Платний пошук (0.5%). Частка клієнтів, які знаходять сторінки через платні рекламні кампанії, менша, але все ще значуща.

Щодо функцій, які малі бізнеси шукають у сервісах для планування зустрічей[1]:

1. Управління календарем (36%). Важлива функція для ефективного планування та управління часом.
2. Автоматичні сповіщення (21%). Компанії цінують можливість автоматизованого сповіщення для підтримання зв'язку з клієнтами.
3. Планування ресурсів (20%). Функція, що допомагає ефективно розподіляти та використовувати ресурси.

Клієнти в основному планують свої бізнес-зустрічі онлайн через смартфони, надаючи перевагу самостійності та зручності у керуванні процесом бронювання. Саме тому оптимізація сервісу під мобільні пристрої надзвичайно важлива.

Факти, які відзначають переваги та позитивний вплив онлайн-бронювання на різні галузі бізнесу:

1. Гнучкість для клієнтів. Онлайн-бронювання дозволяє клієнтам планувати прийоми, навіть поза робочим часом, підвищуючи доступність послуг та обслуговуючи ширший коло клієнтів.

2. Збільшення числа нових клієнтів. Онлайн-запис сприяє росту клієнтури через простоту та комфорт бронювання.

3. За даними аналітики, впровадження програмного забезпечення для планування зустрічей приводить до зростання доходу компаній на 30-45%, а 59% клієнтів невдоволені традиційним телефонним бронюванням.

4. Переваги для сфери медицини. Зручність для пацієнтів. Онлайн-записи на послугу підвищують зручність для пацієнтів, і 35% клієнтів вважають це вирішальним при виборі медичного провайдера.

5. Збільшення кількості записів. Лікарі, які пропонують онлайн-записи, мають на 24% більше записів, ніж ті, хто пропонує лише телефонний запис.

6. Ефективність в управлінні клієнтами. Підключення системи нагадувань про зустрічі дозволяє скоротити кількість неявок на 33% і збільшити кількість відвідувань на 94,5% [1].

Ці факти відзначають переваги та вплив онлайн-планування на різні галузі, підкреслюючи його значущість для покращення обслуговування та

ефективності бізнесу. І таким чином показують актуальність розробки системи онлайн-бронювання зустрічей.

Класифікація програмного забезпечення для онлайн бронювання.

Існують різні типи програмного забезпечення для онлайн бронювання зустрічей, і кожен з них має свої унікальні характеристики та переваги:

1. Стаціонарні програми. Ці програми ліцензуються та встановлюються безпосередньо на комп'ютер користувача. Вони відзначаються широким функціоналом і можливостями кастомізації. Однак їх обмеженням є відсутність онлайн-порталу для доступу співробітників та клієнтів, а також потреба в регулярних оновленнях та обслуговуванні.

2. Веб системи. Цей тип програмного забезпечення надається як хостингове рішення сторонніми постачальниками. Вони дозволяють клієнтам бронювати та планувати зустрічі онлайн, не вимагаючи інсталяції. Веб системи є гнучкими та зручними в управлінні, забезпечуючи доступ та управління в будь-який момент.

3. Сервіси управління клієнтами. Це спеціалізоване програмне забезпечення призначене для великих мобільних робочих груп. Сервіс автоматизує планування та маршрутизацію зустрічей з клієнтами на місці. Використовуючи алгоритми та аналіз робочих моделей, Така система точно прогнозує час прибуття, скорочуючи час очікування клієнтів.

Ці типи сервісів онлайн-запису дозволяють компаніям вибирати рішення, яке найкраще відповідає їхнім конкретним потребам. Також важливо враховувати специфіку галузей, які отримують вигоду від систем на основі зустрічей, таких як охорона здоров'я, краса, технічне обслуговування та розваги.

Класифікація веб-сайтів та їх можливості для вирішення задачі

У наші часи, щоб сприяти стійкому успіху, потрібно постійно залучати нових клієнтів для розвитку і не стояти на місці – необхідно заводити нові корисні знайомства, і один із найкращих способів зробити це – заявити про себе

в мережі, наприклад на основі веб-сайту. Причин для використання чи створення сайтів – безліч. Призначення веб-сайта залежить від потреб власника веб-сайта. Він може мати багатогранні цілі з різними завданнями веб-сайта, або він може мати одну єдину мету.

Є можливість показати потенційним клієнтам, що вони отримають, працюючи з компанією, показуючи високоякісні фотографії на своєму веб-сайті чи просто надавати деяку інформацію кінцевому користувачеві.

Веб-сайт – це сукупність багатьох веб-сторінок, а веб-сторінки – це цифрові файли, написані за допомогою *HTML* (мова гіпертекстової розмітки). І щоб веб-сайт був доступним кожній людині у світі, він повинен цілодобово зберігатися або розміщуватися на веб-сервері.

Компонентами веб-сайта є:

Веб-хост – це послуга по розміщення веб-сайтів на спеціалізованих серверах і забезпечення доступу до них через мережу Інтернет. Компанії, що надають послуги веб-хостингу (хостинг-провайдери), виділяють місце на своєму власному або орендованому сервері і надають необхідну ширину інтернет-каналу.

URL-адреса веб-сайта – адреса, яку користувачеві потрібно ввести у веб-браузер, якщо він хоче відкрити веб-сайт, і веб-сервер доставляє запитуваний веб-сайт.

Домашня сторінка – є дуже поширеною та важливою частиною веб-сторінки. Це перша веб-сторінка, яка з’являється, коли відвідувач відкриває веб-сайт. Вона визначає зовнішній вигляд веб-сайта та спрямовує глядачів на інші сторінки веб-сайта.

Дизайн – це остаточний і загальний вигляд веб-сайта, який є результатом правильного використання та інтеграції елементів, таких як меню навігації, графіка, макет, навігаційні меню тощо.

Вміст – всі веб-сторінки, що містяться на веб-сайті, разом складають його вміст. Хороший вміст на веб-сторінках робить веб-сайт більш ефективним та привабливим.

Структура навігації – це порядок сторінок, набір посилань. Зазвичай він отримується принаймні одним навігаційним меню.

Після введення певної *URL*-адреси в рядок пошуку браузера, браузер запитує сторінку у веб-сервера, а веб-сервер повертає браузеру необхідну вебсторінку та її вміст. І сервер може повертати інформацію, у випадку статичних і динамічних веб-сайтів, порізному. На статичних веб-сайтах сервер повертає веб-сторінки, які є попередньо створеними файлами вихідного коду, створеними за допомогою простих мов, таких як *HTML, CSS* або *JavaScript*. Веб-сторінки повертаються сервером без змін, тому статичні веб-сайти швидкі. Відсутня взаємодія з базами даних. Крім того, вони менш дорогі, оскільки хост не потребує підтримки обробки на стороні сервера різними мовами.

У динамічних веб-сайтах веб-сторінки повертаються сервером, який обробляється під час виконання, означає, що вони не є попередньо створеними веб-сторінками, але вони створюються під час виконання відповідно до вимог користувача за допомогою мов сценаріїв на стороні сервера, таких як *PHP*, *Node.js, ASP.NET* та багато інших, які підтримує сервер. Таким чином, вони повільніші, ніж статичні веб-сайти, але у них можливі оновлення та взаємодія з базами даних. Динамічні веб-сайти частіше використовуються замість статичних веб-сайтів, оскільки оновлення можна виконувати дуже легко в порівнянні зі статичними веб-сайтами (де потрібно вносити зміни на кожній сторінці).

Найпоширеніші категорії веб-сайтів наведено нижче.

Блоги – цими типами веб-сайтів керує окрема особа або невелика група людей, вони можуть висвітлювати будь-які теми – вони можуть давати користувачу поради щодо моди, подорожей, фітнесу, музичні поради. Сьогодні професійне ведення блогів стало популярним способом заробітку в Інтернеті.

Електронна комерція – ці веб-сайти добре відомі як інтернет-магазини. Ці веб-сайти дозволяють нам здійснювати покупку продуктів, онлайн-платежі за продукти та послуги.

Портфоліо – такі типи веб-сайтів діють як розширення резюме фрілансера або компанії. Це надає потенційним клієнтам зручний спосіб перегляду виконані роботи, а також дозволяє їй розширити свої навички чи послуги.

Брошура – ці типи веб-сайтів в основному використовуються малими підприємствами, ці типи веб-сайтів діють як цифрові візитні картки та використовуються для відображення контактної інформації та реклами послуг лише на кількох сторінках.

Новини та журнали – ці веб-сайти потребують менше пояснень, головна мета цих типів веб-сайтів – інформувати своїх читачів про поточні події або розповідати про розваги.

Соціальні мережі – ці веб-сайти зазвичай створюються для того, щоб люди могли ділитися своїми думками, зображеннями, відео та іншими корисними компонентами. Є відомі веб-сайти соціальних мереж, такі як *Facebook, Twitter, Reddit* та багато інших.

Освітні веб-сайти – сайти призначені для відображення інформації за допомогою аудіо, відео чи зображень.

Портал – ці типи веб-сайтів використовуються для внутрішніх цілей у школі, інституті чи будь-якому підприємстві. Ці веб-сайти часто містять процес входу, що дозволяє студентам отримати доступ до їх облікової інформації або дозволяє співробітникам отримувати доступ до своїх електронних листів та сповіщень.

Отже, веб-сайт можна розглядати як цифрове середовище, здатне надавати інформацію та рішення та сприяти взаємодії між людьми, місцями та речами для підтримки цілей організації, для якої він був створений.

Схема класифікації веб-сторінок зображена на рис. 1.1 [1].

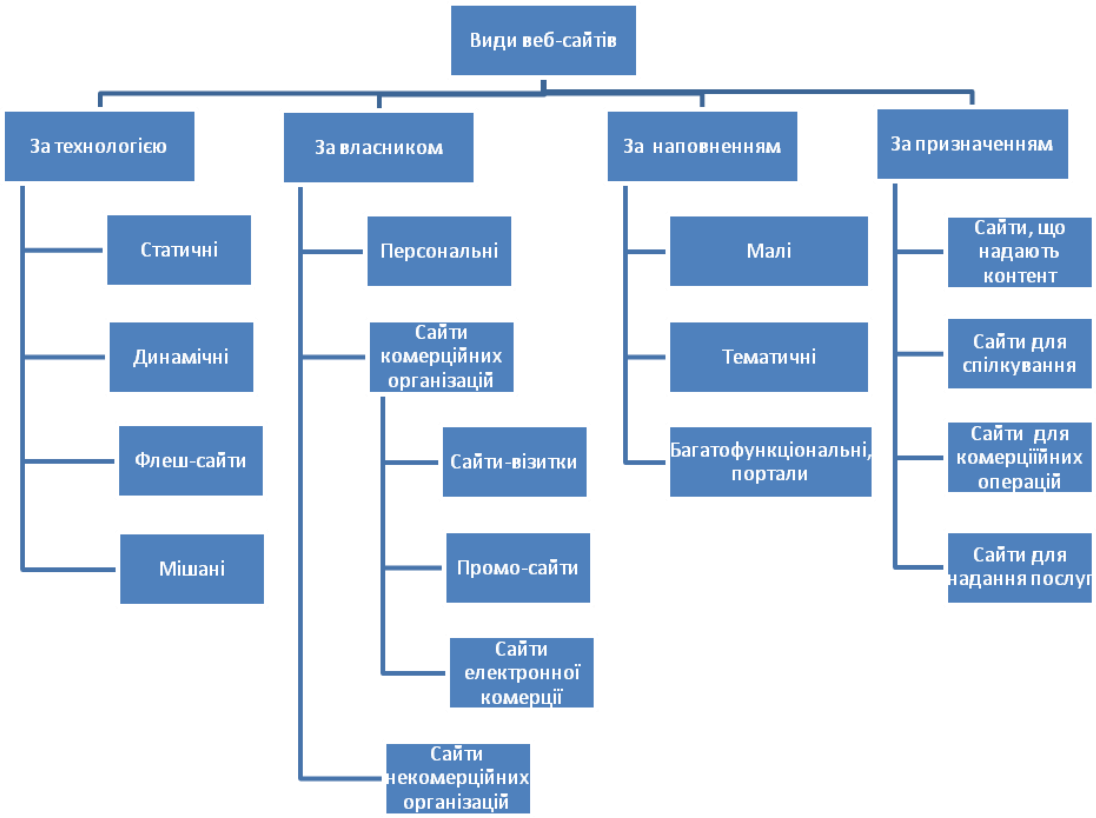


Рис. 1.1 – Класифікація веб-сторінок

Існує безліч платформ і мов, на яких може бути розроблений *web*-сервіс. Важливим моментом є актуальність, цікавість та корисність контенту користувачам.

1.2 Існуючі аналоги

Відмова від листування поштою та перехід на онлайн системи бронювання зустрічей сприятливо позначається на бізнесі, тому подібні послуги почали зростати (у кількісному відношенні) як гриби. Ринок онлайн-планувальників – висококонкурентне середовище. Зараз використовують програми *Calendly*, *Bookafy*, *EasyWeek*, *SimplyBook* та десятки їх аналогів.

Всі вони виконують однакові завдання, але відрізняються інтегрованими сервісами, деякими параметрами для календарів, а також мовами, що підтримуються, і вартістю самої програми.

В Україні є багато сервісів онлайн-запису з різним функціоналом. Розглянемо 6 найбільш поширених систем на українському ринку.

1.2.1 *Calendly*

Calendly – один із найпопулярніших сервісів. У *Calendly* інтуїтивно зрозумілий інтерфейс із симпатичним оформленням (рис.1.2), в яке можна легко вписати власний логотип, корпоративні кольори тощо.

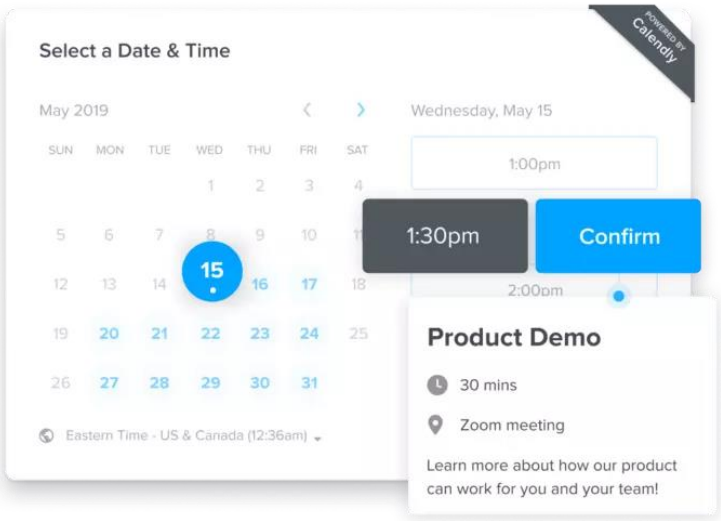


Рис. 1.2 – Інтерфейс програми *Calendly*

У *Calendly* є весь базовий набір опцій, типових для систем бронювання: обмеження кількості вільних часових осередків, оповіщення про нові записи поштою та push-сповіщення. Але найголовніше – величезна кількість інтеграцій із сторонніми службами та програмами, включаючи *Teams, Zoom, iCloud, Google Calendar, Slack, PayPal* та інші.

1.2.2 *EasyWeek*

EasyWeek надає онлайн-платформу, на якій клієнти компанії можуть переглянути доступні товари або послуги, забронювати час і здійснити оплату (рис.1.3). *EasyWeek* є комплексним рішенням, яке пропонує функції онлайн-бронювання і є ефективною заміною звичного запису клієнтів через *Google*-таблиці або паперовий зошит. Сервіс бронювання зручний для співробітників та власників компанії.

Вартість місячної підписки залежить від кількості локацій та співробітників. Для малого бізнесу (1 локація та 1 працівник) діє безкоштовний

базовий тариф. Починаючи з 2 користувачів діятиме тариф 21 євро або більше відповідно до кількості персоналу. Також бувають знижки та акції на річні підписки.

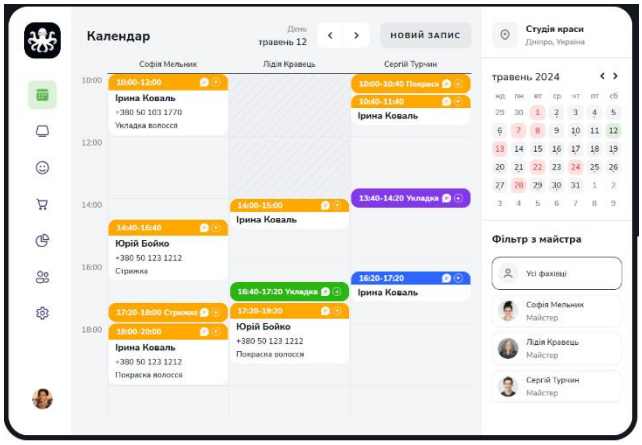


Рис. 1.3 – Інтерфейс планувальника зустрічей *EasyWeek*

1.2.3 *Altegio*

Altegio – ще один популярний сервіс онлайн-запису на українському ринку. *Altegio* надає базові можливості онлайн-запису, маркетингу та бізнес-аналітики. Це програмне забезпечення для онлайн-бронювання та планування зустрічей для підприємств, що надають різні види послуг (рис.1.4).

Включає наступні сервіси:

1. Бронювання онлайн. Можливість отримувати записи цілодобово та без вихідних через *URL*-адресу бронювання та безпечно отримувуати оплату онлайн.

2. Призначення зустрічей. Керування онлайн і офлайн зустрічами: планування, перенос або скасовування їх у кілька кліків. Перегляд власних зустрічей чи зустрічей співробітників.

3. Сповіщення та нагадування. За допомогою сервісу *SMS*-повідомлень та листів електронної пошти збільшується кількість зустрічей і виключаються запізнення. За допомогою функції сегментації клієнтів можна групувати клієнтів

(наприклад, за датою останнього відвідування) та надсилати їм спеціальні пропозиції, щоб повернути їх.

4. Клієнтська база даних. Можливість вводу та відстеження даних всіх клієнтів: історія зустрічей, відсутність виступів та статус членства.

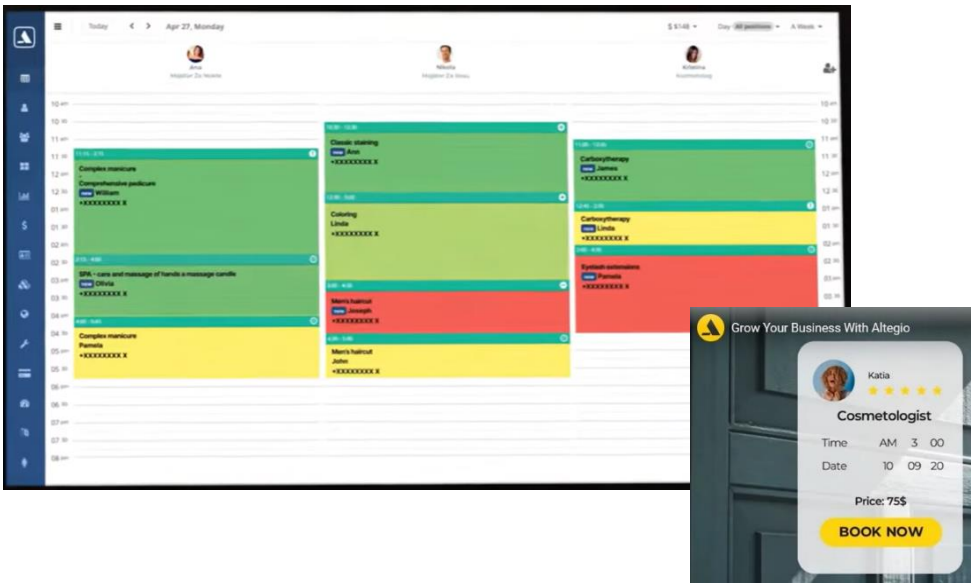


Рис. 1.4 – Інтерфейс сервісу онлайн-запису *Altegio*

Ціновий діапазон користування починається з 26 євро за місячну підписку, за умови тарифу на 3 місяці. Так, мінімальний платіж становитиме 80 євро одразу за 3 місяці.

1.2.4 *Integrica*

Цей сервіс користується популярністю серед салонів краси та надає функції онлайн-бронювання послуг, базові маркетингові інструменти та аналітику.

Ключові можливості:

1. Календар. У календарі (рис.1.5) одразу можна бачити, чи є в майстрів вільні години. Події в календарі не губляться, і керівництво може аналізувати, чи настав час, наприклад, розширювати команду.

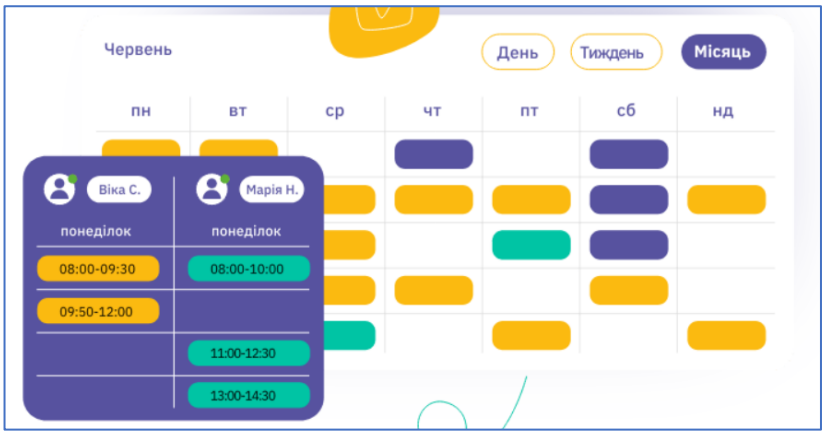


Рис. 1.5 – Календар системи *Integrica*

2. Звіти за записами. Дають можливість оцінювати завантаженість майстрів і прибуток за місяць. В онлайн режимі можна переглядати, який прибуток генерують майстри щодня. Скільки з клієнтів прийшло вперше. Можливість побачити дані за місяць, а також побудувати гнучкі звіти під власні бізнесові завдання.

3. Онлайн-запис. Не потрібно записувати кожного клієнта вручну, вони можуть записуватися самі в чат-боті чи на сайті (рис.1.6). Записи синхронізуються із календарем і керівництво легко може стежити за щоденним розкладом.

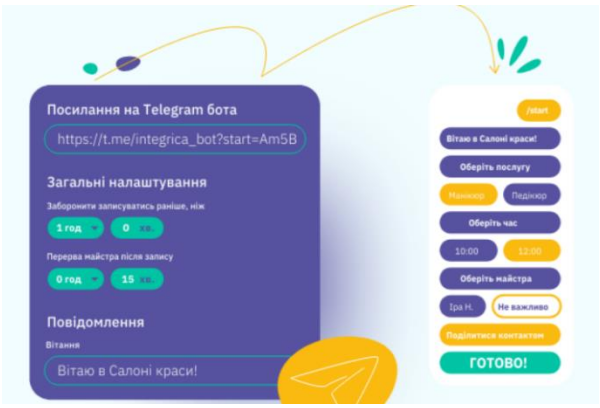


Рис. 1.6 – Онлайн-запис клієнта на сайті *Integrica*

4. База клієнтів. Можливість зберігати історію кожного клієнта й використовувати загальну базу для розсилання. З *Integrica* є можливість пам’ятати про своїх клієнтів усе: до яких майстрів вони ходять, яку суму

зазвичай витрачають, коли в кого день народження. Сукупні знання з бази даних можна використовувати для корисного цільового смс-розсилання.

Вартість тарифу *Light* складає 25 євро, тариф *Ultimate* коштуватиме 100 євро. Різниця у швидкості технічної підтримки та повноті функціоналу.

1.2.5 *CleverBox*

CleverBox CRM – український онлайн-сервіс для управління бізнес-процесами організацій у сфері здоров'я. Система пропонує онлайн-запис до лікарів та інших спеціалістів, базовий фінансовий облік, аналітику та маркетинг (рис.1.7). На сьогоднішній день системи використовують понад 500 клієнтів [1].

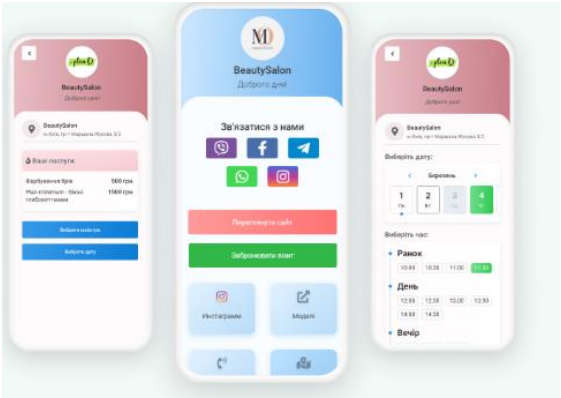


Рис. 1.7 – Онлайн бронювання в системі *CleverBox*

Функціонал та переваги:

1. Клієнтська база. Збір основної інформації клієнта: ім'я, електронна пошта, номер телефону, дата народження, фінансова інформація, придбані послуги, записи дзвінків, історія *sms*-повідомлень. Можливість прикріплення фотографії, документів до картки клієнта.

2. Маркетинг. Онлайн-залучення клієнтів. Реклама в інтернеті, соціальних мережах із прямим посиланням на онлайн-запис.

3. Телефонія. Можливість здійснювати дзвінки з картки клієнта. Збереження запису телефонних дзвінків.

4. Запис клієнта. Віджет онлайн-запису. Запис 24/7. Автоматичне узгодження запису у режимі реального часу. Виняток повторного часу. Весь час

запису займає трохи більше за хвилину. Інтеграція з електронним журналом адміністратора та спеціаліста надання послуг. Оперативне редагування візиту за будь-якими необхідними параметрами: часом, датою, послугою, майстром. Можливість залишити заявку на підбір запису на зручний час. Вибір зручного способу зворотнього зв'язку (*Viber, Telegram*). Автоматичний розрахунок вартості та тривалості візиту в залежності від послуги. Автопідбір за номером телефону та імені клієнта.

Вартість сервісу залежить від кількості працівників. Так, тариф на 1 – 3 працівників коштуватиме 45 євро, а максимальний, на 11 – 30 працівників, – 88 євро.

1.2.6 *Appointer*

Цей планувальник пропонує онлайн-запис на прийом, аналітичні звіти та простий маркетинг.

Переваги використання *CRM*-системи *Appointer* для малого бізнесу:

1. Інтеграція з *IP*-телефонією. Легко інтегрується з *IP*-телефонією від провайдерів. Адміністратор бачить ім'я абонента при вхідному дзвінку або створює нового клієнта.

2. Просте перенесення даних з інших *CRM*. Можливість переносити облікові дані клієнтів з інших програм в *Appointer*, швидко і без втрат.

3. Робота з телефону, планшета, будь-якого браузера. *CRM* не потрібно завантажувати і встановлювати. Для доступу необхідні лише інтернет і браузер, або планшет чи смартфон.

4. *SMS*-повідомлення про записи. Автоматична розсилка *sms*-нагадувань про візит клієнтам. Адміністратор самостійно вибирає частоту відправки нагадувань і редагує їх тексти.

5. Онлайн-запис через сайт і соцмережі. Клієнт самостійно записується на прийом через сайт або іншу соцмережу, тим самим знижуючи навантаження на адміністрацію.

Ціни стартують від 20 євро за місяць з можливістю оплати лише за 3, 6 або 12 місяців одразу. Мінімальний платіж становить 60 євро за 3 місяці.

1.3 Порівняння елементів систем бронювання в існуючих аналогах

Таблиця 1.1

Порівняння компонентів в існуючих аналогах

	Calendly	*EasyWeek*	*Altegio*	*Integrica*	*CleverBox*	*Appointer*
Наявність календаря	+	+	+	+	-	-
Онлайн запис на зустріч	+	+	+	+	+	+
Можливість вести клієнтську базу	-	+	+	+	+	+
Повідомлення про записи	+	+	+	+	+	+
Звіти за записами / ведення статистики	+	+	+	+	+	+
Можливість перенесення даних з інших *CRM*	-	-	-	-	+	+
Платна версія	+	+	+	+	+	+
Можливість роботи з різних пристроїв	+	+	+	+	-	+
Можливість інтеграцій із сторонніми службами та програмами	+	-	-	-	-	+

1.4 Мета та завдання проекту

У рамках дипломного роботи на основі аналізу предметної області було вирішено, що буде розроблено сайт з системою бронювання для компанії, одним з напрямків роботи якої є проведення психологічних та медичних консультацій.

Після проведення аналітичного огляду області розробки і визначення актуальності веб-сервісу, були обґрунтовані та поставлені такі задачі:

- розробити вимоги до веб-сайту онлайн бронювання;
- розробити *UML*-моделі предметної галузі;
- реалізувати базу даних та веб-сайт;
- протестувати демо-версію сайту на помилки.

Веб-сайт повинен виконувати такі основні функції:

- забезпечувати реєстрацію клієнтів – співробітників організації та потенційних учасників зустрічей;
- надавати інформацію для потенційних учасників зустрічей про зустрічі, спектр послуг, що надаються;
- містити форму для онлайн запису на зустріч.

В якості бази даних було обрано сховище документів *MongoDB.* Для реалізації системи – *Angular framework*, який надає можливість зручного кодування та спрощеного тестування

Висновок до першого розділу

У цьому розділі була проведена вступна частина дипломної роботи на тему "Розробка веб-сайту онлайн бронювання зустрічей". Використання системи бронювання забезпечує принципово нові можливості у взаємодії співробітників організації з потенційними учасниками зустрічей. Для клієнтів немає потреби в особистому контакті. З усією доступною інформацією вони можуть познайомитись онлайн, переглянути план зустрічей, оформити запис на обрану зустріч.

В першій частині роботи було проведено:

- інформаційний огляд предметної області;
- розглянуті існуючі аналоги і зроблено їх порівняльний аналіз;
- зроблено постановку завдання проекту.

В наступному розділі необхідно виконати проектування системи бронювання за допомогою *CASE*-технологій.

РОЗДІЛ 2

ПРОЕКТУВАННЯ WEB-СИСТЕМИ ОНЛАЙН БРОНЮВАННЯ

2.1 Вибір програмних засобів та архітертури системи

Створення веб-сайту є комплексною задачею, вона включає розробку серверної та клієнтської частини, які є окремими та незалежними додатками, що служать різним цілям. Тому і технології необхідно обирати окремо для кожного сайту, виходячи з задач, які він вирішує [11].

Для розробки серверу було обрано платформу для розробки вебзастосунків *ASP.NET Core* від компанії *Microsoft*. *ASP.NET Core* є логічним продовженням фреймворку *ASP.NET*, має модульну структуру та поєднує в собі раніше окремі *ASP.NET MVC, ASP.NET WebApi* та *Web Pages* у єдину програмну модель.

Даний фреймворк дозволяє створювати кроссплатформові застосунки, що повністю не залежать від платформи. Тобто ми можемо створювати, запускати та розгортати додатки не тільки на операційній системі (надалі ОС) *Windows*, а і на *Linux* та *MacOS*. В нашому випадку розробка застосунку велася на ОС *Windows*.

Ще однією перевагою *ASP.NET Core* є його модульність. Фреймворк побудований з великого набору відносно незалежних компонентів, які можуть бути завантажені як окремі модулі через пакетний менеджер *Nuget*.

Крім того, *ASP.NET Core* – вільне та відкрите програмне забезпечення, що означає неперервне розширення та покращення, яке робить даний продукт актуальним ще на великий проміжок часу.

В якості мови програмування для розробки веб-застосунків засобами *ASP.NET Core* використовувалася *C#*.

C# – об’єктно-орієнтована мова програмування, розроблена інженерами компанії *Microsoft*, як мова розробки додатків для платформи *.NET* та *.NET Core*. *C#* відносно молода мова програмування, яка була створена на основі мов *C++*

та *Java*; вона була позбавлена деяких моментів, які вважались проблемними у попередників.

Серед переваг, які надає використання мови *C#* можна виділити, поперше, наявність автоматичного збору сміття. Такий механізм дозволяє розробникам не перейматися менеджментом пам'яті: визволення об'єктів із пам'яті відбувається автоматично.

Мова *C#* має статичну типізацію, підтримує поліморфізм та перегрузку операторів. Все це дозволяє використовувати можливості об'єктноорієнтованого програмування в повній мірі і будувати додаток, який легко змінювати, масштабувати та тестувати.

Для реалізації доступу до даних в платформі *ASP.NET Core* передбачено використання *Entity Framework Core* (надалі *EF*). *EF* – це *ORM*-інструмент (*object-relational mapping* – відображення даних на реальні об'єкти), який дозволяє абстрагуватися від самої бази даних та її таблиць і працювати з даними незалежно від типу сховища. Таким чином, на фізичному рівні ми оперуємо поняттями таблиці, індекси, первинні та зовнішні ключі, а на концептуальному, який нам надає *EF,* ми вже працюємо з об'єктами.

В якості середовища розробки було обрано *Rider*. Дана *IDE* (*Integrated Development Environment*) від компанії *JetBrains* оснащена всіма необхідними інструментами для зручної і швидкої роботи з кодом, а саме інтелектуальне автодоповнення, велика кількість шаблонів, підказки для рефакторингу, інструменти для дебагу та для запуска автоматичних тестів, тощо. Крім того, *Rider* є кросплатформеним, що дозволяє його використання в різних ОС.

Для створення фронтенд частини застосунку було обрано *Angular* фреймворк, як одну із найбільш популярних технологій для вирішення подібних задач.

Angular – сучасний фреймворк, повністю побудований на *TypeScript*, і, як наслідок, використання *TypeScript* з *Angular* забезпечує безперебійну роботу сайту[11]. *Angular* розроблено, щоб зробити оновлення максимально простим –

система *Angular* складається з різноманітних груп з понад 1,7 мільйонів розробників, авторів бібліотек і творців контенту.

Angular framework заснований на компонентах, що починаються в єдиному стилі. Наприклад, кожен компонент розміщує код у класі компонента або визначає *@Component* (метадані). Ці компоненти є невеликими елементами інтерфейсу, незалежними один від одного, і пропонують кілька переваг, у тому числі:

1. Можливість повторного використання. Заснована на компонентах структура *Angular* робить компоненти дуже багаторазовими в усьому додатку. Є можливість створювати інтерфейс з рухомими частинами, забезпечуючи при цьому плавний процес розробки.

2. Спрощене модульне тестування. Будучи незалежними один від одного, компоненти значно спрощують модульне тестування. Модулі *Angular.js* мають частини програми, якими легко маніпулювати. Завдяки розділенню модулів є можливість завантажувати необхідні сервіси, ефективно виконуючи автоматичне тестування. Не потрібно запам'ятовувати порядок завантаження модулів, якщо розробник дотримується принципу «один файл– один модуль».

3. Покращена читабельність. Послідовність кодування робить читання коду простою справою для нових розробників у поточному проекті. Це підвищує їхню продуктивність і загальну ефективність проекту.

4. Простота обслуговування. Розділені компоненти можна замінювати більш досконалими реалізаціями. Простіше кажучи, це дозволяє ефективно обслуговувати та оновлювати код.

Програмне забезпечення або його елементи не потребують графічного інтерфейсу для взаємодії один з одним. Програмні продукти обмінюються даними та функціональними можливостями через інтерфейси *API* – інтерфейси прикладного програмування.

В якості бази даних було обрано сховище документів *MongoDB.*

Реляційні бази даних зберігають інформацію в строго регламентованих таблицях і стовпцях. *MongoDB* — це сховище документів, яке зберігає

інформацію в колекціях і документах. У *MongoDB* дані зберігаються неструктуровано у вигляді спеціальних документів. Вони записані у форматі *BSON* – це бінарна версія популярного формату *JSON*. Всі документи в базі — це набір пар «поле-значення», де як значення може бути будь-що, від чисел і дат до інших вкладених документів. Неважливо, в якому вигляді та форматі створено документ – він спокійно «ляже» до бази даних *MongoDB* без попередньої обробки (рис. 2.1).

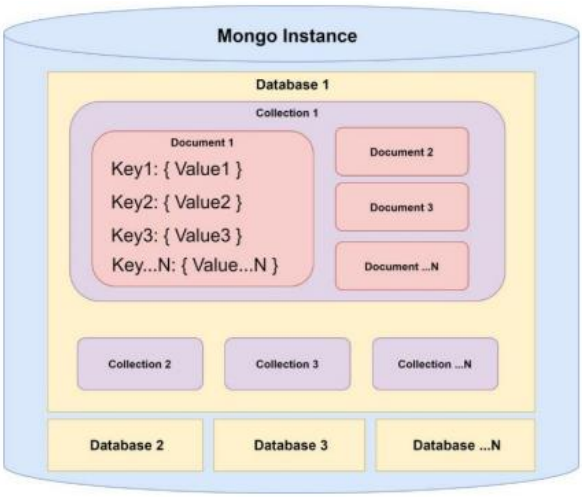


Рис.2.1 – Сховище документів *MongoDB*

Гнучкість, успадкована в цьому типі моделювання даних, означає, що дані можна обробляти за принципом використання і потреби, забезпечуючи переваги продуктивності.

Для проектування системи бронювання використовуються об'єктно-орієнтован технології, які включають застосування мови *UML*.

Уніфікована мова моделювання *UML* (*Unified Modeling Language*) – це універсальна мова візуального моделювання систем. Хоча найчастіше *UML* асоціюється з моделюванням об’єктно-орієнтованих програмних систем, вона має набагато ширше застосування завдяки властивій їй розширюваності [17]. *UML* об'єднав найкращі сучасні технічні прийоми моделювання та розробки програмного забезпечення. По суті, мова *UML* була задумана так, щоб її можна було реалізувати за допомогою її ж інструментальних засобів. Фактично це визнання того, що великі сучасні програмні системи, як правило, потребують

інструментальної підтримки. *UML* діаграми легко сприймаються і при цьому легко генеруються комп'ютерами. Важливо розуміти, що *UML* не пропонує нам якоїсь методології моделювання. Звичайно, деякі методичні аспекти маються на увазі елементами, що становлять модель *UML*, але сам *UML* є лише візуальним синтаксисом, який можна використовувати для створення моделей.

В *UML* існує чотири загальні механізми, послідовно застосовуваних до всієї мови моделювання. Вони описують чотири стратегії підходу до моделювання об'єктів, які в різних контекстах багаторазово застосовуються у *UML*. Це ще раз переконує нас у простоті та елегантності структури *UML*.

Моделі *UML* мають принаймні два виміри: графічний, що дозволяє візуалізувати модель за допомогою діаграм і піктограм, та текстове, що складається із специфікацій різних елементів моделі. Специфікації – це текстові описи семантики елемента. Семантика елементів моделі фіксується у специфікаціях; без них можна тільки здогадуватися, що насправді є елементом.

Набір специфікацій – це суть моделі. Специфікації формують семантичний задній план (*semantic backplane*), який поєднує модель і наповнює її змістом. Різноманітні діаграми – це просто уявлення або візуальні проекції цього плану.

Розробку моделей за допомогою *UML*, як правило, починають з графічної моделі, яка дозволяє візуалізувати систему, а потім по мірі її розвитку додають на задній план все більше і більше семантики. Однак модель можна вважати корисною чи повною, тільки якщо семантика моделі присутня у семантичному задньому плані.

Клас інструментальних засобів розробки систем та його супроводу утворюють програмні продукти названі *CASE*-системами. В якості *CASE*-засобу для реалізаціїї проєктування предметної області заданої теми обрано *Microsoft Visio*.

Під ітоговою *UML* моделлю в *Microsoft Visio* розуміється опис системи з певної точки зору, що використовується для візуалізації, опису та документування характеристик системи. Її використання дозволяє зрозуміти

поведінку та структуру майбутньої програмної системи, зменшити потенційні ризики при розробці та майбутньому використанні.

Загалом, архітектура програмного забезпечення відноситься до структури, макета та основних компонентів системи та визначає, як частини системи взаємодіють і з'єднуються одна з одною. Коли говорять про архітектуру веб додатку, перш за все, описують зв'язки між його структурними частинами високого рівня:

- клієнтська програма, з якою користувачі взаємодіють у браузері;
- веб-сервер, який обробляє запити та надає відповіді;
- бази даних, які зберігають і отримують доступ до даних;
- хмарні сервіси (для хмарних рішень);
- *API* та сторонні інтеграції тощо.

Веб-програми складаються з кількох рівнів, що відповідають за певні функції. Залежно від структури програма може мати кілька рівнів і представляти 2-рівневу архітектуру, 3-рівневу архітектуру або багаторівневу архітектуру. Типова веб-програма має три рівні: *presentation, application* та *data* (рис.2.2).

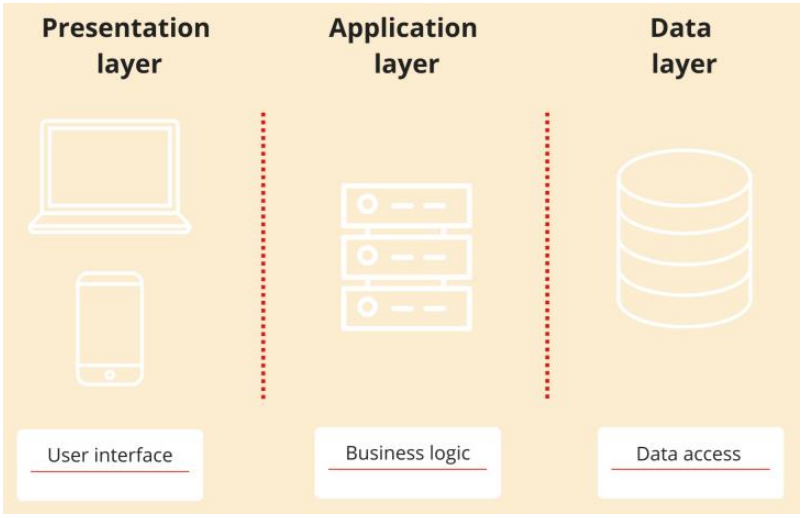


Рис. 2.2 – Рівні високорівневої архітектури веб-додатків

Для розробки веб-сайту була обрана трирівнева клієнт-серверна архітектура.

Презентаційний рівень обробляє взаємодію користувача з клієнтською програмою. Прикладний рівень містить бізнес-логіку та організовує робочі процеси. Рівень даних керує зберіганням даних і керуванням ними. Рівні архітектури веб-програми взаємопов'язані, щоб забезпечити безперебійну роботу [12].

Класичний приклад веб-додатку складається з двох основних компонентів: програми на стороні клієнта, яка виконується в браузері, і коду на стороні сервера. Ці компоненти утворюють невід'ємні частини вебпрограми та можуть приймати різні форми в різних архітектурних стилях (рис.2.3).

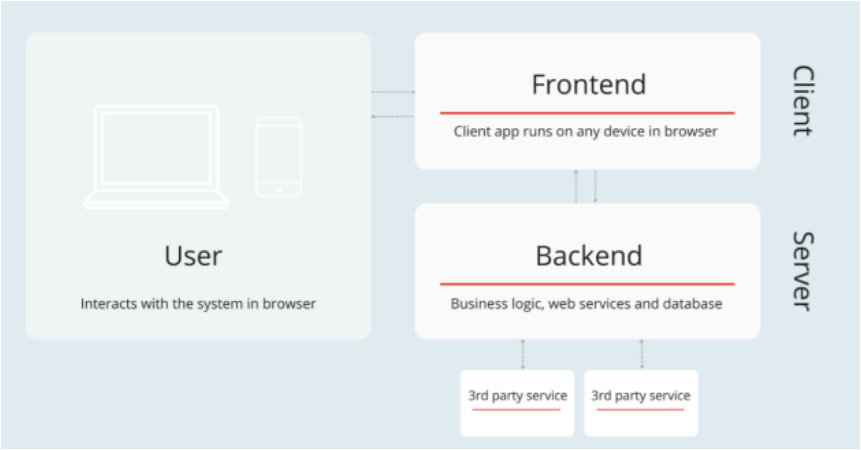


Рис. 2.3 – Компоненти веб-додатку

Клієнтська частина стосується динамічного веб-інтерфейсу, який користувачі можуть відкривати у браузері та взаємодіяти з ним у режимі реального часу – надсилати запити, прокручувати та переглядати сторінки, використовувати онлайн послуги тощо. Чуйний, насичений і швидкий інтерфейс створено за допомогою сучасних мов програмування, бібліотек *JavaScript* і фреймворків. Серверні компоненти отримують, обробляють і відповідають на запити, що надходять від клієнтської програми. На схемі архітектури вебпрограми (рис. 2.3) можна побачити, що серверна частина зазвичай керує бізнес-логікою та виконує важку роботу – перевіряє запити користувачів, зберігає та витягує дані з сервери бази даних відповідають на ці запити.

2.2 Проєктування задач онлайн бронювання за допомогою *CASE* технологій

2.2.1 *Use case* діаграми

Use case діаграма (діаграма варіантів використання) описує варіанти використання функцій системи для різних груп користувачів, іншими словами, те, що робить користувач і що система буде робити у процесі свого функціонування. Діаграма варіантів використання складається з користувачів («акторів») та їх «ролей», для яких система виконує дію, та власне дій (*Use Case),* які описують те, що користувач «актор» хоче одержати від системи.

Діаграми варіантів використання для системи в цілому з позицій адміністратора системи, співробітника компанії та клієнта представлені на рисунках 2.4, 2.5 та 2.6. Будь-який користувач системи, крім відвідувача, мають бути зареєстрованими користувачами, тому для роботи із системою повинні пройти авторизацію.

Користувач адміністратор має доступ до всіх даних системи та має право на їхнє додавання, видалення, оновлення (рис.2.4). Адміністратор системи, співробітники компанії та клієнти можуть задавати параметри пошуку та отримувати звітну інформацію, наприклад, про розклад співробітників або заплановані зустрічі.

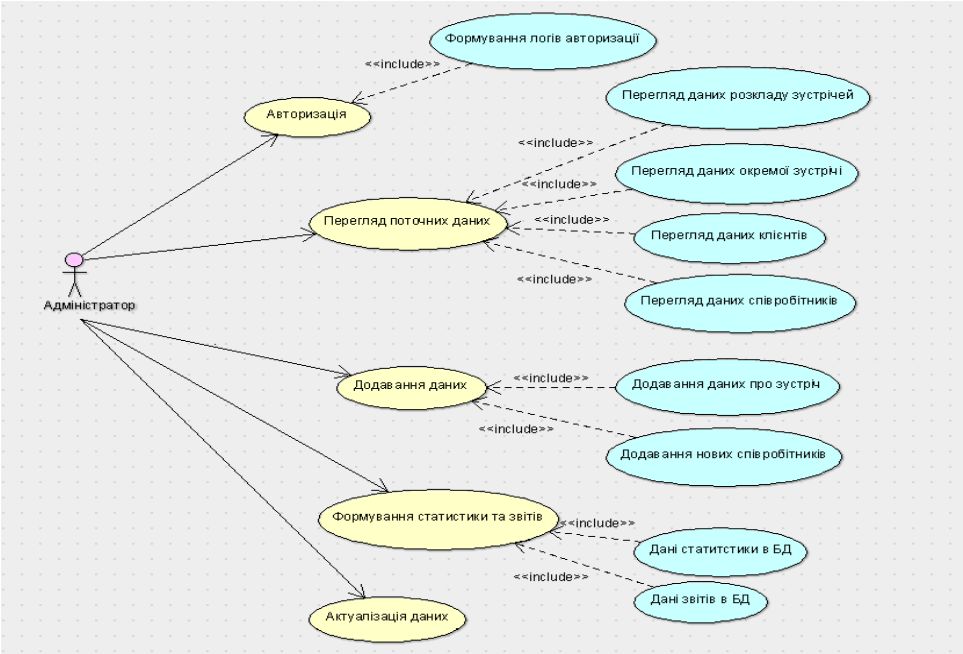


Рис. 2.4 – Діаграма варіантів використання для адмініістратора

Користувач співробітник, крім перегляду загальнодоступної інформації (новин, зустрічей тощо) може переглядати дані про співробітників, зустрічі, учасників, надсилати повідомлення учасникам зустрічей, планувати нові зустрічі, бронювати запис для учасників, переглядати свій календар (рис.2.5).

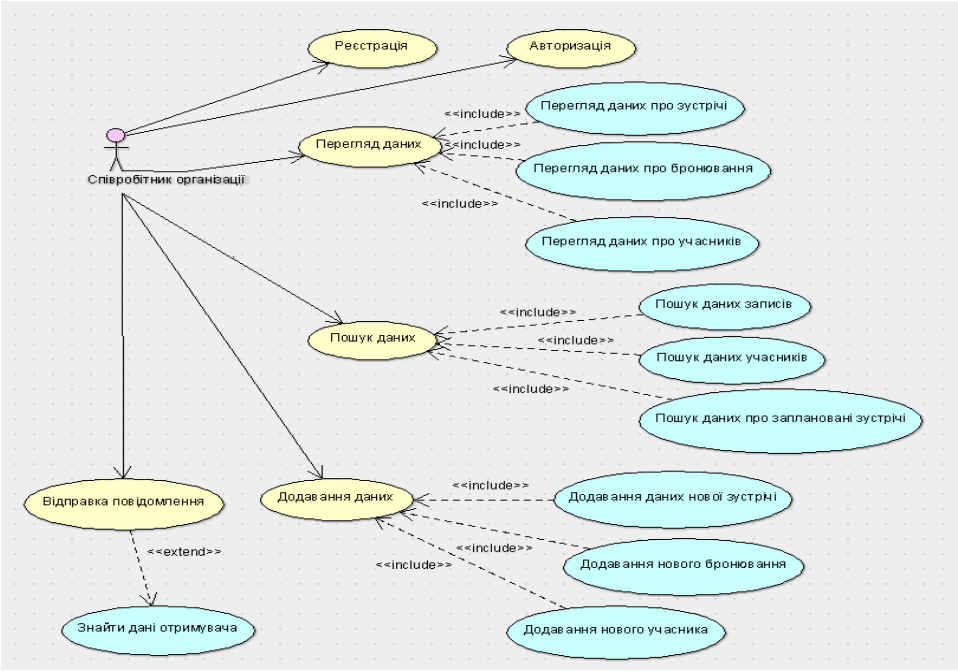


Рис. 2.5 – Діаграма варіантів використання для співробітників

Користувач клієнт організації, крім перегляду загальнодоступних даних про організацію, співробітників, може зробити пошук можливих зустрічей, вибрати цікаву для нього зустріч та, після авторизації, записатися на неї (забронювати), надіслати повідомлення адміністратору або співробітнику. Також він може переглядати перелік зустрічей, на які він зробив бронювання (рис.2.6).

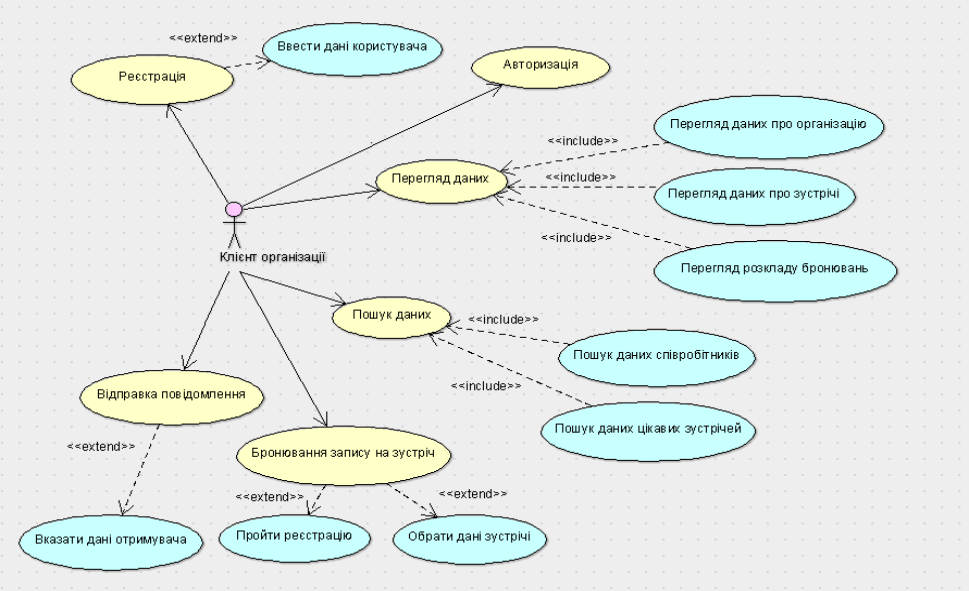


Рис. 2.6 – Діаграма варіантів використання для клієнта організації

2.2.2 *Activity* діаграма

Activity діаграми *UML* (діаграми діяльності) відображають аспект реалізації варіанта використання, що включає дії користувачів (діяльності), дії системи, знаки синхронізації, стани, а також переходи між ними. Діаграма діяльності системи з позиції адміністратора наведена на рисунку 2.7.

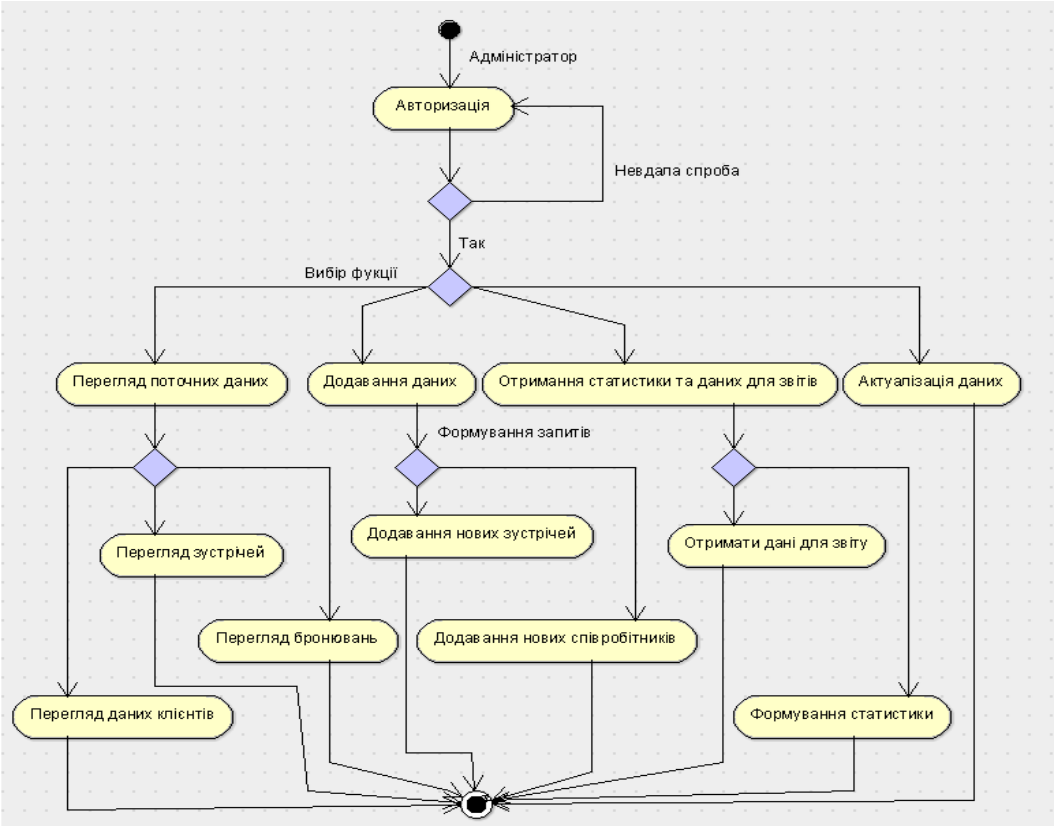


Рис. 2.7 – Діаграма діяльності

Адміністратор системи використовує вкладку "Вхід для адміністратора". Після цього він має ввести логін і пароль. Якщо пароль введено не так, система не розпізнає цього користувача, як адміністратора та пропонує повторно ввести пароль. Якщо логін і пароль введені правильно, система пропонує варіант роботи з базою даних. Можливі варіанти перегляду даних, оновлення або видалення даних. Таким чином, наприклад, можуть бути введені або видалені дані нового клієнта організації. Для управління даними у БД існують спеціальні форми для додавання, видалення, оновлення даних, незалежно від структури цих даних. Даними, що найбільш часто потрібно оновлювати, для адміністратора є дані новин, зустрічей та клієнтів організації.

2.2.3 *State* діаграма

State діаграма (діаграма стану) описує можливі послідовності станів та переходів, які в сукупності характеризують поведінку системи протягом життєвого циклу. Діаграма станів представляє динамічну поведінку сутностей,

на основі специфікації їхньої реакції на сприйняття деяких конкретних подій. Діаграми стану найчастіше використовуються для опису поведінки окремих систем та підсистем. Вони також можуть бути застосовані для специфікації функціональності екземплярів окремих класів, тобто для моделювання всіх можливих змін станів конкретних об'єктів.

Діаграма станів – це граф спеціального виду, який служить для уявлення кінцевого автомата. Основними поняттями, що описують кінцевий автомат, є стан та перехід. Передбачається, що перехід об'єкта зі стану в стан відбувається миттєво. Діаграма станів для варіанта використання наведена на рис.2.8.

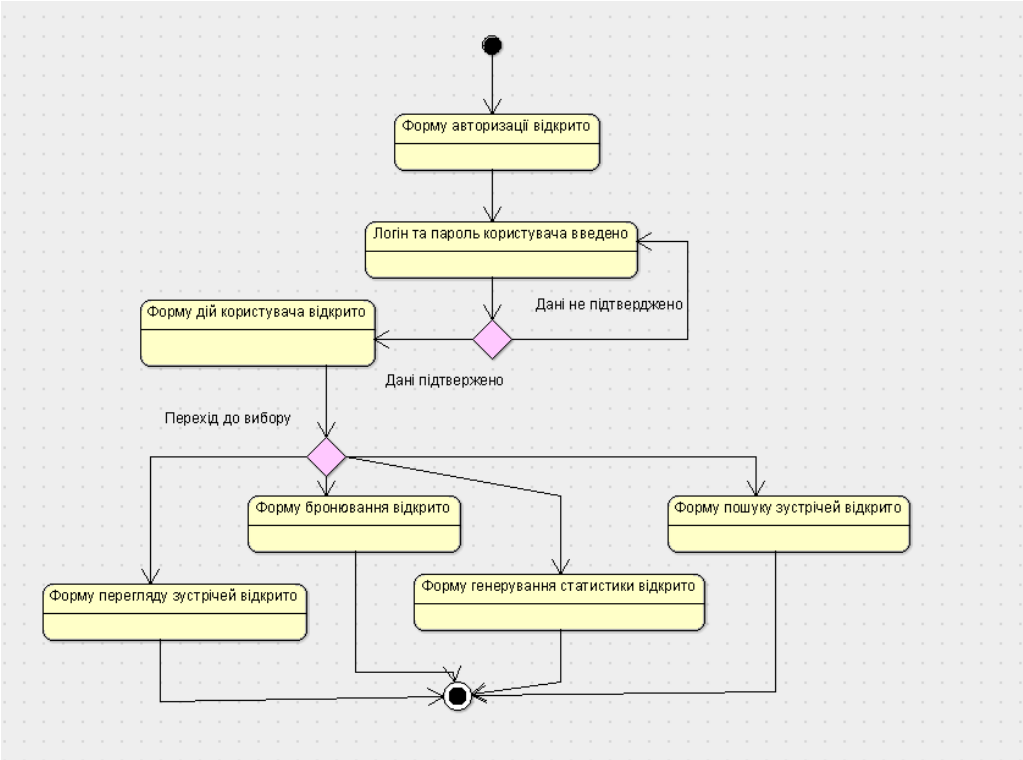


Рис. 2.8 – Діаграма стану для системи бронювання

2.2.4 *Sequence* діаграми

Sequence діаграми *UML* (діаграми послідовності) відображають комунікаційний аспект реалізації варіанта використання та включають у свій склад об'єкти та повідомлення між ними та моделюють взаємодію об'єктів у часі. На діаграмі послідовності зображуються об'єкти, які беруть участь у взаємодії. Для діаграми послідовності основним моментом є динаміка взаємодії об'єктів.

Кожна взаємодія описується сукупністю повідомлень, якими об'єкти, що беруть участь у ньому, обмінюються між собою. Повідомлення є закінченим фрагментом інформації. Прийом повідомлення викликає виконання певних дій об'єктом, якому повідомлення відправлено. Повідомлення передають інформацію та вимагають або передбачають виконання певних дій від об'єкта, що приймає повідомлення. Діаграма послідовності наведена рис 2.9.

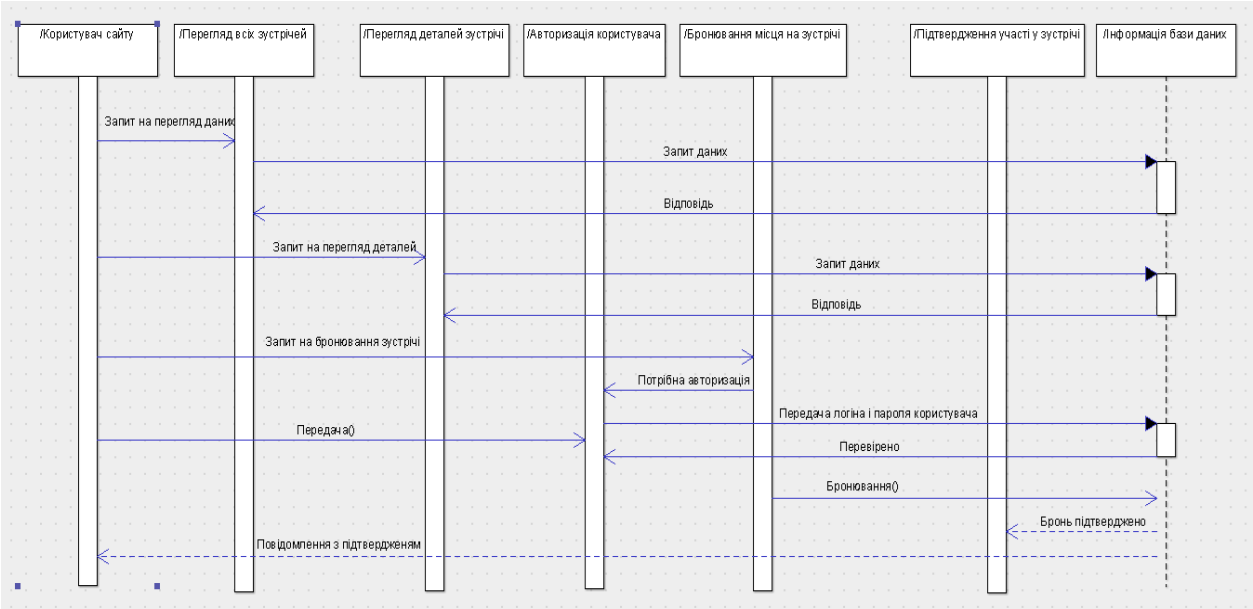


Рис. 2.9 – Діаграма послідовності процесу Бронювання участі в зустрічі

Повідомлення спеціального характеру – це дані передані системі про логін та пароль користувача. Після обробки цих даних система надсилає повідомлення користувачу про готовність або не готовність працювати з користувачем. Основним типом повідомлень частини, пов'язаної з отриманням інформації, є закладені в систему запити до бази даних. У відповідь система відправляє web-сторінку з вбудованими даними запиту.

2.3 Проєктування структури та інтерфейсу веб-сайту

Структура сайта – це різні сторінки сайта, що пов’язані одна з одною за допомогою внутрішніх посилань та їхньої ієрархії. Це те, як інформація на сайті організована та представлена, аби алгоритм добре читав її контекст. Хороша структура веб-сайта полегшує навігацію як для користувачів, так і для сканерів,

що покращує рейтинг *SEO* веб-сайта в пошукових системах. Для доступності і простоти у використанні необхідно створити веб-сайт із чудовим *UX* («*user experience*»), адже чудова структура веб-сайта покращує зручність використання веб-сайта, полегшуючи користувачам пошук того, що вони шукають. Щоб створити структуру веб-сайта, потрібно намітити, як буде організовано її склад на сайті (головна сторінка, категорії, окрема сторінка, система бронювання).

Ось чому структурування веб-сайта має бути першим кроком у будь-якому проєкті веб-дизайну. Більшість веб-сайтів у верхній частині кожної сторінки містять інформацію про весь сайт, таку як логотип веб-сайта, функція пошуку та параметри навігації [21]. *HTML5* містить елемент <*header*>, який можна використовувати для визначення такої області. Подібно до заголовка сторінки, більшість веб-сайтів також мають область внизу кожної сторінки, яка містить інформацію для всього сайта, таку як інформація про авторські права, заяви про конфіденційність або застереження. *HTML5* містить елемент <*footer*>, який можна використовувати для визначення такої області. Веб-сторінка може мати будь-яку кількість навігаційних меню. Для визначення області основного вмісту веб-сторінки або програми використовують елемент *HTML*5 <main>. Зовнішня структура сайта показана на рис. 2.10.

Рис. 2.10 – Схема зовнішньої структури сайта

Основним принципом відмінної структури веб-сайту є інформаційна архітектура, яка гарантує, що вміст організовано, структуровано та позначено ефективно та послідовно. Найпоширенішою структурою веб-сайта є ієрархічна структура, яка була обрана для реалізації дипломної роботи (рис. 2.11), і котра

базується на одній батьківській (головній сторінці) і дочірніх сторінках (категоріях і підкатегоріях), які випливають з головної сторінки.

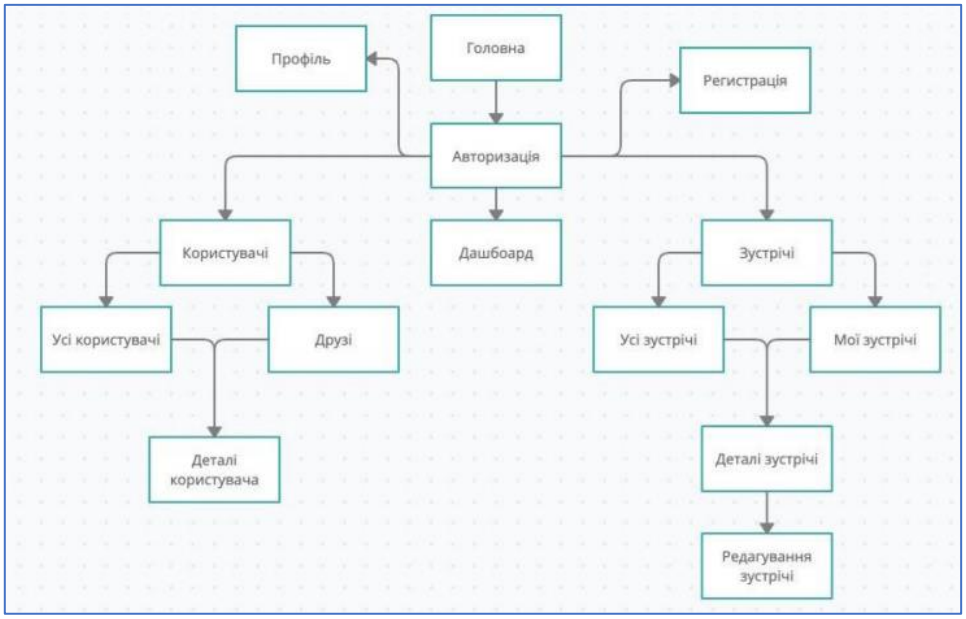


Рис. 2.10 – Логічна структура веб-сайту бронювання

Для захисту даних організації, доступ до сервісу бронювання запису буде надаватися лише зареєстрованим користувачам. Сторінка реєстрації містить поля для вводу імені, прізвища, електронної адреси, мобільного телефону, дати народження та паролю і кнопку для відправлення форми. На рисунку 2.11 представлений макет сторінки реєстрації.

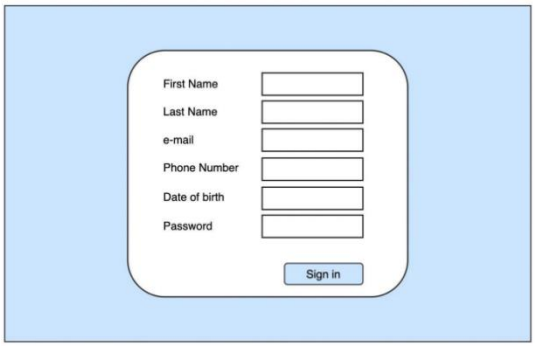


Рис. 2.11 – Макет сторінки реєстрації

Для авторизації в системі необхідно ввести свій *e-mail* та пароль (рис.2.12). Користувач може мати лише один обліковий запис, закріплений за електронною адресою.

На головній сторінці сайту розміщується «шапка», що містить посилання «додому», «особистий кабінет» і вихід з системи, іконка для відкриття чату та бокове меню з посиланнями на сторінки «*Appointments*», «*Trainers*», «*Guests*».

При натисканні на іконку чату, в нижньому правому куті з’явиться форма. Користувач зможе обрати свого тренера з випадного списку та написати йому повідомлення. Коли організатор зустрічі відкриє чат, на формі вже буде відображене повідомлення від користувача.

Сторінка «*Trainers*» містить таблицю з усіма організаторами зустрічей. Наповнивши сайт контентом, користувач зможе переглянути усіх організаторів та інформацію про них. Адміністратор сайту може додавати тренерів, видаляти та редагувати інформацію про них.

На сторінці «*Appointments*» реалізована основна функція застосунку – запис на зустріч. Вона містить поле для вибору дати, фільтри для пошуку зустрічей та календар, де відображається розклад на найближчий тиждень (рис. 2.12).

Рис. 2.12 – Макет сторінки «*Appointments*»

Для наочності, комірка часу, на яку у організатора зустрічі вже назначений прийом, буде підсвічуватись іншим кольором. Обравши потрібну зустріч та натиснувши на вільну клітинку у розкладі, відкривається форма для створення запису. Там можна буде визначити мету прийому та ввести додаткову інформацію. Після успішного запису, користувачу та організатору на електрону пошту буде відправлений лист з деталями про прийом. Кожен користувач має

доступ до «особистого кабінету» (рис.2.13). На цій сторінці можна додавати та редагувати особисту інформацію. Наприклад, якщо ви психолог, ви можете описати свій досвід роботи, розмістити додаткову інформацію про зустрічі, прикріпити фото.

Рис. 2.13 – Макет сторінки «Особистий кабінет»

Важливо закласти фундамент для майбутнього розширення продукту вже на етапі проектування. В подальшому, можливе додавання функції перегляду історії відвідувань в особистому кабінеті, оцінка зустрічей учасниками, тощо.

Висновок до другого розділу

У другому розділі проведений проектування *web*-системи онлайн бронювання та вибір програмних засобів для реалізації завдання.

Етап проектування інформаційних систем є необхідною складовою життєвого циклу програмного забезпечення. Його виконання впливає в кінцевому рахунку на успішність програмного засобу серед кінцевих користувачів. Для проектування програмного забезпечення використовували об’єктно-орієнтовану методологією проектування. На цьому етапі роботи над темою було побудовано *Use case* діаграми, *Activity* діаграму, *State* діаграму та *Sequence* діаграми для системи бронювання.

Також проведено аналіз архітектури веб-додатків та методів розробки веб-застосувань. Сформовано логічну структуру веб-сайту бронювання. Після виконання проектування можна переходити до наступного кроку роботи – реалізації спроектованої системи.

РОЗДІЛ 3

ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-САЙТУ ТА БАЗИ ДАНИХ

3.1 Розробка бази даних

В першу чергу необхідно створити та налаштувати інфраструктуру проекту відповідно до обраної архітектури. Засобами *Rider* створимо головне рішення *MeetingsOrg.sln* та додамо туди наступні проекти:

1. *Domain* – ядро нашого застосунку.
2. *DataAccess* – сервер бази даних.
3. *Services* – сервіси для роботи з доменними моделями.
4. *Identity* – інтеграція *ASP NET Core Identity*.
5. *WebApi* – рівень користувальницького інтерфейсу.

Наступним кроком буде додавання бази даних і потім додавання посилань в файли для забезпечення правильної їх взаємодії.

У веб-системі використовується провідна сучасна платформа баз даних загального призначення – *MongoDB*, яка розроблена, щоб розкрити потужність програмного забезпечення та даних для розробників і програм, які вони створюють.

«App_Meetings» це імя бази даних веб-сайта, яка включає в себе колекції даних: *meets, appoiments, guests, rooms* (рис. 3.1). Ці колекції вміщують в собі основну інформацію про користувачів (ім’я, місто, номер телефону, день народження, соціальні мережі) та про зустрічі/події (назва зустрічі, її тематика, опис, місце та час проведення), користувачів, які будуть на зустрічі – в якості спеціаліста (лікаря або психолога) та клієнта\пацієнта (рис.3.2 та рис. 3.3).

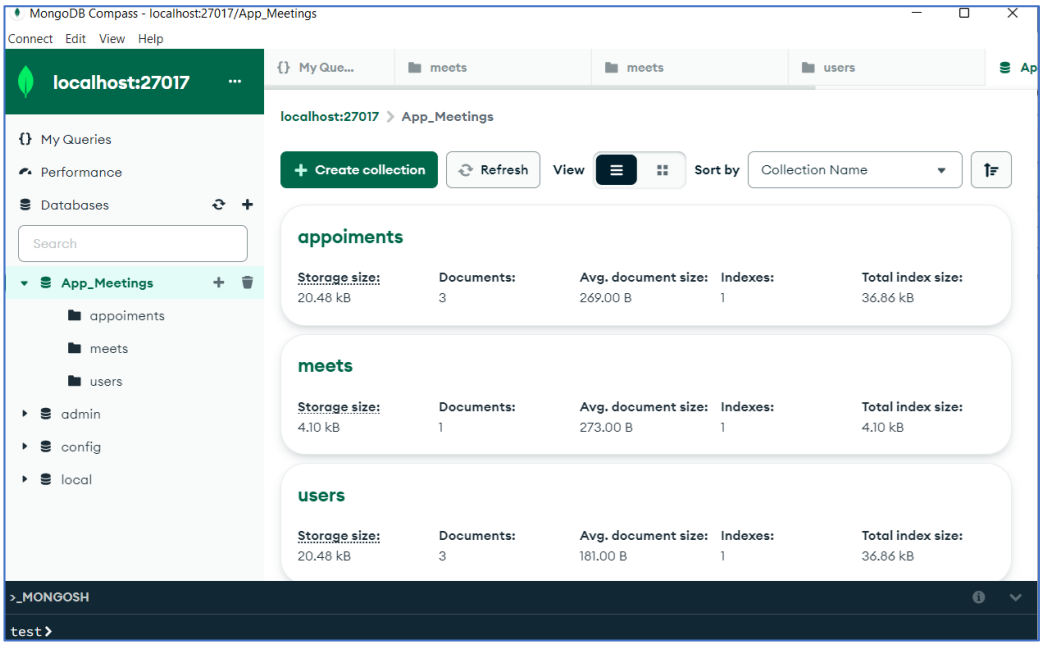


Рисунок 3.1 – Адміністративна панель бази даних «*App_Meetings*»

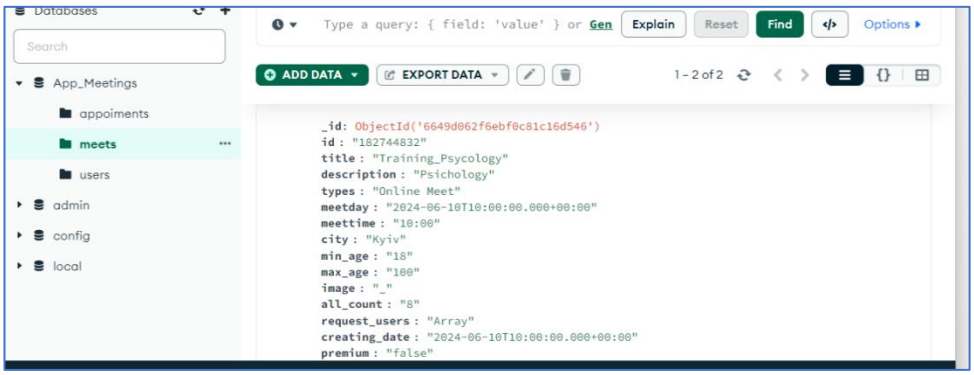


Рисунок 3.2 – Приклад даних в колекції «*meets*»

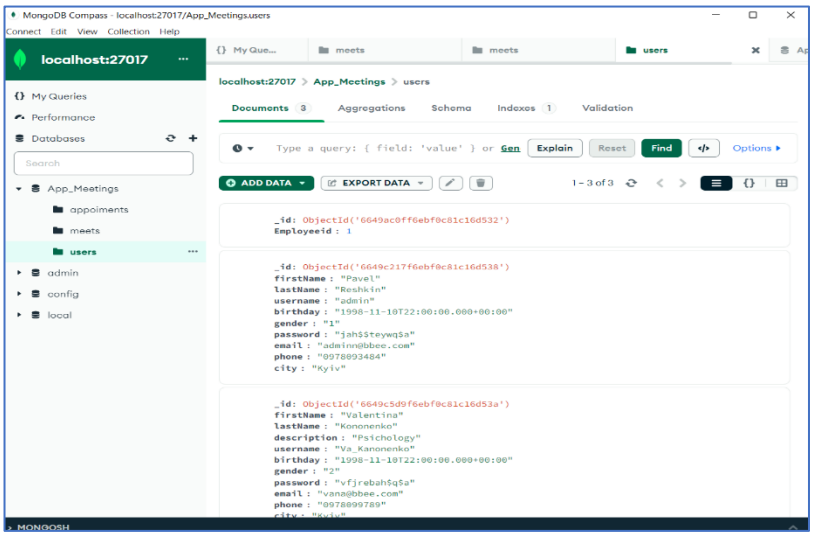


Рисунок 3.3 – Приклад даних в колекції «*users*»

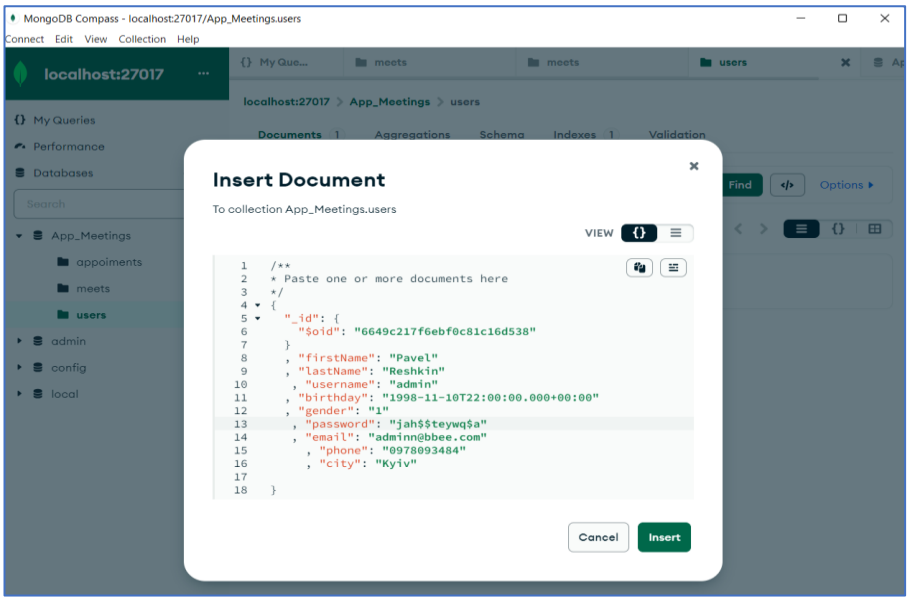
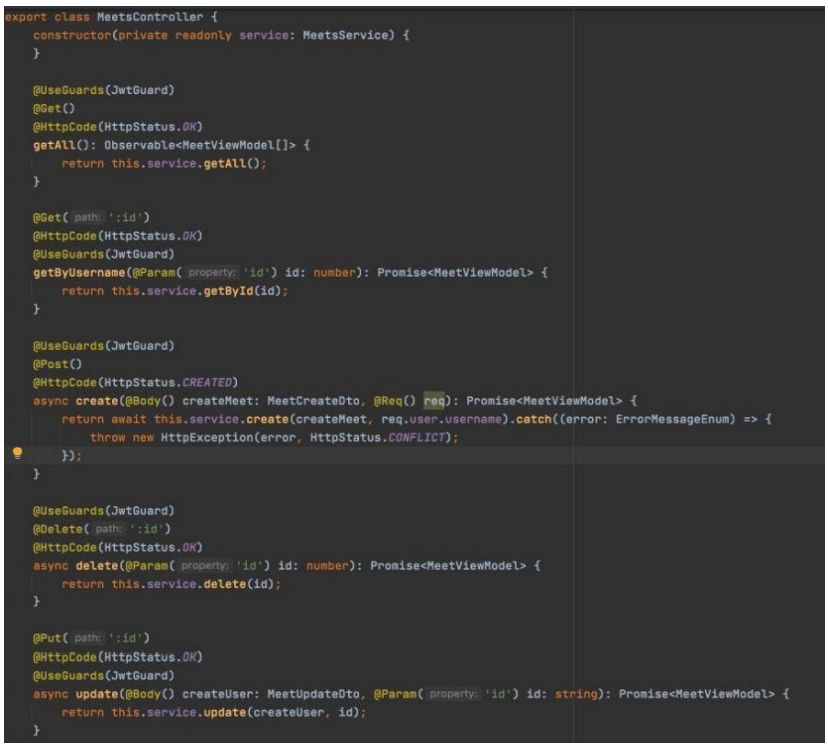


Рисунок 3.4 – Приклад додавання даних в колекцію «*users*»

Для звернення від клієнтської частини системи до серверної використовуються контролери, які визначають, які функції будуть викликані на серверній частині веб-сайту (рис. 3.5, 3.6).

```
@Controller( prefix: 'users')
export class UsersController {
    constructor(private readonly userService: UsersService) {}

    @UseGuards(JwtGuard)
    @Get()
    getAll(@Req() req): Observable<Users[]> {
        const currentUsername = req.user.username;
        return this.userService.getAll(currentUsername);
    }

    @Get( path: ':username')
    getByUsername(@Param( property: 'username') username: string): Promise<Users> {
        return this.userService.getByUserName(username);
    }

    @UseGuards(JwtGuard)
    @Post()
    @HttpCode(HttpStatus.CREATED)
    async create(@Body() createUser: CreateUserDto): Promise<Users> {
        return await this.userService.create(createUser).catch((error: ErrorMessageEnum) => {
            throw new HttpException(error, HttpStatus.CONFLICT);
        });
    }

    @UseGuards(JwtGuard)
    @Delete( path: ':id')
    async delete(@Param( property: 'id') id: number): Promise<Users> {
        return this.userService.delete(id);
    }

    @Put( path: ':username')
    @HttpCode(HttpStatus.OK)
    @Roles(RoleEnum.Admin)
    @UseGuards(JwtGuard, RolesGuard)
    async update(@Body() createUser: UpdateUserDto, @Param( property: 'username') username: string): Promise<Users> {
        return this.userService.update(createUser, username);
    }
```

Рисунок 3.5 – Зміст контролера для користувачів



```
export class MeetsController {
    constructor(private readonly service: MeetsService) {
    }

    @UseGuards(JwtGuard)
    @Get()
    @HttpCode(HttpStatus.OK)
    getAll(): Observable<MeetViewModel[]> {
        return this.service.getAll();
    }

    @Get( path: ':id')
    @HttpCode(HttpStatus.OK)
    @UseGuards(JwtGuard)
    getByUsername(@Param( property: 'id') id: number): Promise<MeetViewModel> {
        return this.service.getById(id);
    }

    @UseGuards(JwtGuard)
    @Post()
    @HttpCode(HttpStatus.CREATED)
    async create(@Body() createMeet: MeetCreateDto, @Req() req): Promise<MeetViewModel> {
        return await this.service.create(createMeet, req.user.username).catch((error: ErrorMessageEnum) => {
            throw new HttpException(error, HttpStatus.CONFLICT);
        });
    }

    @UseGuards(JwtGuard)
    @Delete( path: ':id')
    @HttpCode(HttpStatus.OK)
    async delete(@Param( property: 'id') id: number): Promise<MeetViewModel> {
        return this.service.delete(id);
    }

    @Put( path: ':id')
    @HttpCode(HttpStatus.OK)
    @UseGuards(JwtGuard)
    async update(@Body() createUser: MeetUpdateDto, @Param( property: 'id') id: string): Promise<MeetViewModel> {
        return this.service.update(createUser, id);
    }
```

Рисунок 3.6 – Зміст контролера для подій

3.2 Розробка компонентів системи

3.2.1 Розробка статичної частини системи

Програмна реалізація продукту починається з ядра. В проекті *Domain* створимо папки *Models* та *Enums*, в яких будуть лежати бізнес об’єкти. Почнемо з перерахувань, вони знадобляться для створення моделей.

Додамо наступні перерахування:

1. *UserRole (string – int*) – визначає роль користувача в системі і зберігає значення “*Klient*”, “*Sotr*” та “*Admin”*.

2. *SotrSpecialization (string – int*) – визначає спеціалізацію співробітника, може розширюватись у майбутньому.

3. *MeetsType (string – int*) – визначає тип зустрічі.

4. *AppointmentPeriod (string – int)* – визначає період, за який буде відображений розклад. Має значення “*Week*” та “*Day*”.

Тепер, коли є необхідні перерахування, додамо в папку *Models* такі класи:

BaseModel.cs. Має єдину властивість “*Id*” типу *int*.

2. *Person.cs.* Наслідується від *BaseModel* та має властивості "*FirstName*", "*LastName*", "*Email*", "*PhoneNumber*" типу string та "*DateOfBirth*" типу *DateTime*.

3. *Klients.cs.* Наслідується від класу *Person* та має властивість *int "PersonalNumber"*.

4. *Sotr.cs.* Наслідується від класу *Person* та має властивість "*Specialization*" типу *SotrSpecialization*.

5. *Appointment.cs.* Наслідується від *BaseModel* та має властивості, повязані з записом на зустріч.

6. *ChatUser.cs*. Має властивості "*Id*" типу *int*, "*UserName*" типу *string* "*UserRole*" типу *UserRole*.

Серед базових функцій буде також відправлення листів на електронну пошту користувачам при реєстрації, успішному записі на зустріч або при відміні запису. На даному етапі неважливо яким чином буде відправлятися лист, всі деталі реалізації будуть подані на наступному рівні. Домен має визначати лише контакти для роботи з листами.

Під папкою *Models* створимо класи *MailRequest* та *Message*. *MailRequest* буде мати наступні властивості типу *string*:

– *EmailTo;*

– *EmailFrom;*

– *Subject;*

– *Body*.

Клас *MailRequest* має такі властивості:

– *string Sender;*

– *string Recepient;*

– *string Text;*

– *bool Incoming.*

Створимо папку *Contracts*. Додамо туди інтерфейс *IEmailProvider*, який буде визначати метод S*endEmailAsync* з наступним кодом (Листінг 3.1).

Листінг 3.1. Метод *SendEmailAsync*

```
Task SendEmailAsync(MailRequest mailRequest, CancellationToken token)
```

Створимо папку *Services*, в яку покладемо клас *SendEmailHelper*. Цей клас служитиме для формування листів відповідно до події, що ініціює відправку. Додамо приватне поле типу *IEmailProvider*.

Клас має наступні методи:

– *private CreateEmailForNewKlient* – приймає клієнта як параметр і повертає об'єкт класу *MailRequest*. В тілі методу формується новий лист відповідно до події (реєстрація нового клієнта) та даних клієнта;

– *private CreateEmailForNewSotr* – аналогічно до попереднього тільки для співробіника;

– *private CreateEmailForNewAppointment* – приймає об'єкти класів *Person* та *Appointment* і формує лист до створення нового запису;

– *private CreateEmailForDeletedAppointment* – аналогічно до попереднього, тільки подією є видалення запису;

– *public SendEmailToNewKlientAsync* – приймає об'єкти класів *Klient* та *CancellationToken*, повертає *Task*. В тілі методу викликає відповідний приватний метод для створення листа і метод *IEmailProvider.SendEmailAsync* з отриманим листом;

– *public SendEmailToNewSotrAsync* – аналогічно для співробітника;

– *public SendEmailWithNewAppointmentAsync* – приймає об'єкти класів *Rerson, Appointment* та *CancellationToken*, повертає *Task*. В тілі методу викликає відповідний приватний метод для створення листа і метод *IEmailProvider.SendEmailAsync* з отриманим листом;

– *public SendEmailWhenAppointmentCanceledAsync* – аналогічно з попереднім, але для події видалення запису.

Для реалізації чату необхідно в папку *Contracts* додати інтерфейс *IAuthirizedUsers*. Інтерфейс буде визначати три методи:

– *void AddUser(ChatUser user);*

– *void RemoveUser(string id);*

– *IReadOnlyCollection<ChatUser> GetAuthorizedUsers().*

Під папкою *Services* створимо сервіс *AuthirizedUsers*, який буде реалізовувати інтерфейс з попереднього кроку.

Проект *DataAccess* виконує функції доступу до даних. Для роботи з фреймворком, необхідно завантажити відповідні розширення та додати їх у проект.

Перш за все, потрібно визначити класи, які описують об’єкти, що зберігаються в базі даних, відповідно до розробленої структури в попередньому розділі.

Створимо папку *Entities*. Додамо туди наступні класи:

− *BaseEntity;*

− *Sotr;*

− *Klients;*

− *Rooms;*

− *Appointment.*

Додамо папку *Repositories*, всередині створимо папку *Contracts.* Під папку *Contracts* покладемо інтерфейс *IBaseRepository*. Він буде визначати методи, спільні для всіх об’єктів.

Завдяки засобам мови *C#* можливо створювати узагальнені класи, таким чином, абстрагуємось від конкретної реалізації і, замість п’яти репозиторіїв для кожної бізнес-сутності, маємо лише один – спільний для всіх. Створимо клас *BaseRepository*, який буде реалізовувати інтерфейс з попереднього кроку.

Клас має приватні поля *DbContext* та *DbSet<T>*, приймає контекст бази даних в конструкторі і заповнює поля відповідними значеннями.

Всі необхідні інструменти для роботи з базою вбудовані в *EF*, в нашій реалізації потрібно лише викликати відповідні методи.

Наступним етапом буде наповнення проекту *Services*. Саме тут будуть визначені сервіси для роботи з доменними моделями, а також сервіси з підключенням зовнішніх бібліотек.

Так як на різних рівнях є оперування різними моделями (на рівні доступу до даних – *Entity*, на рівні сервісів – *Domain models*), потрібен механізм, який буде конвертувати одні моделі в інші. Для вирішення цієї задачі необхідно використовувати готове рішення від *Microsoft – AutoMapper*.

Щоб користуватися можливостями *AutoMapper*, спершу потрібно завантажити пакет розширення (рис. 3.7) за допомогою *Nuget Manager*.

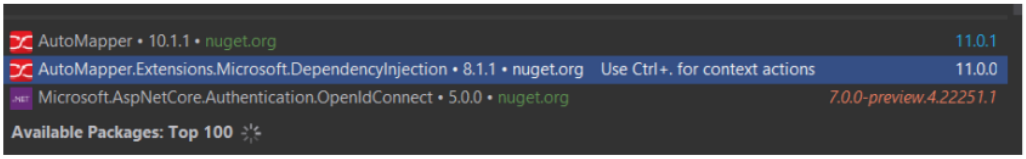


Рис. 3.7 – Пакет розширення «*AutoMapper*»

Далі створимо папку *AutoMapper*, в якій буде лежати клас *AutoMapperProfile*. Цей клас буде унаслідуватись від класу *Profile*. Для цього в клас необхідно включити простір імен *AutoMapper*.

Цей клас не буде визначати жодного метода, вся необхідна логіка лежить в пакеті, який було завантажено. Потрібно лише визначити конструктор, який буде викликати вбудовані методи *CreateMap<TTarget,TSourse>()* та *ReverseMap()* для кожної можливої конфігурації.

Тепер створимо папку *Services*, а в ній папку *Contracts*. Додамо інтерфейс *IKlientsService*. Інтерфейс визначає сигнатуру основних методів для *CRUD (Create, Update, Delete, Get)* операцій. А також містить два визначення методів для отримання кількості клієнтів, та отримання клієнтів за електронною поштою. В папку *Services* додамо клас *KlientService*. Він буде реалізовувати інтерфейс з попереднього кроку.

Клас *KlientService* має приватні поля.Значення цих полів вводяться в конструкторі, за допомогою ін’єкції залежностей. Кожен метод викликає відповідний метод репозиторію, конвертує отриману *Entity* в цільову модель, та, за необхідності викликає метод сервісу відправки листів. Використання інтерфейсу *IBaseRepository<TEntity>* забезпечує незалежність від способу

отримання та зберігання даних, тим самим збільшує простір для модифікації та розширення.

Аналогічно необхідно створити інтерфейси та сервіси для кожної бізнес-сутності (*Sotr, Appointment*).

На етапі реалізації домену застосунку було закладено інфраструктуру для відправки листів на електронну пошту. Сервіси – це місце, де ця інфраструктура буде нарощуватися деталями реалізації.

Для реалізації цієї функції буде використане стороннє *API*:

− *MailKit* (*development environment*);

− *SendGrid* (*production environment*).

MailKit – проста, але доволі насичена функціоналом бібліотека для відправлення листів за допомогою *SMTP* протоколу. Для відправлення листа засобами *MailKit* попередньо необхідно встановити з'єднання з *SMTP* сервером.

SendGrid – велика платформа електронного маркетингу, що надає багато інструментів для створення та відправлення електронних листів. Для використання *API* від *SendGrid*, потрібно спершу зареєструватися та отримати *API Key*.

Використання двох провайдерів також робить нашу систему більш надійною: при виникненні проблем з одним провайдером, функція відправки листів все ще буде доступна завдяки роботі іншого. Перш за все, в проект *Services* необхідно завантажити відповідні розширення (рис 3.8).

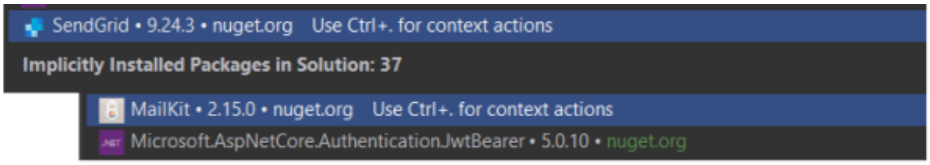


Рис. 3.8 – Пакет розширення для відправки листів

Створимо папку *Providers*, додамо туди класи:

1. *EmailConfiguration* – визначає налаштування, необхідні для відправлення листів.

2. *EmailProvider* – реалізує відправлення листів за допомогою *MailKit*.

3. *SendGridProvider* – реалізує відправлення листів за допомогою *SendGrid API*.

Метод *SendEmailAsync* (Листінг 3.2) приймає *MailRequest* та *CancellstionToken*. В тілі методу з полів *MailRequest* формується новий лист, засобами *MailKit* встановлюється з'єднання з *smtp*-сервером, відбувається аутентифікація відправника, надсилається лист та розривається з'єднання.

Листінг 3.2. Метод *SendEmailAsync*

```
public async Task SendEmailAsync(MailRequest mailRequest,
CancellationToken token)
 {
 var email = new MimeMessage
 {
 Sender = MailboxAddress.Parse(this.emailConfiguration.Email)
 };
 email.To.Add(MailboxAddress.Parse(mailRequest.EmailTo));
 email.Subject = mailRequest.Subject;
 email.Body = new TextPart(MimeKit.Text.TextFormat.Html)
 {
 Text = mailRequest.Body
 };
 using var smtp = new SmtpClient();
 await smtp.ConnectAsync(emailConfiguration.Host,
emailConfiguration.Port, SecureSocketOptions.StartTls, token);
 await smtp.AuthenticateAsync(emailConfiguration.Email,
emailConfiguration.Password, token);
 await smtp.SendAsync(email, token);
 await smtp.DisconnectAsync(true, token);
 }
```

Клас *SendGridProvider* також приймає конфігурацію в конструкторі і реалізує *IEmailProvider*. Засобами *SendGrid* створюємо новий лист, та викликаємо вбудований метод *sendEmailAsync* (Листінг 3.3).

Листінг 3.3. Метод *SendEmailAsync*

```
public async Task SendEmailAsync(MailRequest mailRequest,
CancellationToken token)
 {
 var client = new SendGridClient(emailConfiguration.ApiKey);
 var from = new EmailAddress(emailConfiguration.Email,
emailConfiguration.DisplayName);
 var subject = mailRequest.Subject;
 var to = new EmailAddress(mailRequest.EmailTo);
 var body = mailRequest.Body;
 var email = MailHelper.CreateSingleEmail(from, to, subject, null,
```

```
body);
 await client.SendEmailAsync(email, token);
 }
```

Наразі, маємо всю необхідну логіку роботи застосунку, проте без рівня інтерфейсу користувача, вона немає практичної цінності. Наповнимо контентом проект *WebApi*.

Клас *Program* створюється засобами *IDE* автоматично. На рисунку 3.9 показаний зміст класу.

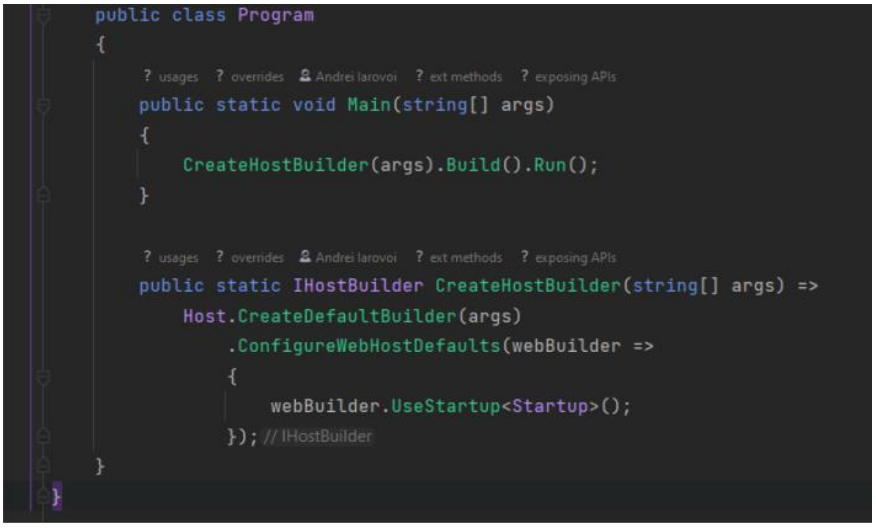


Рис. 3.9 – Клас *Program.cs*

Додаток в *ASP.NET Core* являє собою об’єкт *Microsoft.AspNetCore.Builder.WebApplication*. Цей об’єкт містить всю конфігурацію застосунку, його маршрути, залежності, тощо. Для створення об’єкту *WebApplication* потрібен спеціальний клас *WebApplicationBuilder*, який і створюється в методі *CreateHostBuilder()*. У отриманого об’єкта викликається метод *Build()*.

В кінці необхідно запустити додаток. Це робиться викликом методу *Run()*.

Клас *Startup* також створюється автоматично. Він виступає точкою входу в систему. Цей клас виконує конфігурацію застосунку, налаштовує сервіси, які будуть використовуватися, завантажує компоненти для обробки запиту.

Клас може визначати конструктор та методи *Configure()* і *ConfigureServices()*.

Конструктор є необов’язковим. Тут, як правило, виконується початкова конфігурація застосунку, заповнюються значеннями властивості *Configuration* та *Environment*.

В методі *ConfigureServices* необхідно виконати наступне:

– отримуємо *connectionString* бази даних, створюємо її контекст;

– реєструємо всі сервіси, що використовуються в системі;

– додаємо зв’язки між інтерфейсами та класами, що їх реалізують.

Appsettings.json – файл, що містить необхідні налаштування для роботи застосунку. Сюди необхідно додати *connection string* (Листінг 3.4) для з’єднання з базою даних та *MailSettings*.

Листінг 3.4. *Connection string* для з’єднання з базою даних

```
"ConnectionStrings": {
 "KodConnectionString":
"Data Source=.;Initial Catalog=KodMeeting;Integrated Security=True",
 },
 "MailSettings": {
 "Email": "kodMeeting.intern@gmail.com",
 "DysplayName": "KodMeeting",
 "Password": "kodClinic3294",
 "Host": "smtp.gmail.com",
 "Port": "587",
 "ApiKey":
"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"
 }
```

Для відображення списку бізнес-сутностей (клієнти, співробітники, тощо), створимо папку *ViewModels*, в яку покладемо клас *listViewModel*. Цей клас буде узагальненим і приймати параметр *T*, де *T* це *BaseViewModel*. Клас має дві властивості – *int Count4* та *IEnumerable<T> Values*.

В архітектурі *ASP.NET WebApi* важливою ланкою виступають контролери. При отриманні запиту, система маршрутизації обирає для обробки потрібний контролер і передає йому дані запиту. Контролер обробляє ці запити і відправляє назад результат обробки. При використанні контролерів існують деякі умовності, так, клас контролера повинен мати суфікс “*Controller*” та атрибут [Controller] і реалізувати абстрактний базовий клас *ControllerBase*.

Запити бувають різних типів, наприклад *GET* чи *POST*, фреймворк *ASP.NET Core* визначити тип запиту застосувавши відповідний атрибут *[HttpGet], [HttpPost], [HttpPut]* чи *[HttpDelete]*.

Додамо папку *Controllers*. Створимо контролери відповідно до кожної бізнес-сутності, які будуть реалізовувати усі *CRUD* операції.

3.2.2 Створення екранних форм

Фреймворк *Angular* підтримує модульну структуру і дозволяє створювати окремі компоненти, з яких, як з цеглинок, будуть побудовані екранні форми. Такий підхід забезпечує гнучкість і дозволяє повторне використання коду.

Почнемо розробку з створення компонентів головної сторінки. Вгорі буде розташований *header*, який міститиме назву розділу, який буде відкритий в цей момент (*Home, Appointments,* тощо) та посилання на особистий кабінет.

Для цього створимо папку *header,* в якій буде лежати наш компонент, його *CSS*-файл та *View Model*. Фреймворк надає нам зручний інструмент для створення будь-яких сутностей використовуючи вбудовані шаблони, який має назву *Angular CLI*.

Angular CLI – модуль, що реалізує інтерфейс командної строки для створення, розробки та підтримки додатків *Angular*. Щоб скористатися його можливостями, відкриємо термінал, перейдемо в папку *header* та введемо команду

```
ng generate component header --app –flat
```

де *component* – тип створюваної сутності; *header* – назва сутності; *app* – префікс компоненту; *flat* – параметр, що вказує на те, що для сутності не буде створена окрема директорія, а всі файли пов’язанні з нею будуть покладені туди, звідки була викликана команда.

В результаті виконання команди *Angular* згенерує три файли в папці *header*:

– *header.component.html;*

– *header.component.scss;*

– *header.component.ts*.

Файл з розширенням *.html* називається шаблоном компонента, він відповідає за відображення даних і містить *HTML*-розмітку. Для створення шаблонів, окрім стандартних тегів *HTML*, будемо використовувати бібліотеку *Angular Material*, яка надає велику кількість шаблонів готових до використання за можливістю їх модифікації.

Створимо шаблон компоненту з використанням завантаженої бібліотеки. Всі використані компоненти необхідно зареєструвати в файлі *app.module.ts*. Створенні компоненти реєструються додаванням їх до масиву *declarations*, а всі сторонні компоненти, що були використанні додаються до *imports* (рис.3.10).

```
@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
    SidenavMenuComponent,
    DoctorModalComponent,
    RoomModalComponent,
  ],
  imports: [
    HomeModule,
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    MatSnackBarModule,
    FormsModule,
    AuthModule,
    AuthRoutingModule,
    ReactiveFormsModule,
    BrowserAnimationsModule,
    MaterialModule,
    AppointmentModule,
    DoctorModule,
    RoomsModule,
    PatientModule,
    UserModule,
    ChatModule,
    UserRoutingModule,
  ],
```

Рис. 3.10 – Реєстрація компонентів в файлі *app.module.ts*

Файл *header.component.scss* містить визначення стилів компоненту. Важливо пам’ятати, що завдяки ієрархічній структурі шаблону *HTML*, стилі батьківських елементів також будуть застосовуватися до дочірніх класів. Файл *header.component.ts* називають вью-моделлю компонента. Тут міститься код *TypeScript*, який визначає поля та методи і містить логіку відображення даних. Для того, щоб ми побачили створений компонент, його потрібно помістити на головну сторінку в файл *app.component.html*, що є батьківським для всіх

елементів *DOM* (*Document Object Model*), які будуть відображенні. Таким чином, додамо компоненти бокового меню, а на головний екран виведемо зображення з інструкцією по користуванню сервісом.

Елементи меню є інтерактивними і мають підсвітку при наведені на них мишею. В правому верхньому куті відображається електронна пошта користувача та іконка, при натисканні на яку, відкривається вкладене меню (рис.3.11).

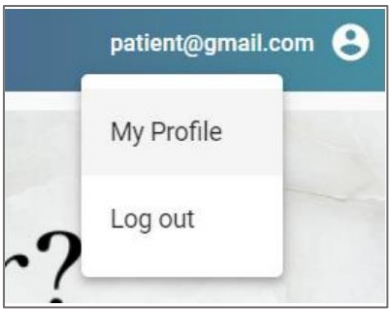


Рис. 3.11 – Меню сайту

Наступним етапом буде створення сторінок авторизації та реєстрації. Так як, неавторизованому користувачу недоступні можливості сайту, на боковому меню він буде бачити лише вкладку *Home*, а в хедері буде розташована назва організації.

Для авторизації потрібно ввести свою електронну пошту та пароль і натиснути на кнопку "*Log in*" (рис. 3.12).

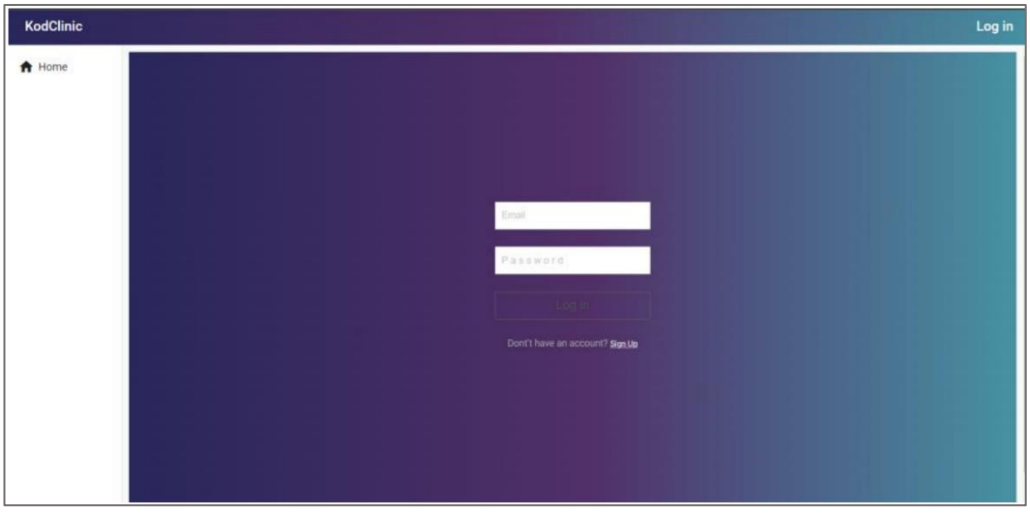


Рис. 3.12 – Форма авторизації

В полях для введення даних, розміщений плейсхолдер, який служить підказкою для користувача. Кнопка ”*Log in*” підсвічується при наведенні миші, і є неактивною, поки поля пусті. Нижче розташований напис, що є підказкою для незареєстрованих користувачів і містить посилання на сторінку реєстрації. Додамо форму подібну для реєстрації (рис. 3.13), що містить поля для введення даних та кнопку підтвердження.

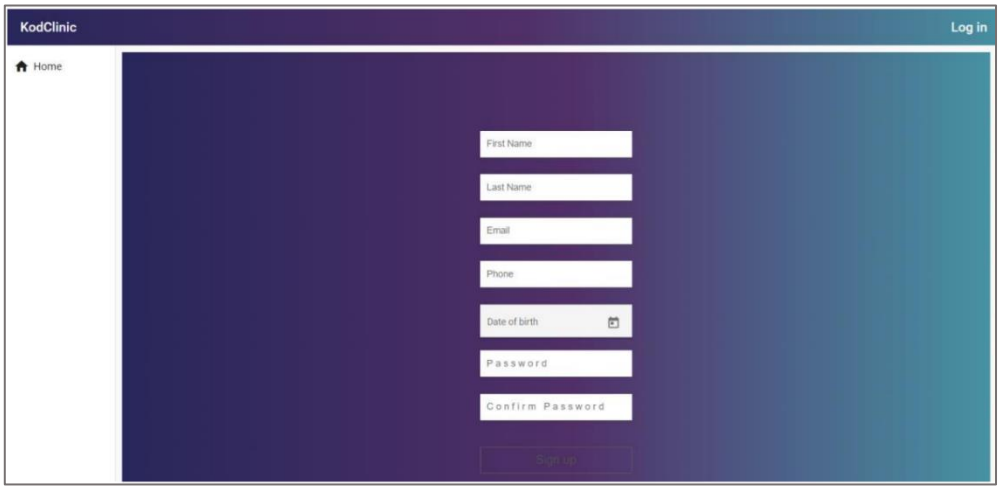


Рис. 3.13 – Форма реєстрації

Для вибору дати народження використано компонент *DatePicker* з бібліотеки *Material*. Даний компонент відображає календар і дозволяє переміщатися між роками та місяцями (рис. 3.14).

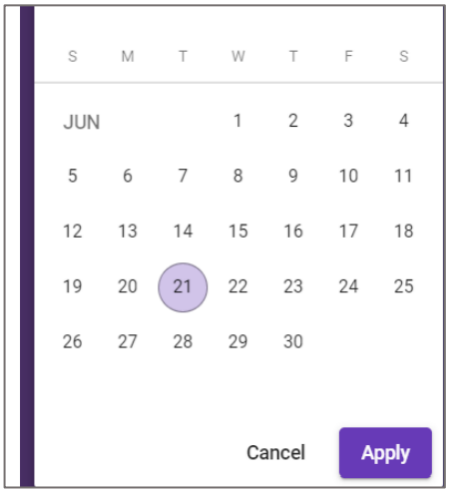


Рис. 3.14 – Компонент *DatePicker*

Перейдемо до створення сторінок з інформацією про учасників та організаторів зустрічей. Дана інформація буде відображена у вигляді таблиці. Таблиця співробітників буде містити інформацію про імена, прізвища та електронну пошту (рис. 3.15).

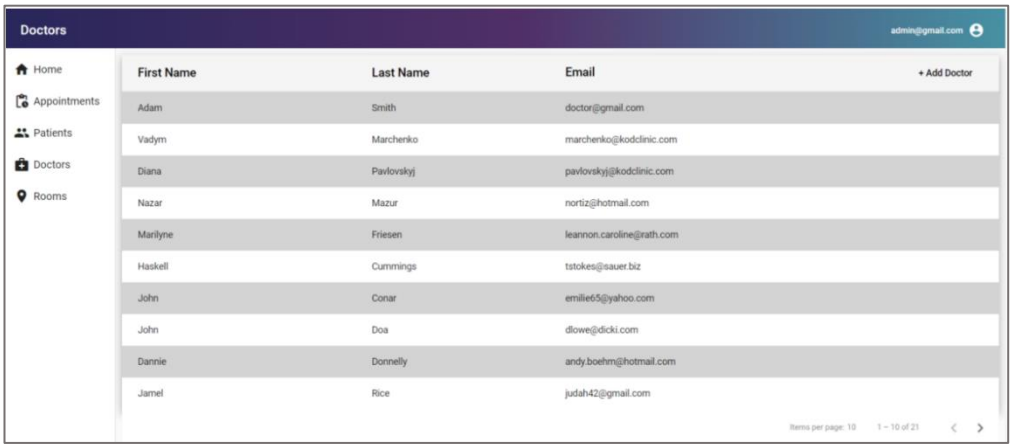


Рис. 3.15 – Вигляд сторінки Співробітники

В таблиці одночасно виводяться дані про десять співробітників, в нижньому правому куті можна побачити інформацію про кількість сторінок у таблиці, а також натиснувши на стрілочки можна переміщатись між сторінками. В правому верхньому куті наявна кнопка “Add Employee”, яка в подальшому буде використовуватись адміністратором. При натисканні на співробітника зі списку, з’являється модальне вікно, в якому можна побачити більш детальну інформацію.

Всі поля доступні для редагування. Тут також розташовані кнопки “*Close*”, “*Delete*” та “*Save*”. Аналогічно створимо сторінки для клієнтів.

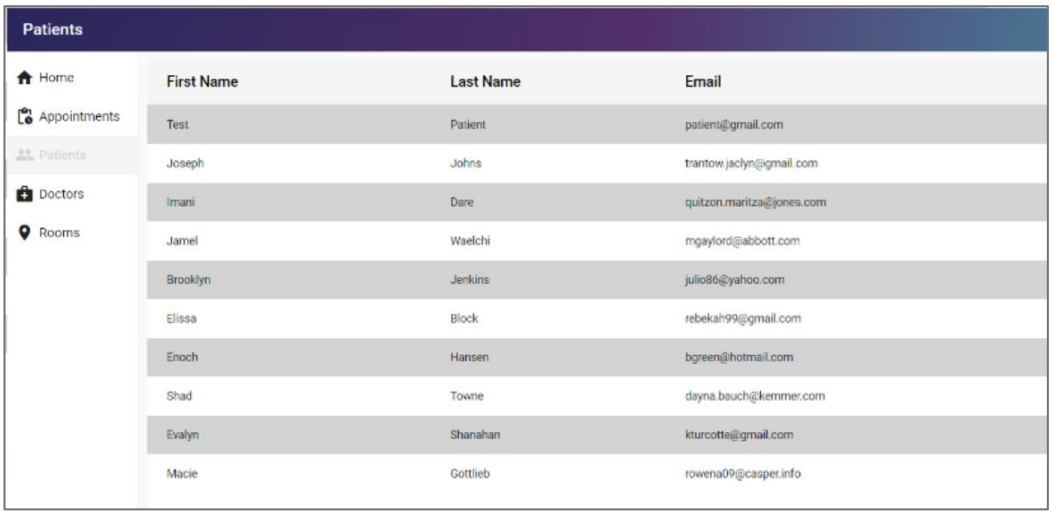


Рис. 3.16 – Модальне вікно клієнта

На сайті передбачена наявність особистого кабінету користувача, де буде відображатися його особиста інформація та фото. Зовнішній вигляд сторінки можна побачити на рисунку 3.17.

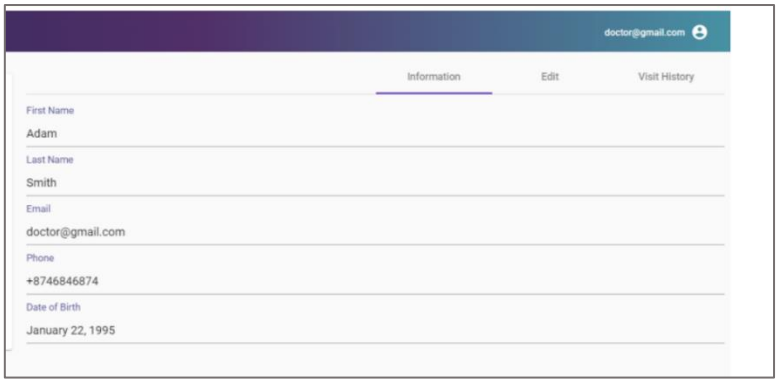


Рис. 3.17 – Сторінка «Особистий кабінет»

Наступним кроком буде створення мессенджера. Для цього спершу додамо іконку, по натисканню на яку, в подальшому, буде відкриватися віконце повідомлень. Іконка буде розташована в правому нижньому куті і буде доступна з будь-якої сторінки сайту. Сам чат уявляє собою вікно, що розташоване в правому нижньому куті, має поле для введення тексту, кнопку “Send”, випадний список для вибору співрозмовника та поле для відображення повідомлень (рис. 3.18).

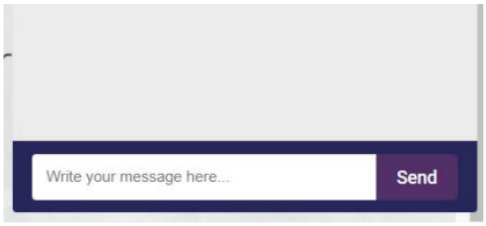


Рис. 3.18 – Вікно повідомлень

Останнім кроком буде створення сторінки "*Appointments*", на якій буде реалізована основна функціональність сайту – запис на зустріч. Сторінка містить календар, де буде відображатися розклад можливих зустрічей (рис.3.19).

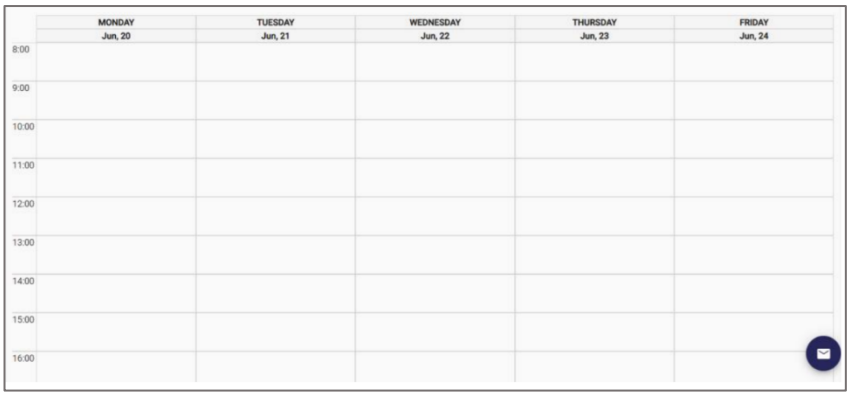


Рис. 3.19 – Сторінка "Appointments"

В лівому верхньому куті розташований селектор для вибору дати та випадні списки для вибору тематики. Оформлення було вирішено зробити мінімалістичним, натомість використати акцентні кольори синьо-блакитної гами, що будуть контрастувати з білим. Мінімалістичні іконки бібліотеки *Material* як найкраще підходять до даного рішення.

3.2.3 Розробка динамічної частини системи

На рівні фронтенду відбувається оперування клієнтськими моделями, які визначають які саме дані необхідно відображати. Тож першим кроком буде створення всіх необхідних моделей:

– *loginModel;*

– *registrationModel;*

– *klientModel;*

– *sotrModel;*

– *appointmentModel;*

– *chatUserModel;*

– *messageModel;*

Для відображення даних *Angular* надає нам механізм зв'язування даних (*data-binding*). *Binding* дозволяє нам в режимі реального часу з'єднувати елементи графічного інтерфейсу з *HTML*-шаблону та вью-модель. Для того, щоб зв'язати поле моделі та елемент *HTML*, визначимо в класі *appointment-tile.component.ts* (компонент, що відповідає за відображення активного запису в календарі) поле *employee* типу *Employee*. Тепер в файлі *appointment-tile.component.html* ми можемо звертатися до цього поля, помістивши його ім'я в подвійні фігурні дужки:

```
<div> Employee: {{employee?.firstName}} {{employee?.lastName}}</div>
```

В цьому випадку ми отримуємо ім'я та прізвище співробітника і кожен раз, коли це значення буде змінюватись в моделі, зміна відбудеться і на графічному інтерфейсі.

Використовуючи можливості *Angular* зв'яжемо створені на попередніх кроках моделі та *HTML*-шаблони, визначимо методи, що будуть обробляти події натискання на кнопки та введення інформації. Тепер, коли шаблони відображають стан моделей, а кнопки мають відповідні обробники, необхідно зв'язати сторінки між собою. Так як ми маємо багато компонентів (сторінок), необхідно забезпечити коректний перехід з однієї сторінки на іншу. Для цього існує механізм маршрутизації (*routing*). Спершу, для кожної сторінки, має бути створений файл *routingmodule.ts*. Зробити це можна за допомогою *Angular CLI* командою: *ng new routing-app --routing –defaults*

Згенерований клас містить визначення масиву *Routes*. Щоб зареєструвати маршрут, до цього масиву треба додати об'єкт типу *Route*, який містить шлях та посилання на компонент. Наприклад, для сторінки *Appointments* (Листінг 3.5).

Листінг 3.5. Додавання об’єкт типу *Route* до масиву

```
const routes: Routes = [
{
path: 'appointments',
component: AppointmentViewComponent,
 }
];
```

Тепер, щоб додати посилання на сторінку *Appointments*, до структури *DOM* необхідно додати елемент *<router-outlet>*, який сповіщає *Angular* про необхідність оновити сторінку для обраного шляху, та додати атрибут *routerLink* на елемент, що буде посиланням на сторінку.

Листінг 3.6. Додавання посилання на сторінку *Appointments*

```
<a *ngIf="user?.hasRole([adminRole, SotrRole, KlientRole])" mat-list-
item
routerLink="/appointments">
<mat-icon>pending_actions</mat-icon><span
class="navcaption">Appointments</span>
</a>
```

Таким чином потрібно налаштувати *routing* для всіх сторінок. Серверний та клієнтські застосунки будуть спілкуватися між собою за допомогою *HTTP* протоколу. Для цього в *Angular* визначений клас *HttpClient*, в якому реалізовані методи *get, post, put* та *delete*. Останнім кроком буде створення сервісів для відправлення запитів на сервер.

Базовий варіант побудови системи передбачає наявність адміністратора, до обов’язків якого входить:

– облік (створення/видалення/редагування) співробітників;

– облік клієнтів;

– облік зустрічей.

Обмеження прав доступу може бути реалізоване засобами *ASP.NET Identity* за допомогою створення ролей. Ролі дозволяють створювати групи користувачів з певними правами та залежно від приналежності до тієї чи іншої групи, розмежувати доступ до ресурсів програми.

Щоб користуватися можливостями ролей, необхідно зареєструвати служби авторизації в класі *Startup.cs*. Зробити це можна викликавши метод *AddRoles<IdentityRole>().* Цей метод визначений в просторі імен *Microsoft.AspNetCore.Identity*. Він створює новий claim типу *Role* і додає його в базу даних *Identity*. Тепер ми можемо виконувати перевірки на основі ролей. Для цього на контролери додаються атрибути, що дозволяють виконання операції лише за наявності зазначеної ролі.

Листінг 3.6. Додавання атрибутів на контролері для окремих ролей

```
[CustomAuthorize(UserRoles = new[] { UserRole.Admin })]
[HttpDelete("{id}")]
public async Task<IActionResult> DeleteDoctorAsync(int id,
CancellationToken token)
{
 Doctor doctor = await doctorService.GetByIdAsync(id, token);
 if (doctor == null)
 {
 return new NotFoundResult();
 }
 return new OkObjectResult(await doctorService.DeleteAsync(doctor,
token));
}
```

Додамо атрибути перевірки ролей на всі методи контролерів відповідно до прав, якими володіють користувачі кожної групи. Додамо перевірки ролей і в клієнтській частині застосунку. Для цього створимо клас *UserRoleGuard*, він буде отримувати поточного користувача, і викликати в нього метод *hasRole(userRoles: Arrau<string>)*, який виглядає наступним чином:

```
hasRole(userRoles: Array<string>): boolean {
let hasRole: boolean = false;
userRoles.forEach((role: string) => {
if (this.roles.indexOf(role) >= 0) {
hasRole = true;
}
});
return hasRole;
}
```

3.3 Тестування

Тестування є важливою частиною розробки програмного забезпечення, при наявності якісного тестування можна уникнути несподіваної поведінки та

виявити слабкі місця системи. Крім того, якщо користувач часто буде стикатися з дефектами, він може взагалі перестати користуватися додатком.

Тестування поділяється на:

− мануальне (ручне);

− автоматичне.

Для тестування серверного додатку пишуться автоматичні *unit*-тести з використанням фреймворку *NUnit*. *Unit*-тести дозволяють протестувати окремі модулі системи. Їх використання перевіряє роботу маленьких модулів, дозволяючи тим самим швидко локалізувати проблему, якщо така виникає.

Автоматичні тести створюються розробниками і їх рекомендується виконувати при кожній зміні в системі, щоб швидко перевірити, чи не спричинили нові зміни появу помилок у вже протестованій частині додатку.

Для тестування фронтенду були використані методи мануального тестування. Мануальне тестування виконується тестувальником без використання програмних засобів шляхом моделювання дій користувача. Перш за все, потрібно відкрити всі сторінки та перевірити, чи все відображається коректно:

— розмітка не має дефектів (вирівнювання дотримано, елементи на своєму місці);

— ми бачимо ті дані, які мають бути відображені (ім’я користувача присутнє, прізвище присутнє, тощо);

— ми бачимо їх там, де вони мають бути (в графі «ім’я» присутнє саме ім’я, а не номер телефону, наприклад);

— ми не бачимо тих даних, які не маємо бачити (клієнту не доступна інформація про додавання нових зустрічей).

Наступним кроком буде перевірка роботи кнопок та посилань:

— при наведені миші елемент підсвічується/не підсвічується;

— кнопка активована/дезактивована;

— чи активується кнопка, якщо ввести дані в поле;

— при натисканні на кнопку відправляється певний запит на сервер (це можна перевірити відкривши в браузері інструменти розробника на вкладці “Network”);

— при натисканні на кнопку відбувається відповідна дія;

— при натисканні на посилання відбувається перехід на потрібну сторінку.

Під час тестування веб-сервісу була проведена перевірена перевірка на коректність його роботи. Здійснювалися такі перевірки:

1. Валідація поля на порожнечу. Якщо поле не заповнено – показує помилку після відправки форми.

2. Перевірка на захищеність засобів авторизації.

3. Перевірка кросбраузерності веб-сайту. Було перевірено у браузерах: *Chrome, Firefox, Safari*. Для тестування кросбраузерності використовувався такий інструмент, як *BrowserStack*, який дає змогу запускати кросбраузерні тести на 3000+ комбінаціях браузерів на реальних пристроях.

Висновок до третього розділу

У третьому розділі було розроблено серверну частину застосунку відповідно до обраної архітектури. Було створено рішення, яке містить рівні домену, доступу до даних, сервісів та інтерфейсу користувача.

Побудовано графічний інтерфейс користувача та написано необхідні скрипти для взаємодії з користувачем та коректного відображення даних. Реалізовано систему ролей та розмежування доступу до даних відповідно до прав користувача.

В даному розділі було протестовано клієнтську частину застосунку. Проведено мануальне тестування інтерфейсу користувача та виконано запуск додатку у найбільш популярних браузерах. Як результат, маємо робочий демонстраційний веб-застосунок, що виконує функції, що були передбачені завданням.

РОЗДІЛ 4

ЕКОНОМІЧНІ РОЗРАХУНКИ

4.1 Організаційно-економічне обґрунтування роботи

4.1.1 Актуальність та результати проєкта

При виконанні дипломної роботи було створено веб-сайт онлайн бронювання зустрічей за рахунок використання нереляційної бази даних *MongoDB* та сучасної системи *Angular framework*, яка надає можливість зручного кодування та спрощеного модульного тестування..

По масштабу проект, що розробляється, відноситься до середніх проектів, які невеликі по масштабу, прості і обмежені об'ємами. По термінах реалізації проект короткостроковий (термін реалізації до 3 років). По складності проект, що розробляється, є проектом середньої складності. По характеру цільового завдання – комбінований проект. По характеру проекту – галузевий проект.

4.1.2 Огляд аналогів і конкурентів

Відмова від листування поштою та перехід на онлайн системи бронювання зустрічей сприятливо позначається на бізнесі, тому подібні послуги почали зростати (у кількісному відношенні) як гриби. Ринок онлайн-планувальників – висококонкурентне середовище. Зараз використовують програми *Calendly, Integrica, ScheduleOnce, SimplyBook* та десятки їх аналогів.

Всі вони виконують однакові завдання, але відрізняються інтегрованими сервісами, деякими параметрами для календарів, а також мовами, що підтримуються, і вартістю послуг, що надаються. В Україні є багато сервісів онлайн-запису з різним функціоналом.

Calendly – один із найпопулярніших сервісів. У *Calendly* інтуїтивно зрозумілий інтерфейс із симпатичним оформленням, в яке можна легко вписати власний логотип, корпоративні кольори тощо.

У *Calendly* є весь базовий набір опцій, типових для систем бронювання: обмеження кількості вільних часових осередків, оповіщення про нові записи поштою та push-сповіщення. Але найголовніше – величезна кількість інтеграцій із сторонніми службами та програмами, включаючи *Teams, Zoom, iCloud, Google Calendar, Slack, PayPal* та інші.

EasyWeek надає онлайн-платформу, на якій клієнти компанії можуть переглянути доступні товари або послуги, забронювати час і здійснити оплату. EasyWeek є комплексним рішенням, яке пропонує функції онлайн-бронювання і є ефективною заміною звичного запису клієнтів через *Google*-таблиці або паперовий зошит. Сервіс бронювання зручний для співробітників та власників компанії.

Вартість місячної підписки залежить від кількості локацій та співробітників. Для малого бізнесу (1 локація та 1 працівник) діє безкоштовний базовий тариф. Починаючи з 2 користувачів діятиме тариф 21 євро або більше відповідно до кількості персоналу. Також бувають знижки та акції.

Altegio – ще один популярний сервіс онлайн-запису на українському ринку. *Altegio* надає базові можливості онлайн-запису, маркетингу та бізнес-аналітики. Це програмне забезпечення для онлайн-бронювання та планування зустрічей для підприємств, що надають різні види послуг.

Включає наступні сервіси:

1. Бронювання онлайн. Можливість отримувати записи цілодобово та без вихідних через *URL*-адресу бронювання та безпечно отримуйте оплату.

2. Призначення зустрічей. Керування онлайн і офлайн зустрічами: планування, перенос або скасовування їх у кілька кліків. Перегляд власних зустрічей чи зустрічей співробітників.

3. Сповіщення та нагадування. За допомогою сервісу *SMS*-повідомлень та листів електронної пошти збільшується кількість зустрічей і виключаються запізнення. За допомогою функції сегментації клієнтів можна групувати клієнтів (наприклад, за датою останнього відвідування) та надсилати їм спеціальні пропозиції, щоб повернути їх.

4. Клієнтська база даних. Можливість вводу та відстеження даних всіх клієнтів: історія зустрічей, відсутність виступів та статус членства.

5. Ціновий діапазон користування починається з 26 євро за місячну підписку, за умови тарифу на 3 місяці. Так, мінімальний платіж становитиме 80 євро одразу за 3 місяці.

Integrica користується популярністю серед салонів краси та надає функції онлайн-бронювання послуг, базові маркетингові інструменти та аналітику.

Ключові можливості:

1. Календар. У календарі одразу можна бачити, чи є в майстрів вільні години, чи вони вже з головою в роботі. Події в календарі не губляться, і керівництво може аналізувати, чи настав час, наприклад, розширювати команду.

2. Звіти за записами. Дають можливість оцінювати завантаженість майстрів і прибуток за місяць. В онлайн режимі можна переглядати, який прибуток генерують майстри щодня.

3. Онлайн-запис. Не потрібно записувати кожного клієнта вручну, вони можуть записуватися самі в чат-боті чи на сайті. Записи синхронізуються із календарем і керівництво легко може стежити за щоденним розкладом.

4. База клієнтів. Можливість зберігати історію кожного клієнта й використовувати загальну базу для розсилання.

5. Вартість тарифу *Light* складає 25 євро, тариф *Ultimate* коштуватиме 100 євро. Різниця у швидкості технічної підтримки та повноті функціоналу.

CleverBox CRM – український онлайн-сервіс для управління бізнес-процесами організацій у сфері краси та здоров'я. Система пропонує онлайн-запис, базовий фінансовий облік, аналітику та маркетинг. На сьогоднішній день системи використовують понад 500 клієнтів [1].

Функціонал та переваги:

1. Клієнтська база. Збір основної інформації клієнта: ім'я, електронна пошта, номер телефону, дата народження, фінансова інформація, придбані послуги, записи дзвінків, історія sms-повідомлень.

2. Маркетинг. Онлайн-залучення клієнтів. Реклама в інтернеті, соціальних мережах із прямим посиланням на онлайн-запис.

3. Телефонія. Можливість здійснювати дзвінки з картки клієнта. Збереження запису телефонних дзвінків.

4. Запис клієнта. Віджет онлайн-запису. Запис 24/7. Автоматичне узгодження запису у режимі реального часу. Виняток повторного часу. Весь час запису займає трохи більше за хвилину. Інтеграція з електронним журналом адміністратора та спеціаліста надання послуг. Оперативне редагування візиту за будь-якими необхідними параметрами: часом, датою, послугою, майстром. Залишити заявку на підбір запису на зручний час. Вибір зручного способу зворотний зв'язок (*Viber, Telegram*).

5. Вартість сервісу залежить від кількості працівників. Так, тариф на 1 – 3 працівників коштуватиме 45 євро, а максимальний, на 11 – 30 працівників, – 88 євро.

Appointer. Цей планувальник пропонує онлайн-запис на прийом, аналітичні звіти та простий маркетинг.

Переваги використання *CRM*-системи *Appointer* для малого бізнесу:

1. Інтеграція з *IP*-телефонією. Легко інтегрується з *IP*-телефонією від провайдерів. Адміністратор бачить ім'я абонента при вхідному дзвінку або створює нового клієнта.

2. Просте перенесення даних з інших *CRM*. Можливість переносити облікові дані клієнтів з інших програм в *Appointer*, швидко і без втрат.

3. Робота з телефону, планшета, будь-якого браузера. *CRM* не потрібно завантажувати і встановлювати. Для доступу необхідні лише інтернет і браузер, або планшет чи смартфон.

4. *SMS*-повідомлення про записи. Автоматична розсилка *sms*-нагадувань про візит клієнтам. Адміністратор самостійно вибирає частоту відправки нагадувань і редагує їх тексти.

5. Онлайн-запис через сайт і соцмережі. Клієнт самостійно записується на прийом через сайт або іншу соцмережу, тим самим знижуючи навантаження на адміністрацію.

6. Ціни стартують від 20 євро за місяць з можливістю оплати лише за 3, 6 або 12 місяців одразу. Мінімальний платіж становить 60 євро за 3 місяці.

4.1.3 Порівняння елементів систем бронювання в існуючих аналогах

Таблиця 4.1

Порівняння компонентів в існуючих аналогах

	Calendly	*EasyWeek*	*Altegio*	*Integrica*	*CleverBox*	*Appointer*
Наявність календаря	+	+	+	+	-	-
Онлайн запис на зустріч	+	+	+	+	+	+
Можливість вести клієнтську базу	-	+	+	+	+	+
Повідомлення про записи	+	+	+	+	+	+
Звіти за записами / ведення статистики	+	+	+	+	+	+
Можливість перенесення даних з інших *CRM*	-	-	-	-	+	+
Платна версія	+	+	+	+	+	+
Можливість роботи з різних пристроїв	+	+	+	+	-	+
Можливість інтеграцій із сторонніми службами та програмами	+	-	-	-	-	+

4.2 Визначення мети і результатів роботи

Метою роботи є проектування і розробка веб-сайту для онлайн бронювання зустрічей для організації, яка займається соціалізацією молоді.

Об’єктом дослідження є процеси проектування і розробки веб-сайту для онлайн бронювання різного плану зустрічей (тренингів, мастер-класів, учбових занять).

Предметом дослідження є методи проектування і розробки веб-сайту для онлайн бронювання.

Методи дослідження. В ході виконання роботи можуть бути використані такі методи як аналіз, абстрагування, узагальнення, аналогія, класифікація.

Основними задачами, які необхідно вирішити в ході роботи, є:

1. Аналіз предметної області, дослідження існуючих аналогів та постановка задачі.

2. Проектування бази даних та системи бронювання.

3. Розробка демонстраційної версії сайту онлайн бронювання зустрічей.

Таблиця 4.2

Етапи і стадії роботи

№	Стадії	Зміст	Тривалість виконання
1	Технічне завдання	Дослідженні предметної області. Дослідження існуючих аналогів	Січень
2	Технічний проект	Проектування структури сайту.	Лютий
3	Робочий проект	Програмна реалізація сайту. Наповнення контентом.	Березень - квітень
4	Впровадження	Налагодження та виправлення помилок (bugs). Тестування. Підготовка підсумкових документів	Травень

Таблиця 4.3

Склад робіт по життєвому циклу проекту

№ робіт	Назва робіт	Тривалість робіт,дні
0-1	Вивчення предметної області	15
1-2	Вибір мови програмування та середовища розробки	5
2-3	Планування строків виконання необхідних робіт	8
3-4	Концептуальне проектування	10
4-5	Проектування БД	8
5-6	Створення БД	5
6-8	Створення структури сайту	8
5-7	Створення алгоритму роботи інтерфейсу	8
7-8	Реалізація користувацького інтерфейсу	6
6-8	Створення тестової версії	8
8-9	Тестування проекту в роботі	6
9-10	Виправлення помилок	2
10-11	Підготовка супровідної документації	2
11-12	Оцінка результатів проекту і підведення підсумків	2
12-13	Підготовка підсумкових документів і закриття проекту	2

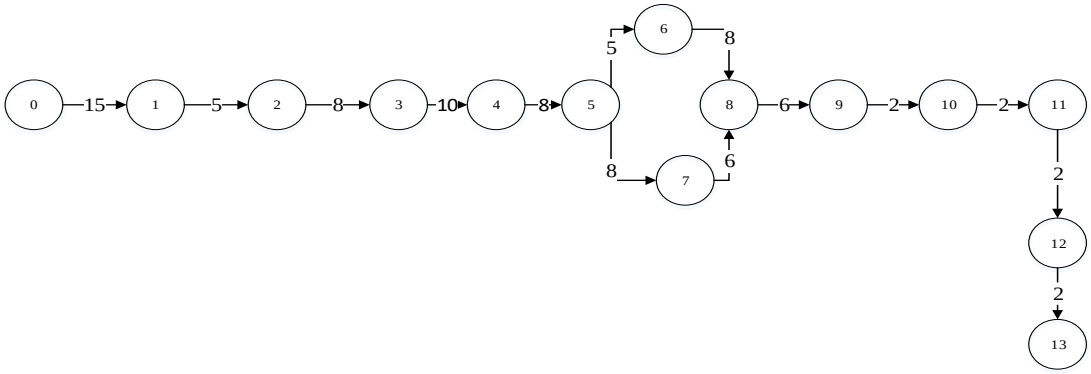


Рис. 4.1 – Мережевий графік проєкту

Розрахунок параметрів мережевого графіка:

T_{ij}– тривалість робіт;

$T_{рн}$ – ранній початок робіт;

$T_{ро}$ – раннє закінчення робіт;

$T_{пн}$ – пізній початок робіт;

$T_{по}$ – пізнє закінчення робіт;

R_j – резерв часу.

Таблиця 4.4

Розрахунок параметрів мережевого графіку

Назва роботи	Попередня	T_{ij}	$T_{рн}$	$T_{ро}$	$T_{пн}$	$T_{по}$	R_j
0-1	————	15	0	15	5	5	0
1-2	0-1	5	15	20	23	28	8
2-3	1-2	8	20	28	28	36	8
3-4	2-3	10	28	38	36	46	8
4-5	3-4	8	38	46	38	46	0
5-6	4-5	5	46	51	47	52	1
6-8	5-6	8	51	59	52	60	1
5-7	4-5	8	46	54	46	54	0
7-8	5-7	6	54	60	54	60	0
8-9	6-8; 7-8	6	60	66	60	66	0
9-10	8-9	2	66	68	66	68	0
10-11	9-10	2	68	70	68	70	0
11-12	10-11	2	70	72	70	72	0
12-13	11-12	2	72	74	72	74	0

4.3 Розрахунки ціни програмного продукту

Як вихідні дані для визначення трудомісткості розробки ПП використовується типовий склад етапів і укрупнені норми часу на розробку програмних засобів (ПЗ).

Розроблювальному ПП відповідає програма-анало*г Integrica* з обсягом 6000 умовних машинних команд і трудомісткістю $T_р = 210$ люд.-год.

Трудомісткість розробки ПП включає розробку наступних етапів:

технічного завдання – ТЗ;

технічного проекту – ТП;

робочого проекту – РП;

впровадження – ВН.

Трудомісткість розроблювального ПП визначається по кожному етапу окремо на підставі трудомісткості аналога з урахуванням складності розробки, ступеня новизни і ступеня використання в розробці стандартних модулів на підставі формул (4.1) – (4.4).

$$T_{тз} = T_р \cdot L_1 \cdot К_н; \quad (4.1)$$

$$T_{тп} = T_р \cdot L_2 \cdot К_н; \quad (4.2)$$

$$T_{рп} = T_р \cdot L_3 \cdot К_н \cdot К_т; \quad (4.3)$$

$$T_{вн} = T_р \cdot L_4 \cdot К_н; \quad (4.4)$$

де $T_р$– укрупнена норма часу на розробку аналога ПП, люд.-годин, що коректуються поправочним коефіцієнтом, який враховує умови розробки ПП.

$К_к = 0,8;$

$T_р = 210 \cdot 0,8 = 168$ люд.-год.

Даний проект має доступні аналоги, тому його можна віднести до ступеня новизни: В.

L_j– питома вага і-го етапу розробки в залежності від ступеня новизни:

$L_1 = 0,12;$

$L_2 = 0,11;$

$L_3 = 0,61;$

$L_4 = 0,16.$

$К_н$– поправочний коефіцієнт, який враховує ступінь новизни (0,7);

$К_т$ – поправочний коефіцієнт, який враховує ступінь використання програм в розробці (0,7);

Тоді: $T_{тз} = 168 \cdot 0,12 \cdot 0,7 = 14,11$(дні)

$T_{тп} = 168 \cdot 0,11 \cdot 0,7 = 12,93$(дні)

$T_{рп} = 168 \cdot 0,61 \cdot 0,7 \cdot 0,7 = 50,21$(дні)

$T_{вн} = 168 \cdot 0,16 \cdot 0,7 = 18,81$(дні)

Тривалість розробки ПП у літах визначається за формулою (4.5).

$$T_{\text{ПП}} = T_{\text{тз}} + T_{\text{тп}} + T_{\text{рп}} + T_{\text{вн}} \quad (4.5)$$

де $T_{\text{ПП}}$ – сумарна тривалість розробки.

$$T_{пп} = 14{,}11 + 12{,}93 + 50{,}21 + 18{,}81 = 96 \text{ (дні)} = 0{,}26 \text{ (рік)}$$

За час розробки візьмемо середнє значення розрахованих показників тривалості проектування та розробки ПП:

$$T_{пп\text{сер}} = \frac{74 + 96}{2} = 85 \text{ (дні)}$$

Таблиця 4.5

Розрахунок витрат на матеріали

Найменування матеріальних витрат	Од. вим.	Кількість	Ціна за одиницю, грн.	Вартість, грн.
Диски	шт.	2	10	20
Папір	лист	200	0,24	48
Фарба (картридж)	шт.	1	58	58
Флеш-карта, 4 Gb	шт.	1	130	130
Разом				256
Транспортно-заготівельні витрати – 10%				256·0,10=25,6
Усього:				281

Таблиця 4.6

Розрахунок основної заробітної плати

Найменування робіт	Трудомісткість робіт у днях	Місячний оклад	Денна заробітна плата	Заробітна плата
1	2	3	4	5
Розробка ПП	85	6500	295	25075
Контроль керівника	20	10000	455	9100
Усього:	33	–	–	34175

Додаткова заробітна плата враховує оплату чергових відпусток, премії, інші доплати. Приймається в розрахунках 10% від основної (4.6).

$$ЗП_{дод} = ЗП_{осн} \cdot 0,10 \quad (4.6)$$

$$ЗП_{дод} = 34175 \cdot 0,10 = 3418 \text{ грн.}$$

Єдиний соціальний внесок (Єсв) приймаються в розмірі 22% від суми основної і додаткової заробітної плати (4.7).

$$Єсв = (ЗП_{осн} + ЗП_{доп}) \cdot 0,22 \quad (4.7)$$

$$Єсв = (34175+3418) \cdot 0,22 = 8270 \text{ грн.}$$

Витрати, зв'язані з використанням обчислювальної техніки, визначаються за (4.8).

$$С_{еом} = t^{еом} \cdot К^{еом}_{в} \cdot Ц^{еом} \cdot К^{еом}_{е}; \quad (4.8)$$

де $Т^{евм}$– час використання ЕОМ для розробки даного ПП, години;

$Т^{евм}$– 181годин.

$К^{евм}$– поправочний коефіцієнт обліку часу використання ЕОМ (1,08);

$Ц^{евм}$– ціна однієї години роботи на ЕОМ, грн. (5 грн.);

$К^{евм}$– коефіцієнт обліку швидкодії ЕОМ (1,0 – швидкодія ЕОМ більш 20х10 опер/з; 1,2 – швидкодія ЕОМ менш 20х10 опер/з,).

$$С_{еом} = 181 \cdot 1,08 \cdot 5 \cdot 1,0 = 977 \text{ грн.}$$

Накладні витрати враховують адміністративні, загальновиробничі витрати, витрати на збут. Приймаються в розмірі 50% від основної заробітної плати (4.9).

$$Н_{в} = 0,30 \cdot ЗП_{осн} \quad (4.9)$$

$$Н_в = 0,50 \cdot 34175 = 17088 \text{ грн.}$$

На підставі здійснених розрахунків складається калькуляція планової собівартості ПП.

Таблиця 4.7

Калькуляція собівартості ПП

Найменування статей Витрат	Сума витрат (грн.)	Питома вага, %
1. Матеріали	281	0,4
2. Основна заробітна плата	34175	53
3. Додаткова заробітна плата	3418	5
4. Єдиний соціальний внесок	8270	13
5. Витрати, зв'язані з обчислювальною технікою	977	1,6
6. Накладні витрати	17088	27
Разом:	64209	100

Ціна ПП визначається по формулі (4.10).

$$Ц = С + П_р, \quad (4.10)$$

де С – витрати на розробку програмної продукції (планова собівартість), грн.

$П_р$– розмір прибутку, розрахований по формулі (4.11).

$$П_р = С \cdot \%Р_н / 100; \quad (4.11)$$

де $Р_н$– плановий рівень рентабельності (25%);

Прибуток у ціні ПП становить:

$$П = (64209-281) \cdot 0,25 = 15982 \text{ грн}$$

Ціна ПП становить: Ц = 64209*1,1+15982 = 86612 грн.

4.4 Розрахунок капітальних витрат

Для розрахунку економічної ефективності проекту визначаються капітальні і поточні витрати, зв'язані з використанням програмного продукту.

Розрахунок капітальних витрат, зв'язаних із впровадженням програмного продукту здійснюється по формулі (4.12).

$$К_2= К_{пп} +К_п+ К_{ко}+ К_{во} \quad (4.12)$$

де $К_{пп}$ – ціна програмного продукту;

$К_п$–попередвиробничі витрати;

$К_{ко}$– вартість комп'ютерного устаткування;

$К_{во}$ – вартість допоміжного устаткування, необхідного для надійної роботи грн.

Ціна програмного продукту складає 82640грн.

Попередвиробничі витрати містять у собі усі витрати, зв'язані з налагодженням і впровадженням гри – постановка задач та їхня алгоритмізація, розробка, налагодження і впровадження програмного продукту (ПП).

Приймаються $К_п$ у розмірі 100% від вартості розробленого ПП.

$$К_п= 86612 \text{ грн.}$$

Вартість комп'ютера ($К_{ко}$) становить 15000 грн.

Вартість допоміжного устаткування визначається укрупнено в розмірі 10% від вартості комп'ютера ().

$$К_{во}= 15000\cdot 0,10 =1500 \text{ грн.}$$

$$К_2 = 86612+ 15000 + 1500 = 103112 \text{ грн.}$$

Капітальні вкладення до впровадження гри булі відсутні,тому $К_1=0$.

Розрахунок поточних (експлуатаційних) витрат, зв'язаних з використанням ПП ($С_і$), здійснюється по формулі (4.13).

$$С_і=С_{опл}+ С_а+ С_{ел}+ С_р+ С_{доп}+ С_п \quad (4.13)$$

де $С_{опл}$– річний фонд основної і додаткової оплати праці персоналу, що обслуговує програмний продукт з нарахуваннями;

$С_а$ – сума річних амортизаційних відрахувань від вартості основного і допоміжного устаткування програмного продукту;

$С_{ел}$– вартість витрат на електроенергію в рік;

$С_р$– вартість річного ремонту основного і допоміжного устаткування;

$С_{доп}$ – річна вартість допоміжних матеріалів, зв'язаних з експлуатацією.

$С_п$ – вартість річного утримання приміщень.

Під час проведення аналізу предметної області за базовий варіант було обрано діяльність компанії без використання спеціалізованої програми. Весь процес контролю виконують 2 співробітника, заробітна плата яких складає по 7800 грн.

Розрахунок річного фонду основної і додаткової оплати праці персоналу з нарахуванням ($С_{опл}$).

Річний фонд основної заробітної плати персоналу (4.14)

$$ЗП_{осн} = \cdot (ЗП_{окі} \cdot Ч_{і},) \cdot 12 \qquad (4.14)$$

де $Ч_і$ – чисельність, фахівців і-тої категорії, що обслуговують ПП;

$ЗП_{окі}$– місячний оклад фахівця і-тої категорії.

$$ЗП_{осн} =(7800 \cdot 2)\cdot 12 = 187200 \text{ грн.}$$

Фонд додаткової заробітної плати (4.15):

$$З_{дод}= ЗП_{осн}\cdot К_{дод} \qquad (4.15)$$

де $К_{дод}$– коефіцієнт додаткової заробітної плати (приймається в розмірі $К_{дод}$ = 0,2)

$$З_{дод}= 187200 \cdot 0{,}2= 37440 \text{ грн.}$$

Єдиний соціальний внесок приймається в розмірі 22% від суми основної і додаткової заробітної плати.

$$Єсв = (ЗП_{осн} + ЗП_{дод}) \cdot 0,22$$

$$Єсв = (187200 + 37440) \cdot 0,22 = 49420 \text{ грн.}$$

Загальні витрати на оплату праці:

$$С_{опл1} = ЗП_{осн} + ЗП_{доп} + Єсв = 187200 + 37440 + 49420 = 274060 \text{ грн.}$$

При відсутності інших витрат до впровадження ПП $С_1$ = 274060грн.

Після впровадження ПП роботу виконує одна людина: фахівець з окладом 7400 грн.

Річний фонд основної заробітної плати фахівця з окладом 7400 грн. становить:

$$ЗП_{осн} = 7400 \cdot 12 = 88800 \text{ грн}$$

Фонд додаткової заробітньої плати:

$$ЗП_{доп} = 88800 \cdot 0,20 = 17760 \text{ грн}$$

Єдиний соціальний внесок:

$$Єсв = (88800 + 17760) * 0,22 = 23443 \text{ грн}$$

Загальні витрати на оплату праці:

$$С_{опл2} = ЗП_{осн} + ЗП_{доп} + Відр.сс = 88800 + 17760 + 23443 = 130003 \text{ грн}$$

Розрахунок амортизаційних відрахувань визначається по формулі (4.16), де $Н_а$ – норма амортизаційних відрахувань(для комп'ютерних устаткувань $Н_а$=50%).

$$С_а = (К_{ко} + К_{во}) \cdot На / 100 \qquad (4.16)$$

$$С_а = (15000 + 1500) \cdot 0,5 = 8250 \text{ грн}$$

Річна вартість споживаної електроенергії $С_{ел}$ визначається по формулі (4.17).

$$С_{ел} = М_у \cdot Т_{ко} \cdot Ц_е \cdot К_и \qquad (4.17)$$

де $М_у$ – установлена сумарна потужність комп'ютерного устаткування,(0,45кВт);

$Т_{ко}$–річний фонд часу роботи ЕОМ, який визначається виходячи з кількості робочих днів в році ($Д_р$), тривалості робочого дня (Т) і з урахуванням часу на профілактичні огляди за рік ($Т_{огл}$).

$$Т_{ко}= Д_р * Т – Т_{огл}$$

$$Т_{ко}= 250 \cdot 8 – 300 = 1700 \text{ год.}$$

$Ц_е$– вартість 1 квт-години ел. енергії (3,22 грн.);

$К_и$ – коефіцієнт інтенсивного використання потужності ($К_{иреком.}$=0,9)

$$С_{ел} = 0,45 \cdot 1700 \cdot 3,22 \cdot 0,9 = 2217 \text{ грн.}$$

Витрати на ремонт ($С_р$) приймаються в розмірі 6% від вартості комп’ютерного устаткування:

$$С_р = 15000 \cdot 0,06 = 900 \text{ грн.}$$

Витрати на допоміжні матеріали ($С_{доп}$) приймаються в розмірі 2% від вартості комп’ютерного устаткування:

$$С_{доп} = 15000 \cdot 0,02= 300 \text{ грн.}$$

Загальні витрати:

$$С_2 =130003+8250+2217+ 900 +300 = 141670 \text{ грн}$$

4.5 Розрахунок показників економічної ефективності проекту

Очікуваний економічний ефект визначається по формулі (4.18).

$$Е_о = (С_1 – С_2) – Е_н \cdot (К_2 – К_1) \qquad (4.18)$$

де $С_1$, $С_2$– поточні витрати відповідно до і після впровадження проекту; $(С_1 – С_2)$ – річна економія на поточних витратах, грн.;

$К_2$ – капітальні витрати на впровадження ПП, грн.;

$К_1$ – капітальні витрати до впровадження ПП, грн.;

$E_н$ – нормативний коефіцієнт ефективності одноразових витрат (рекомендовано $E_н$ =0,25).

$$E_о = (274060–141670) – 0,25 \cdot 103112 = 132390-25778=106612грн$$

Потім розраховується коефіцієнт ефективності капітальних витрат по формулі (4.19).

$$E = (C_1– C_2) /К_2 – К_1 \quad (4.19)$$

$$E = 132390/103112= 1,28$$

Так, як $E>E_н$, то проект ефективний.

Розраховується строк окупності капітальних витрат на впровадження проекту за формулою (4.20).

$$T = 1/E =1/1,28= 0,78 року=9,4міс \quad (4.20)$$

Результати економічних розрахунків відображаються в підсумковій таблиці 4.9.

Таблиця 4.9

Техніко-економічні показники програмного продукту

№	Найменування показників	Одиниця Виміру	Значення показника	
			до впрова-дження проекту	після впро-вадження проекту
1	Трудомісткість розробки проекту	днів		85
2	Ціна ПП	грн.		86612
3	Капітальні витрати	грн.	__	103112
4	Поточні витрати	грн/рік	274060	141670
5	Економічний ефект від реалізації проекту	грн/рік		106612
6	Строк окупності	років		0,78
7	Економічна ефективність			1,28

4.6 Бізнес план стартапу проекту

Таблиця 4.10

Міні-бізнес план стартап-проекту для веб-сайта

Основна ціль проекту	Основною ціллю проекту стартапу розробки сайту для онлайн бронювання є досягнення успіху в комунікації з клієнтами, сайт буде привертати потенційних клієнтів та приносити прибуток.
Аналіз ринку	Основна група цільової аудиторії включає молодь вікової групи з 18 до 45 років, які можуть бути клієнтами організації, що замовляє сайт.
Розробка продукту	Розробка складається з проектування бази даних та системи бронювання і розробки демонстраційної версії сайту онлайн бронювання зустрічей.
Маркетинг та реклама	Розроблення веб-сайту, де керівники можуть знайти інформацію про сайт (веб-додаток), оновлення та можливості спілкування з приводу впровадження. Також створення сторінки у популярних соціальних медіа для активного спілкування з аудиторією. Розроблення трейлеру про сайт для розміщення їх на платформах, таких як YouTube і Twitch, для привертання уваги потенційних клієнтів.
Фінансові доходи	Основним джерелом доходів для більшості сайтів є продаж самого сайту. Крім того, дохід може отримуватися на основі потенційно більшої кількості записів клієнтів і відтак більшої кількості виконаних зустрічей, а також підписок та доходу по вбудованій рекламі.

Висновок до четвертого розділу

Розробка даного проекту є вигідною, бо має швидку окупність і перспективи додаткового прибутку у майбутньому. Розробка сайтів є прибутковим бізнесом, попит на них залишається високим.

Порівняння коефіцієнту ефективності та одноразових витрат дозволяє зробити висновок, що впровадження даного продукту є економічно вигідним, тому розробка цього програмного продукту є актуальною. Трудомісткість проекту становить 85 днів; строк окупності проекту – 0,78 року (9місяців); капітальні витрати становлять 103112 грн. Ціна програмного продукту – 86612 грн.

РОЗДІЛ 5

ОХОРОНА ПРАЦІ

5.1 Загальні вимоги до особистого робочого місця

Сучасний розвиток технічного та технологічного стану виробництва передбачає постійну автоматизацію та оптимізацію виробничих процесів. Сьогодні, напевно, важко уявити компанію, господарська діяльність в якій здійснювалась би без використання комп'ютерної техніки. Через масовий характер робіт, що виконуються працівниками за допомогою комп'ютера, законодавством України чітко врегульовано норми та вимоги до використання комп'ютерної техніки на підприємстві, безпосередньо й охорона праці при роботі з комп'ютером.

Вимоги до приміщення. Приміщення, в яких планується установка та подальша робота з комп'ютером, повинні відповідати проектній документації будинку, погодженій з уповноваженими державними органами. Крім того, роботодавець повинен враховувати санітарні нормативи освітлення, вимоги до параметрів мікроклімату (температура, відносна вологість), ступеня і сили вібрації, звукового шуму і вогнестійкості приміщення, а також характеристики електромагнітного, ультрафіолетового та інфрачервоного полів. Конкретні показники зазначених санітарних норм див. в Державних санітарих правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98, затверджених Постановою Головного державного санітарного лікаря України №7 від 10 грудня 1998 року.

Правила поширюються на умови й організацію праці при роботі з візуальними дисплейними терміналами (ВДТ) усіх типів вітчизняного та зарубіжного виробництва на основі електронно-променевих трубок (ЕПТ), що використовуються в електронно-обчислювальних машинах (ЕОМ) колективного використання та персональних ЕОМ (ПЕОМ).

Так, наприклад, роботодавцю заборонено встановлювати комп'ютери в приміщеннях, розташованих у підвалах будинків. Для уникнення можливих аварій та замикань, поряд з приміщеннями, де вестиметься робота з комп'ютером (над чи під ними), також не дозволяється проведення робіт, що потребують здійснення надмірно вологих технологічних процесів. Відповідне приміщення повинно бути укомплектоване системами центрального або індивідуального опалення, кондиціонування чи вентиляції повітря. Але при установці зазначених систем, необхідно переконатись, що батареї опалення, водопровідні труби, вентиляційні кабелі тощо, надійно сховані під захисними щитками, які перешкоджатимуть можливому потраплянню робітника під напругу.

У кожній кімнаті, де обладнуватимуться робочі місця співробітників, що працюватимуть на комп'ютері, повинні бути наявні елементи природного та штучного освітлення. При цьому, на вікнах слід встановити легко регульовані жалюзі чи штори, які дозволять працівникам коригувати рівень освітлення в приміщенні. Бажано розмістити комп'ютери в кімнаті таким чином, щоб світло потрапляло на екрани моніторів з півдня чи північного сходу. З метою досягнення максимального рівня безпечності і охорони праці при роботі з комп'ютером, виробничі приміщення необхідно обладнати аптечками першої медичної допомоги, системами автоматичної пожежної сигналізації і вогнегасниками. В приміщенні, в якому разом працюють 5 або більше комп'ютерів, на видимому місці установлюється службовий вимикач, який у разіпотреби дозволить повністю відключити електричне живлення кімнати.

Вимоги до особистого робочого місця працівника. Роботодавець, який використовує найману працю робітників, повинен забезпечити відповідність їхніх робочих місць комфортним та безпечним умовам. Розмір одного робочого місця має становити не менше 6 квадратних метрів. При необхідності, суміжні робочі місця співробітників, що працюють з комп'ютером, слід розділити перегородками висотою до 2 метрів. При визначенні достатнього розміру приміщення і робочого місця на одну особу необхідно додатково враховувати

шафи, сейфи, тумби або інші предмети меблів чи обладнання, які знаходяться в кімнаті. На столі працівника можливо розмістити допоміжні для роботи пристрої (принтери, колонки, сканери), а також місця для зберігання документів, за умови, що це не обмежуватиме видимість екрану і не заважатиме працівнику.

У разі надмірного шуму чи вібрації технічного обладнання, роботодавець повинен забезпечити працівників антивібраційними килимками. Робочий стілецьспівробітника має бути підйомно-поворотним, легко регульованим за висотою тазабезпечувати належну підтримку та зручне положення спини і хребта особи. Щодня необхідно проводити вологе прибирання приміщення, та очищати робоче місце та безпосередньо монітор комп'ютера від запиленості.

На підприємстві забороняється: проводити ремонт та технічне обслуговування комп'ютера за робочим місцем працівника; самочинно ремонтувати або намагатись здійснити технічне налагодження комп'ютера без залучення компетентних спеціалістів; складувати на робочому місці зайві документи, деталі та предмети, що не потрібні для роботи; використовувати монітори з нечітким зображенням та монітори, у яких наявні поламки екрану; працювати з матричним принтером без антивібраційного покриття та зі знятою кришкою. Допускати до роботи осіб, які не пройшли затверджений на підприємстві курс охорони праці для роботи з комп'ютером, не дозволяється.

Соціальні та профілактичні засоби захисту робітників, які працюють з комп'ютером. При прийнятті на роботу кожна особа має пройти лікарський огляд. Окрім того, при подальшій трудовій діяльності в компанії, така особа підлягає регулярному лікарському огляду не рідше ніж раз на 2 роки. Обов'язковим є проходження таких лікарів як терапевта, невропатолога та офтальмолога.

В компанії мають бути чітко встановлені перерви для відпочинку працівників (окрім обідньої), як правило, тривалістю 10-15 хвилин раз на годину або дві, в залежності від складності роботи.

В будь-якому випадку, роботодавець повинен передбачити такий розпорядок роботи на підприємстві, щоб час неперервної роботи з комп'ютером

був не більше ніж 4 години. Додатково, для збереження належного рівня здоров’я та професійної придатності робітників, рекомендується виділити на підприємстві окреме побутове приміщення для перепочинку працівників і зняття ними нервово- емоційного напруження, що виникає при роботі з комп’ютером.

5.2 Загальні вимоги безпеки при роботі з комп'ютером та іншою оргтехнікою

До самостійної роботи з комп'ютером, ноутбуком, принтером, ксероксом, сканером, плазмовою панеллю, *LCD*-дисплеєм та іншою оргтехнікою допускаються особи, які досягли 18 річного віку, пройшли медичний огляд, ознайомлені з інструкцією з охорони праці при роботі з оргтехнікою, не мають протипоказань за станом здоров'я.

Під час роботи на комп'ютері та іншій оргтехніці на робітники можуть впливати наступні небезпечні та шкідливі фактори:

електрострум і випромінювання;

перенапруження зору під час роботи з електронними пристроями, монітором, особливо при нераціональному розташуванні екрана по відношенню до очей.

3. Освітлювальні установки повинні забезпечувати рівномірне освітлення іне повинні утворювати засліплюючих відблисків на клавіатурі, а також на екрані монітора за напрямом очей.

При роботі з комп'ютером, принтером, ксероксом та іншою периферійною технікою не допускається розташування робочого місця в приміщеннях без природного освітлення, без наявності природної або штучної вентиляції.

Робоче місце з комп'ютером та оргтехнікою повинно розміщуватися на відстані не менше 1м від стіни, від стіни з віконними отворами - на відстані не менше 1,5 м.

Кут нахилу екрана монітора або ноутбука по відношенню до вертикалі повинен складати 10-15 градусів, а відстань до екрана – 500-600 мм.

Кут зору екрана повинен бути прямим і становити 90 градусів.

Для захисту від прямих сонячних променів повинні передбачатися сонцезахисні пристрої (плівка з металізованим покриттям, регульовані жалюзі з вертикальними панелями та ін).

Освітлення повинно бути змішаним (природним та штучним).

У приміщенні кабінету і на робочому місці необхідно підтримувати чистоту і порядок, проводити систематичне провітрювання.

Про всі виявлені під час роботи несправності обладнання необхідно доповісти керівнику, у випадку поломки необхідно припинити роботу до усунення аварійних обставин. При виявленні можливої небезпеки, попередити оточуючих та негайно повідомити керівнику; утримувати в чистоті робоче місце,не захаращувати його сторонніми предметами.

Про нещасний випадок очевидець, працівник, який його виявив, або сам потерпілий повинні доповісти безпосередньо керівникові установи і вжити заходів з надання медичної допомоги.

Особи, винні в порушенні вимог, вимагаємих даною інструкцією зохорони праці при роботі з комп'ютером, принтером, ксероксом та іншою оргтехнікою, притягаються до дисциплінарної відповідальності у відповідності з чинним законодавством.

5.3 Вимоги безпеки перед початком роботи з комп'ютером (ноутбуком) та іншою оргтехнікою

1. Оглянути і переконатися у справності обладнання, електропроводки.У разі виявлення несправностей, до роботи не приступати. Повідомити про цекерівника і, тільки після усунення несправностей і його дозволу, приступити до роботи.

2. Перевірити освітлення робочого місця, за необхідності, вжити заходів до його нормалізації.

3. При виявленні будь-яких несправностей, комп'ютер та оргтехнікуне вмикати і негайно повідомити про це головному з охорони безпеки.

4. Ретельно провітрити приміщення з персональним комп'ютером та оргтехнікою, переконатися, що мікроклімат у приміщенні знаходиться в допустимих межах: температура повітря в холодний період року 22-24°С, в теплий період року – 23-25°С, відносна вологість повітря – 40-60%.

5. Включити монітор і перевірити стабільність і чіткість зображення на екрані, переконатися у відсутності запаху диму від комп'ютера та оргтехніки.

5.4 Вимоги безпеки під час роботи з комп'ютером та іншою оргтехнікою

Вмикайте і вимикайте комп'ютер, ноутбук та іншу оргтехніку тільки вимикачами, забороняється проводити вимкнення витягуванням вилки з розетки.

Забороняється знімати захисні пристрої з обладнання і працювати без них.

Не допускати до комп'ютера та оргтехніки сторонніх осіб, які неберуть участі в роботі

Забороняється переміщати та переносити системний блок, монітор, принтер, будь-яке обладнання, яке знаходиться під напругою.

Забороняється під час роботи пити будь-які напої, приймати їжу.

Забороняється будь-яке фізичне втручання у пристрій комп'ютера, принтера, сканера, ксерокса під час їх роботи.

Забороняється залишати включене обладнання без нагляду.

Забороняється класти предмети на комп'ютерне обладнання, монітори, екрани та оргтехніку.

Суворо виконувати загальні вимоги з електробезпеки та пожежної безпеки.

Під час усунення застрявання паперу на ксероксі чи принтері, задля уникнення ураження електрострумом, необхідно відключити обладнання від електромережі. Необхідно також вимикати обладнання від мережі при тривалому простої.

Під час регламентованих перерв, з метою зниження нервово- емоційного напруження, стомлення зорового аналізатора, усунення впливу гіподинамії та гіпокінезії, запобігання розвитку познотонічного стомлення, слід виконувати комплекси вправ для очей або організовувати фізкультурні паузи.

Комп'ютер, будь-які його периферійні пристрої, оргтехніку необхідно використовувати у суворій відповідності з експлуатаційною документацією до них.

Під час виконання роботи необхідно бути уважним, не звертати уваги на сторонні речі.

Про всі виявлені несправності та збої в роботі апаратури необхідно повідомити безпосередньо інженера з обслуговування комп'ютерної техніки.

5.5 Вимоги безпеки після закінчення роботи з комп'ютером, принтером, ксероксом, сканером та іншою оргтехнікою

Вимкнути комп'ютер, ноутбук, телевізор, плазмову панель, *LCD*-екран, принтер, ксерокс, сканер, колонки та іншу оргтехніку від електромережі, для чого необхідно вимкнути тумблери, а потім акуратно витягнути штепсельні вилки з розетки.

Протерти зовнішню поверхню комп'ютера чистою вологою тканиною. При цьому не допускайте використання розчинників, одеколону, препаратів в аерозольній упаковці.

Прибрати робоче місце. Скласти диски у відповідне місце зберігання.

Ретельно провітрити приміщення з персональним комп'ютером таіншою оргтехнікою.

5.6 Вимоги техніки безпеки та безпеки життєдіяльності в аварійних ситуаціях при роботі з комп'ютером та іншою оргтехнікою

Якщо на металевих частинах обладнання виявлено напругу (відчуття струму), заземлюючий провід обірваний, необхідно вимкнути обладнання, негайно доповісти керівникові про несправності електрообладнання і без його вказівки до роботи не приступати.

При припиненні подавання електроенергії, вимкнути обладнання.

При появі незвичного звуку, запаху паленого, мимовільного відключення комп'ютера та оргтехніки, негайно припинити роботу і поставити до відома керівника.

При виникненні пожежі негайно вимкнути обладнання, знеструмити електромережу за винятком освітлювальної мережі, повідомити про пожежу всім працюючим і приступити до гасіння осередку пожежі наявними засобами пожежогасіння.

При нещасному випадку необхідно, насамперед, звільнити потерпілого від травмуючого фактора, звернутися до медпункту, зберегти, по можливості, місце травмування в тому стані, в якому воно було на момент травмування. При звільненні потерпілого від дії електроструму слідкуйте за тим, щоб самому не опинитися в контакті з токоведучою частиною та під напругою.

5.7 Вимоги безпеки в аварійних ситуаціях при роботі з комп'ютером

При виникненні несправності в роботі комп'ютера або порушення цілісності заземлення його корпусу слід вимкнути прилад і відключити його від електромережі. Роботу продовжити тільки після усунення несправності.

При загорянні працівник повинен:

- негайно відключити електроприлад від мережі живлення;
- повідомити про те, що трапилась пожежа адміністрації установи і в найближчу пожежну частину за телефоном 101;
- приступити до гасіння осередка загоряння за допомогою первинних засобів пожежогасіння.

При отриманні травми слід надати першу допомогу, при необхідності відправити його до лікувальної установи і повідомити про це адміністрацію.

Висновки до п’ятого розділу

Сформовані вимоги безпеки, а саме:

- вимоги безпеки при роботі з комп'ютером та іншою оргтехнікою;

- вимоги безпеки перед початком роботи з комп'ютером;
- вимоги безпеки під час роботи з комп'ютером, ноутбуком;
- вимоги безпеки після закінчення роботи з комп'ютером;
- вимоги техніки безпеки та безпеки життєдіяльностів аварійних ситуаціях при роботі з комп'ютером та іншою оргтехнікою;
- вимоги безпеки в аварійних ситуаціях при роботі з комп'ютером.

ЗАГАЛЬНИЙ ВИСНОВОК

Сьогодні надзвичайно важливо використовувати цифрові технології у бізнесі, адже саме онлайн-запис полегшує спілкування з клієнтами та покращує обслуговування. Для успішного бізнесу варто слідкувати за трендами у сфері онлайн-бронювання, адже це прямо впливає на прибуток. Використання технологій планування зустрічей стає ключем до привертання та утримання уваги клієнтів.

Розробка веб-системи, яка надаватиме можливість записатися на зустріч онлайн є досить актуальною задачею. Такий підхід значно пришвидшить процес обрання варіанту зустрічі та запису, дозволить скоротити використання людських ресурсів.

В результаті виконання дипломної роботи було створено базу даних та розроблено веб-сайт для онлайн бронювання зустрічей.

У ході дипломною роботи було:

1. Проведено аналіз: досліджено предметну область, проаналізовано існуючі продукти-аналоги, виділено їх переваги та недоліки.

2. Сформовано вимоги до розроблюваного застосунку.

3. Спроектовано базу даних та архітектуру додатку.

4. Обрано технології для програмної реалізації проекту.

5. Розроблено веб-застосунок відповідно до технічного завдання.

Систему було спроектовано та розроблено з урахуванням можливостей для розширення. В подальшому, можна реалізувати функції перегляду історії відвідувань, додавання відгуків про зустрічі, тощо, або застосунок може бути інтегрований у вже існуючу систему.

Основні переваги використання системи онлайн бронювання включають цілодобову доступність для клієнтів, спрощене керування бронюванням, автоматичне планування та нагадування, а також покращений досвід роботи з клієнтами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що нового на ринку онлайн-планувальників [Електронний ресурс]. – Режим доступу: https://easyweek.com.ua/specifika-rinku-sistem-onlajn-zapisu.html
2. Робота з MongoDB, її основні функції [Електронний ресурс]. – Режим доступу: https://www.freecodecamp.org/news/learn-mongodb-a4ce205e7739/
3. Install MongoDB Community Edition on Windows [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/
4. Java, огляд мови [Електронний ресурс]. – Режим доступу: https://www.java.com/ru/download/help/whatis_java.html
5. Ознайомлення з синтаксисом Java [Електронний ресурс] . – Режим доступу: https://metanit.com/java/tutorial/
6. Визначення підходу програмування для обраної мови програмування [Електронний ресурс]. – Режим доступу: https://highload.today/oop-v-java-chetyre-printsipa-s-primerami/
7. Оптимізація зробленого продукту [Електронний ресурс] . – Режим доступу: https://www.ibm.com/docs/ru/aix/7.2?topic=monitoring-java-performance-guidelines
8. Тестування, та як його робити [Електронний ресурс]. – Режим доступу: https://www.baeldung.com/java-test-driven-list
9. Класифікація веб-сайтів [Електронний ресурс]. – Режим доступу: https://sites.google.com/site/yaremusinform/home
10. Communication: Online vs. Face-to-Face Interactions [Електронний ресурс]. – Режим доступу: https://psychminds.com/communication-online-vs-face-to-faceinteractions/
11. What is CSS, and why is it important? [Електронний ресурс]. – Режим доступу: https://www.bigcommerce.com/ecommerce-answers/what-css-and-whyit-important/

12. JavaScript (JS) [Електронний ресурс]. – Режим доступу: https://www.techopedia.com/definition/3929/javascript-js
13. Javascript [Електронний ресурс]. – Режим доступу: https://scriptdev.ru/ts/007/
14. What is TypeScript and why should you use it? [Електронний ресурс]. – Режим доступу: https://learn.coderslang.com/0056-what-is-typescript-and-why-shouldyou-use-it/
15. Angular [Електронний ресурс]. – Режим доступу: https://www.typescriptlang.org/docs/handbook/angular.html
16. 8 Proven Reasons You Need Angular for Your Next Development Project [Електронний ресурс]. – Режим доступу : ttps://www.grazitti.com/blog/8-proven-reasons-you-need-angular-for-your-next-development-project/51
17. Вдовичин Т. Я., Лазурчак Л. В. Проектування інформаційно пошукових систем як засіб використання сучасних технологій. Вчені записки ТНУ імені В.І. Вернадського. Серія: Технічні науки. 2022. Т. 33(72). № 4. С. 66–71.
18. Ізмайлова О. В. Проектування інформаційних систем: навч. посіб. Київ: КНУБА, 2022. 88 с.
19. What is API: Definition, Types, Specifications, Documentation [Електронний ресурс]. – Режим доступу : https://www.altexsoft.com/blog/engineering/what-isapi-definition-types-specifications-documentation/
20. REST API [Електронний ресурс]. – Режим доступу: https://blog.skillfactory.ru/glossary/rest-api/Murray N., Coury F., Lerner A., Taborda C. ng-book: The Complete Guide to Angular, 5th edition. Technical Editor: Frode Fikke, 2018. 626 p.
21. What Are Angular Services and Why Should You Use them? [Електронний ресурс]. – Режим доступу : https://chudovo.com/what-are-angular-services-andwhy-should-you-use-them/
22. What is Bootstrap and Why you should use it in Web Development [Електронний ресурс]. – Режим доступу : https://www.yogihosting.com/what-isbootstrap/
23. Magolan G., Bell J., Guijarro D., Peretti A., Housley P. Nest.js: A Progressive Node.js Framework, Kindle Edition. Bleeding Edge Press, 2018. 350 p.

24. What is MongoDB? A quick guide for developers [Електронний ресурс]. – Режим доступу : https://www.infoworld.com/article/3623357/what-is-mongodb-aquick-guide-for-developers.html
25. Bradshaw S., Brazil E., Chodorow K. MongoDB: The Definitive Guide, 3rd Edition. O'Reilly Media, 2019. 514 p.
26. Page Regions [Електронний ресурс]. – Режим доступу: https://www.w3.org/WAI/tutorials/page-structure/regions
27. Про затвердження змін та доповнень Правил безпечної експлуатації електроустановок – Законодавство України [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0213-00#Text (дата звернення 15.05.2024).
28. 6 правил роботи за комп’ютером без шкоди для здоров'я. [Електронний ресурс]. – Режим доступу: https://pon.org.ua/novyny/5812-6-pravil-roboti-zakompyuterom-bez-shkodi-dlyazdorovya.html
29. Правила безпечної роботи на комп'ютері [Електронний ресурс]. – Режим доступу:https://www.pedcollege.kiev.ua/index.php/77robotakoledzhu/okhoronapratsi/589-pravyla-bezpechnoi-roboty-na-kompiuteri.html
30. Винокурова Л. Е., Васильчук М. В., Гаман М. В. Основи охорони праці: Підручн. для проф.-техп. навч. закладів. К. : Вікторія, 2001. 192 с.
31. Санітарно-гігєнічні норми [Електронний ресурс]. – Режим доступу: https://te.dsp.gov.ua/robota-v-ofisiosnovnisanitarno-gigiyenichnivymogy/
32. Басюркіна Н.Й., Свистун Т.В. «Оцінка науково-технічної ефективності: Методичні вказівки». Одеса: ОНТУ, 2023р. 18с